New Frontiers in String Algorithms and Nearest Neighbour Search

Negev Shekel Nosatzki

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2025

ABSTRACT

New Frontiers in String Algorithms and Nearest Neighbour Search

Negev Shekel Nosatzki

This thesis revolves arounds two main frameworks of problems: (i) **String Problems** such as the Longest Increasing Subsequence (LIS), Longest Common Subsequence (LCS) and the Edit Distance (ED); and (ii) **Approximate Nearest Neighbour Search Problems** under different metrics.

For **String Problems**, we show the following results:

- For ED, a constant approximation algorithm which runs in near-linear time. In particular, an algorithm for approximating the edit distance between two strings of length n in time $n^{1+\epsilon}$ up to a constant factor, for any $\epsilon > 0$.
- For parametrized LIS, the first sub-polynomial (in the parameter) approximation. In particular, we show that for any $n \in \mathbb{N}$ and $\lambda = o(1)$, there exists a (randomized) *non-adaptive* algorithm that, given a sequence of length n with $\mathsf{LIS} \geq \lambda n$, approximates the LIS up to a factor of $1/\lambda^{o(1)}$ in $n^{o(1)}/\lambda$ time.
- For LCS, we give the first sub-polynomial approximation in linear time.

For **Approximate Nearest Neighbour Search Problems**, we show the following results:

- The first instance of a simple, practical algorithm that provably leverages data-dependent hashing to improve upon data-oblivious LSH.

- We introduce the notion of metric embeddings into *similarity measures* over $\mathbb{R}_+^m$, and use such new embeddings to develop Approximate Nearest Neighbor Search (ANN) algorithms which improve the approximation factor exponentially on the (longstanding) state-of-the-art on two important metrics:
  - Edit distance over length-k strings: $\text{poly}(\log k)$ approximation;
  - ℓ_p over $\mathbb{R}^d$, for $p > 2$: $O(\log p)$ approximation (known to be asymptotically optimal in the relevant models of computation).

While the two frameworks might apriori feel unrelated, recent progress have shown they are ultimately intertwined. First and foremost, both frameworks have proven to be core problems in the area of fine-grained complexity, where conditional hardness has been shown under the Strong Exponential Time Hypothesis (SETH), among other conjectures. Second, the algorithmic techniques for solving the frameworks are inter-related, and we will use Similarity Search techniques to solve string problems and vice versa.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I am deeply grateful to my advisors, Alexandr Andoni and Tal Malkin. Alexandr's expertise, depth of insight, and unwavering support have helped shape my research direction. Alex, thank you for being there in every step of the way, going above and beyond your role as an advisor, helping me both personally and professionally and dedicating countless hours in brainstorming together unique concepts and ideas and bringing together those ideas to light. Tal, your intellectual rigor, critical feedback, and constant encouragement were instrumental in navigating this journey. I feel fortunate to have had the opportunity to work with both of you and I am hopeful we will continue working together in the future.

I would also like to thank my other coauthors for their collaboration, sharp ideas, and shared enthusiasm for the problems we tackled together. Working with you was not only intellectually enriching but also a genuine pleasure.

To the anonymous reviewers of our work: thank you for your thoughtful feedback, challenging questions, and suggestions that helped refine and improve this research.

Finally, and most profoundly, I thank my family. To my wife: your love, patience, and belief in me never wavered, even through the most difficult times. Your support made this possible. To my kids: thank you for bringing joy and perspective to my days, and for being a constant reminder of what truly matters. And to my parents: thank you for your unconditional love, encouragement, and for teaching me the values that carried me through this journey.

Chapter 1

Introduction

This thesis revolves around two main frameworks of problems: *String Problems* and *Similarity Search Problems*, as well as their fundamental connection to the area of Fine-Grained Complexity. The main fine-grained hardness Conjecture for these problems is SETH:

Conjecture 1 (Strong Exponential Time Hypothesis (SETH))**.** *For every $\epsilon > 0$ there exists a $k \geq 3$ such that k-SAT over n variables cannot be solved in time $O(2^{(1-\epsilon)n})$.*

1.1 String Problems (Part I)

We study three fundamental string problems:

- The **Longest Increasing Subsequence (LIS)** problem.
- The **Longest Common Subsequence (LCS)** problem.
- The **Edit Distance (ED)** problem.

The LIS of a sequence x is the longest subsequence of x which is strictly increasing. The LCS of two strings x and y over alphabet Σ is simply their longest (not necessarily contiguous) common subsequence. The edit distance is defined as the minimum number of character deletions, insertions, and substitutions required to transform x into y. The LIS problem is also equivalent to a special case of LCS where one of the strings is known to be monotonic. All three are classical problems with dynamic programming solutions, resulting in quadratic time for LCS and ED, and near-linear time for LIS. All the aforementioned problems are also very fundamental with wide spectrum of applications, and have been subject to a plethora of studies in the past few decades and especially in recent years.

Despite significant research attempts over decades, little progress was obtained. For LIS, the classical $O(n \log n)$ Patience Sorting algorithm [113; 149] was slightly improved to $O(n \log \log n)$ [86], and for LCS and ED, a $O(n^2/\log^2 n)$ run-time algorithm [150] remains the fastest one known to date. See also the surveys of [171] and [187]. With the emergence of the fine-grained complexity field, researchers crystallized the reason why beating quadratic-time is hard by connecting it to SETH [37] (and even more plausible conjectures [3]).

In the quest for faster algorithms, researchers started studying whether we can settle for approximation algorithms. An early version of estimating the LIS underpins one of the first sublinear-time algorithms: to test whether an array is sorted, or monotonically increasing [95] — i.e., whether the length of the LIS is n or is at most $(1-\epsilon)n$. [95] gave a $O(\log n/\epsilon)$ time algorithm, and this running time was later shown to be tight [5; 97]. Since then, there have been numerous influential results on testing monotonicity and other similar properties; see, e.g., [91; 180; 96; 69; 49;

52; 178; 46; 25; 194] and the book [106]. Moreover, LIS is often a subroutine in LCS algorithms: for example, when strings are only mildly repetitive [108, Chapter 12], or more recently in approximation algorithms [110; 173].

While monotonicity results focus on the case when $\mathsf{LIS} \approx n$, the case when $\mathsf{LIS} \ll n$ has seen much less progress. The first result for this regime [189] shows how to $(1+\epsilon)$-approximate the length when the LIS is still large: they distinguish the case when $\mathsf{LIS} \geq \lambda n$ from the case when $\mathsf{LIS} \leq (\lambda - \epsilon)n$ (for any $\lambda > 0$) using $(1/\epsilon)^{O(1/\epsilon)} \log^{O(1)} n$ time, which only gives a $(1+\epsilon)$-factor approximation in truly sublinear time if $\lambda = \Omega(\log\log n / \log n)$. For arbitrary $\lambda < 1$, assuming that $\mathsf{LIS} \geq \lambda n$, [185] gave an algorithm that achieves $O(1/\lambda^3)$-factor approximation of LIS in $\tilde{O}(\sqrt{n}/\lambda^7)$ time.[1] Recently, [161] improved upon this result by presenting an algorithm with approximation $O(1/\lambda^\epsilon)$ and runtime $O(n^{1-\Omega(\epsilon)}(\log n/\lambda)^{O(1/\epsilon)})$. Independently, [172] obtained a non-adaptive $O(1/\lambda)$-approximation algorithm using $\tilde{O}(\sqrt{r}/\lambda^2)$ queries, where r is the number of distinct values. The authors of [172] also proved a lower bound, showing that any *non-adaptive* algorithm that estimates the LIS to within an additive error of ϵn requires $(\log n)^{\Omega(\log(1/\epsilon))}$ queries.

To put the above results into context, contrast LIS with the problem of estimating the weight of a binary vector of length n: when the weight is at least λn, we can approximate the weight up to a factor of $1+\epsilon$ by sampling $O_\epsilon(1/\lambda)$ positions, which is optimal. So far, one cannot rule out that a similar performance is achievable for

[1] The formal definition of an approximation algorithm here is one that has to return $\widehat{\mathsf{LIS}}$ such that $\widehat{\mathsf{LIS}} \leq \mathsf{LIS}$, and the approximation factor is $\mathsf{LIS}/\widehat{\mathsf{LIS}}$. Equivalently, one can conceptualize an α-approximation algorithm (for $\alpha \geq 1$) as one that can distinguish the case when $\mathsf{LIS} \geq \lambda n$ from the case when $\mathsf{LIS} < \lambda n/\alpha$.

estimating LIS too, in fact for λ as large as $1/\log n$. The aforementioned prior results not only have approximation factors that are polynomial in $1/\lambda$ but also time / sample complexities that are worse by a polynomial factor in n or $1/\lambda$. Hence the following guiding question remains open:

Question 2. *Can we estimate the length of LIS in essentially the time needed to estimate the weight of a binary vector?*

For LCS, a major open problem is finding the best approximation for linear-time algorithms. A trivial $\sqrt{n}$-factor approximation was only recently improved to $\approx n^{0.497}$ [112] and then to $n^{0.4}$ [61].

In parallel, a λ-parametrized version was developed (in a similar fashion to LIS from above), which gives a subquadratic solution with improved approximation when the longest common subsequence is promised to be above a certain threshold [185].

In contrast to the slow progress for LIS and LCS, the progress for ED has been more significant. A linear-time $\sqrt{n}$-factor approximation follows immediately from the exact algorithm of [198; 164; 147], was followed by improved approximations: first to $n^{3/7}$ [39], then to $n^{1/3+o(1)}$ [42], and to $2^{\tilde{O}(\sqrt{\log n})}$ [30] (based on the ℓ_1 embedding of [175]). In the regime of $O(n^{1+\epsilon})$-time algorithms, the best approximation prior to our work is $(\log n)^{O(1/\epsilon)}$ [21]. Predating some of this work was the sublinear-time algorithm of [41] achieving n^{ϵ} approximation when d is large. Attempts to improve the approximation to a constant-factor yielded the first truly sub-quadratic quantum algorithm for approximating edit distance [50] as well as a classical one [70]. This was followed by a series of advances [10; 105; 139; 56]. Our work completes such line of work, providing a near-linear time, constant factor approximation algorithm [28]

which is presented hereforth.

Despite such recent success, there are many questions which remain open and the goal for finding the fastest algorithms with the best approximation still stands.

Our Contributions. For ED, our main result is a $n^{1+\epsilon}$ algorithm for computing the edit distance up to a constant approximation.

Theorem 3 (ED Approximation. See Chapter 3)**.** *There is a randomized algorithm that, for any $\epsilon > 0$, $n \geq 1$, and alphabet Σ, and any two strings $x, y \in \Sigma^n$, approximates the edit distance between x and y in expected $O(n^{1+\epsilon})$ time up to $f(1/\epsilon)$-factor approximation, for some universal non-decreasing function $f(\cdot)$.*

This result was published in [28] and is described in full in Chapter 3. While we do not derive the function $f(1/\epsilon)$ explicitly, we note that it is doubly exponential in $1/\epsilon$.

To obtain this, our algorithm deviates from previous approaches in several ways. One notable change is that instead of the more standard recursion algorithm, our top-level algorithm iterates bottom–up over all substring lengths $w = \gamma, \gamma^2, \ldots n$, where $\gamma = n^\epsilon$, and for each w computes a new metric $\mathfrak{D}_w$ using $\mathfrak{D}_{w'}$, where $w' = w/\gamma$. Conceptually we do so in two phases. First, we build another metric on w-length strings $\mathtt{ad}_w$, accessible via a fast distance oracle, which align-approximate[2] $\mathtt{ed}$. Second, equipped with a fast oracle for $\mathtt{ad}_w$ (itself using $\mathfrak{D}_{w'}$), we build an "efficient representation" for the entire metric, while using only $n^{1+O(\epsilon)}$ calls to $\mathtt{ad}_w$ oracle. Naturally, this "efficient representation" will not be able to capture the entire $\mathtt{ad}_w$

[2] Align-approximate is intuitively an approximation that is good enough for an optimal alignment. See Definition 18 for a formal definition.

metric (that would require $\gg n$ query complexity), but it will capture just enough to preserve the edit distance between x and y. Then we build an efficient distance oracle for this efficient representation, which will yield the desired metric $\mathfrak{D}_w$. Figure 1 in Section 3.2 showcases the high-level flow of this algorithm. An interesting anecdote is that the process of creating such representation is done by defining "similarity embedding" of substrings, connected to Part II of this research.

For LIS, our work comes close to answering Question 2 in the affirmative. In particular, considering the λ-parametrized version, we obtain an algorithm that runs in time near-linear in $1/\lambda$, and achieves sub-polynomial in $1/\lambda$ approximation. We show the following:

Theorem 4 (LIS Approximation. See Chapter 4)**.** *For $n \in \mathbb{N}$, and any $\lambda = o(1)$, there exists a (randomized) non-adaptive algorithm that, given a sequence of length n with LIS $\geq \lambda n$, approximates the length of LIS up to a $1/\lambda^{o(1)}$ factor in $O\left(\frac{1}{\lambda} \cdot n^{o(1)}\right)$ time with high probability.*

This result was published in [29] and is described in full in Chapter 4. We find this result quite surprising, as one may guess that when $\lambda \approx 1/\sqrt{n}$ (which is the LIS length of a random permutation), one would need to read essentially the entire sequence (implying a $\Omega(1/\lambda^2)$ lower bound on the number of queries needed). On the contrary, when $\lambda \approx 1/\sqrt{n}$, our run-time (and hence sample complexity) in such regime is $n^{1/2+o(1)}$. This also implies a streaming LIS algorithm which nearly recovers the *streaming complexity* from [107; 94] (albeit with worse approximation).

Our algorithm improves upon prior work substantially in terms of both approximation and run-time: (i) we provide the first sub-polynomial approximation for LIS

in sub-linear time; and (ii) our *run-time complexity* essentially matches the trivial *sample complexity* lower bound of $\Omega(1/\lambda)$, which is required to obtain any non-trivial approximation of the LIS.

As part of our solution, we develop two novel ideas which may be of independent interest. First, we define a new Genuine-LIS problem, in which each sequence element may be either genuine or corrupted. In this model, the user receives unrestricted access to the actual sequence, but does not know a priori which elements are genuine. The goal is to estimate the LIS using genuine elements only, with the minimal number of tests for genuineness. The second idea, *Precision Tree*, enables accurate estimations for composition of general functions from "coarse" (sub-)estimates. *Precision Tree* essentially generalizes classical precision sampling, which works only for summations. As a central tool, the *Precision Tree* is pre-processed on a set of samples, which thereafter is repeatedly used by multiple components of the algorithm, improving their amortized complexity.

Finally, for LCS, our main result is the first sub-polynomial factor approximation algorithm for LCS, which runs in linear time.

Theorem 5 (LCS Approximation. See Chapter 5)**.** *LCS_n can be approximated up to $n^{o(1)}$-factor, with high probability, in $O(n)$ time.*

This result appears in [173] and is described in full in Chapter 5. The algorithm is obtained using an instantiation of LIS (sub-)algorithms from Theorem 4. While we do not track the exact approximation factor in our analysis, we note it is $\exp\left(\frac{\log(n)}{\log^c \log(n)}\right)$ for some absolute constant c. Interestingly, we could not find a way to significantly improve the bound above, obtainable in linear time, even if one can afford $n^{1.99}$

runtime. This relates to the issue of balancing of dense and sparse cases described in Chapter 4, which is the core component of our algorithm.

To improve the bound, one would need to find a more efficient way to solve the Block-LIS algorithm from Chapter 4.

One implication of this work is that an affirmative resolution to Question 2 would yield surprising results, and in particular, a near-linear time, $(1+\epsilon)$ approximation to LCS, which also provides near-linear ϵn additive approximation to the edit distance.

1.2 Nearest Neighbour Search (Part II)

In the classical problem of c-approximate Nearest Neighbor Search (ANN), we need to preprocess a dataset P of n points in some metric space $\mathcal{X}$ with distance function d, so that, given a query point $q \in \mathcal{X}$, we report a point $p \in P$ within distance $c \cdot \min_{p \in P} \mathsf{d}(q, p)$. We study the high-dimensional ANN, which is motivated both by practice and theory. ANN has a wide range of applications for various metrics which are typically high-dimensional; see, e.g., book chapters [88; 191; 197]. Furthermore, ANN, as well as the tools behind the algorithms, are core primitives in design of algorithms for high-dimensional questions, frequently arising in the study of algorithms for massive datasets; see, e.g., surveys [119; 9] and specifically recent work on, say, density estimation [75] or terminal embeddings [79].

Recently, SETH hardness bounds were used to better understand the ANN problem [184], showing that any polynomial space and preprocessing-time data-structure cannot yield a truly sub-linear query time for $1 + \epsilon$ approximation.

At a high level, algorithms for ANN roughly fall into two classes: direct algorithms,

and via metric embeddings.

Direct algorithms have been developed for a number of standard spaces, notably ℓ_2, ℓ_1, ℓ_∞, ℓ_∞-product, ℓ_p, and general norms.

Most such direct algorithms are based on *Locality Sensitive Hashing (LSH)*, where the idea is to hash the points such that the probability of collision is much higher for points which are close to each other (at distance $\leq r$ for some fixed radius r) than for those which are far apart (at distance $> cr$). Given such hash functions, one can retrieve near neighbors by hashing the query point and retrieving elements stored in buckets containing that point. If the probability of collision is at least p_1 for the close points and at most p_2 for the far points, the algorithm solves the d-dimensional (c, r)-ANN using $n^{1+\rho}$ extra space and dn^ρ query time, where $\rho = \log(1/p_1)/\log(1/p_2)$ [114].

The performance of an LSH family is thus determined by the value of the exponent ρ, for which we now know precise bounds for some core metrics. The original LSH paper [124] obtained $\rho = 1/c$ for the Hamming space. For the Euclidean space, subsequent research yielded a better exponent: $\rho = 1/c^2 + o(1)$ [90; 13]. These bounds are tight for both Hamming and Euclidean space, as shown by the lower bounds of [162; 174], thus completing this line of research.

Subsequent work showed that one can outperform LSH-based algorithms by using *data-dependent* hashing. Quantitatively speaking, the work of [17; 31] developed algorithms with space overhead $n^{1+\rho}$ and query time dn^ρ, where $\rho = 1/(2c-1) + o(1)$ for the d-dimensional Hamming space and $\rho = 1/(2c^2-1) + o(1)$ for Euclidean space. Note that, for large c, this is a quadratic improvement of the query time (with better space) over best possible LSH algorithms; the new bounds are known to be optimal

for data-dependent hashing [32; 23]. In addition to the quantitative improvement, the new space partitions are *qualitatively* different: they adapt to the geometry of a given dataset, opening space for a new type of algorithmic techniques for the ANN problem.

The second class of ANN algorithms leverages metric embeddings, and will be discussed later on in this chapter.

At a high level, the ANN part of this thesis adds the following contributions to ANN algorithms:

- Past work on DDH algorithms successfully made progress in asymptotic bounds, but has also proven to be particularly complex and far from practical. In contrast, we develop a simple DDH version of the most basic LSH algorithm for Hamming [124] (hence, it is likely much more practical that other DDH algorithm), which improves upon optimal LSH bounds. Incidentally, it is built on LSH Forest, a classic algorithm often used in practice.

- We develop a new method for designing ANN, based on embeddings into similarity measures, thus extending and qualitatively improving upon a very successful program of metric embedding approach to ANN, precisely where it has proven to be limited for designing data-dependent ANN. We apply this method on 2 central and well-studied metrics: ℓ_p and edit distance, improving the approximation factor exponentially in each one.

1.2.1 Theory and practice of ANN

The idea of data-dependent hashing has become quite popular in practice in recent years [204; 203], though it is for the following subtly different reason: datasets that arise in practice are hypothesized to not be "worst case", and instead exhibit some structure, such as *low intrinsic dimension*. Algorithms used in practice try to exploit (implicitly or explicitly) precisely this additional structure. Let us remark that most of these algorithms do not have any guarantees, on correctness or performance (but see the related work section for some recent examples that do). In fact, it has become an important challenge to bridge this gap between theoretical and practical algorithms (see, e.g., a National Research Council report [85, Section 5], or the survey [203, Section 7]).

While data-dependent hashing algorithms come close to bridging the gap—they are empirically better in practice and are now proven to be better in theory—yet, the gap persists: the theoretical and practical algorithms are still remarkably different. In particular, the data-dependent algorithms from [17; 31] are quite far from the aforementioned practical algorithms, or being practical at all. One step towards practice was undertaken in [16], who gave an efficient implementation of one of the components of [31]. This component is however data-*independent*: it is a better LSH scheme for a dataset that is assumed to already have a nice structure (loosely speaking, it looks like a random set). In particular, the algorithm of [16] has no data-dependent steps, and hence it remains an open question to develop practical data-dependent algorithms, with theoretical guarantees á la [31].

One immediate barrier for making [31; 17] more practical is the fact that these

algorithms use a certain decomposition procedure that partitions a *worst case* dataset into several subsets which are *pseudo-random*. This procedure is precisely the data-dependent component of the algorithm. This decomposition is not particularly efficient, and generously contributes to the $n^{o(1)}$ factors in the bounds for space and query time. Making such reductions for finite subsets of $\mathbb{R}^d$ more efficient would be of independent interest—in particular, for problems where we know algorithms with better performance for the pseudo-random datasets than the worst-case datasets (e.g., [200; 131; 132; 7]).

In this thesis, we develop new, simple data-dependent algorithms that are *provably* better than data-oblivious LSH algorithms, even for *worst case* instances. Our first work gives algorithms that while not improving the state-of-the-art bounds of [31] quantitatively (in fact they are worse), the main advantage is simplicity and relation to algorithms used in practice. Our second line of work provides substantial improvement to long-standing theoretical bounds (as will be described later).

1.2.2 LSH Forest

Both our aforementioned lines of work are based on *LSH Forest*, a heuristic introduced in [43] and is very popular in practice (see, e.g., [47]). It can be applied as a black box to any LSH hash family.

The construction of LSH Forest data-structured is done by building n^{ρ} LSH Trees, where each tree is associated with a randomly sampled hash function from the family, and the tree itself is simply a (logical) prefix tree on the set of all labels, with each leaf corresponding to a labeled point in the dataset.

We note here that direct implementation of the above LSH Forest does not lead

to improved algorithms over data-independent ones, and in fact has slightly worse bounds. The main motivation for LSH Forest popularity stems from its simple and practical nature.

In this work, we show however, that careful modifications and instantiation of LSH Forest can lead to improved theoretical bounds while keeping its practical nature.

LSH Forest will be useful for us for multiple applications (which will be described later). For now, we will show how to modify it to obtain improved theoretical bounds for Bit-Sampling LSH. Before describing our improved algorithm, let us briefly recall the Bit Sampling LSH algorithm from [124; 114]. We hash an n-point dataset $P \subseteq \{0,1\}^d$ using a hash function $h(x) = x_I$, where x_I is a restriction of the point $x \in \{0,1\}^d$ on a set of indices $I \subseteq [d]$. The set I is chosen to be a random subset of $[d]$ of fixed, carefully chosen size. Given a query $q \in \{0,1\}^d$, we retrieve all the data points $p \in P$ such that $h(p) = h(x)$ and check them all until we find a point within distance cr from q. In order to boost the probability of success to constant, we repeat the above construction n^ρ times, where $\rho = \frac{1}{c}$. It is not hard to construct an example, where we must take ρ to be at least $\frac{1}{c} - o(1)$ and, moreover, this bound is tight for *any* data-independent LSH for the Hamming distance [174].

One of our new, improved algorithm is based on the LSH Forest algorithm, which proceeds as follows when instantiated to the Bit Sampling LSH. We choose a random permutation of the coordinates $\pi : [d] \to [d]$ and apply it to each vector $x \in P$. Then we build a *trie* on the permuted vectors.[3] For a given query q, we descent down the trie until we reach precisely one leaf and compare against the point stored in that

[3]This can be also seen as applying kd-tree on P with for a *random order* of the coordinates, stopping the splitting until we have exactly one point.

leaf. As before, we boost the probability of success by building n^ρ such tries.

While the above algorithm still requires $\rho \geq 1/c$ in the worst case, here is how we modify it to bypass the $1/c$ bound. For a constant parameter $k = O(1)$, and for every tree node v, we choose a fixed set of k *pivots* T_v. Then, when answering a query, while descending down the tree, we check against all the pivots T_v of every node v we visit on the way to the leaf. Overall, we compare the query point with at most dk data points. The following theorem summarizes the main result of this paper assuming that there is only the near neighbor within distance cr from the query.

Theorem 6 (Improved LSH Forest for bit-sampling, informal. See Chapter 6)**.** *For every $k \geq 1$, there exists a procedure of choosing k pivots in each node of the tree of* LSH *Forest, such that, in order to get a data structure for (c, r)-*ANN *for the Hamming distance with constant probability of success, it is enough to sample $O(n^\rho)$ tries, where:*

$$\rho \leq \frac{1}{\ln 4} \cdot \frac{1}{c} \cdot \left(1 + O\left(\frac{1}{k}\right)\right) + O\left(\frac{1}{c^2}\right).$$

Choosing k to be large enough, the exponent is $\rho = 0.73/c + O(1/c^2)$. The resulting query time is $O(n^\rho d^2 k)$ and space is $O(n^{1+\rho} k + nd)$.

This result was published in [33] and is described in full in Chapter 6. Our resulting bound of, essentially, $\rho \approx \frac{1}{\ln 4 \cdot c}$ corresponds to the bound that follows from using the Bit Sampling LSH of [124; 114] on *a random dataset* in $\{0,1\}^d$. Hence the algorithm can be seen as a worst-case–to–random-case reduction in the case of Bit Sampling LSH. While there exist space partitions that are better than Bit Sampling on a random dataset [17; 31; 16], they are less efficient in practice, and, in particular require a *larger number of parts*, requiring a tree with a large branching factor. This seems required at least in some cases: in particular, for the *Euclidean space*, [16]

prove that, for any space partition where ρ differs from the optimal exponent by ϵ, the number of parts must be exponential in $1/\epsilon$. We conjecture a similar phenomenon happens for the Hamming case; indeed, in all the *known* constructions, we need at least 10^6 parts to improve upon the Bit Sampling for random instances, for $c = 2$. Furthermore, in all known space partitions, the decoding time is proportional to the number of parts, and hence it is a very important consideration in practice.

1.2.3 Embeddings into similarity measures for ANN

While LSH complexity is fairly well-understood for the high dimensional Euclidean space, among several other metrics (see also survey [18] references and more recent [145; 27]), there are many others for which the bounds are still unknown. Similarity search for string problems like the ED and the LCS are important problems with many applications. However, despite their importance, progress has been slow, and was mostly focused on embedding of such hard metrics into "easier ones". These techniques resulted in partially positive results, which depend on either crude approximations [176; 15; 121], allowing for very large space [122], or providing a solution for the small distance regime [154]. On the other hand, there is little evidence that ED and LCS must admit any worse bounds than, respectively, ℓ_1 and Jaccard Similarity, and the current large gaps in the upper bounds beg resolution.

To address those "less understood" metrics, we first elaborate on the second class of ANN algorithms, namely the ones leveraging metric embedding, where the idea is to reduce ANN under some fixed metric to ANN under, say, ℓ_1/ℓ_2, or ℓ_∞, for which we can use the aforementioned "direct" algorithms. We start by stating the more traditional bi-Lipschitz embedding definition.

Definition 7 (bi-Lipschitz embedding into ℓ_2)**.** *A metric* $(\mathcal{X}, \mathsf{d})$ *embeds into* ℓ_2 *with distortion* c *there exists a map* $\varphi : \mathcal{X} \to \mathbb{R}^d$*, for some* $d \geq 1$*, such that for any* $p, q \in \mathcal{X}$*:*

$$\mathsf{d}(p, q) \leq \|\varphi(p) - \varphi(q)\|_2 \leq c \cdot \mathsf{d}(p, q).$$

Such embeddings have been studied for a number of metrics including edit distance, Ulam distance, Earth-Mover distance (Wasserstein-1 distance), ℓ_p-product spaces, general symmetric norms, Fréchet distance, and others. Alas, for many of these metrics, the best possible distortion remains a long-standing open question. A notorious example is edit (Levenshtein) distance over length-k strings: early results showed that the best distortion c_1 for embedding into ℓ_1 is $\Omega(\log k) \leq c_1 \leq 2^{O(\sqrt{\log k \log\log k})}$ [175; 134; 144] — improving over these bounds is a 20 years old open question [153].

More recently, researchers discovered that a weaker form of embeddings, namely *average* embeddings, yield ANN algorithms as well. In particular, [24] developed an algorithm where the aforementioned data-dependent hashing is constructed based on the theory of *non-linear spectral gaps* [158; 165]. The latter is (formally) related via duality to average embeddings, which we define below (restated and simplified from [166]):

Definition 8 (Average embedding into ℓ_1, [166])**.** *A metric* $(\mathcal{X}, \mathsf{d})$ *embeds into* ℓ_1 *with average distortion* c *if for every* $P \subset \mathcal{X}$*, there exists a map* $\varphi_P : \mathcal{X} \to \mathbb{R}^d$*, for some dimension* $d \geq 1$*, such that for any* $p, q \in \mathcal{X}$*:*

$$\frac{\|\varphi_P(p) - \varphi_P(q)\|_1}{\mathbb{E}_{x,y \in P} \|\varphi_P(x) - \varphi_P(y)\|_1} \leq c \cdot \frac{\mathsf{d}(p,q)}{\mathbb{E}_{x,y \in P} \mathsf{d}(x,y)}.$$

In other words, after a rescaling, the embedding φ_P is required to be c-Lipschitz

(i.e., expansion bounded by c), whereas the contraction is 1 on *average*. Most notably, motivated in part by a question from [26], Naor [166] showed that $c = \Theta(\log d)$ for a general d-dimensional norm; previously, [165] also showed that $c = \Theta(p)$ for ℓ_p. We remark that, average embeddings have been used in algorithm design since at least [182] otherwise.

Later, [145; 11] showed how one can directly use average embedding into ℓ_1/ℓ_2 to construct ANN algorithm with $O(c \log c)$ approximation, space/preprocessing time of $n^{1+\epsilon}$ and query time of n^{ϵ}, for any $\epsilon > 0$, up to factors that depend on the complexity of storing/computing the function φ_P. As a corollary, [145] obtained the first ANN algorithm for ℓ_p with $O(p)$ approximation and polynomial preprocessing.

Very recently, [38] extended the above notion to average sketching, which should be seen as a generalization of embedding into ℓ_1. The authors obtained a new trade-off between space and approximation for ℓ_p.

In our work, we introduce a new type of (average) metric embeddings — into similarity measures — and show how to instantiate it on *LSH Forest* (described above) to obtain efficient ANN data structures. While our focus will mainly be on ℓ_p and edit distance, we believe our techniques have broader applications. In particular, it turns out that embeddings into similarity measures are *provably more powerful*, meaning we can obtain smaller distortion than that possible via average embeddings into, say, ℓ_1. While we give two such examples, we expect that the general approach may yield new ANN algorithms for many other classic metrics as well. This work is set to appear in [FOCS'25].

We start by defining the notion of embedding into a similarity measure; in fact we consider a couple of embedding definitions below. For now, a similarity measure

over $\mathbb{R}^m_+$ is simply a symmetric function $\mathcal{S} : \mathbb{R}^m_+ \times \mathbb{R}^m_+ \to [0,1]$ such that $\mathcal{S}(x,x) = 1$ for all x [74; 80].

Definition 9 ((c,m)-UASE; see complete Definition 115)**.** *We say a metric $(\mathcal{X}, \mathsf{d})$ admits a (c,m)-Uniform Average Similarity Embedding (UASE) into $\mathcal{S}$ if for every $P \subseteq \mathcal{X}$, there exists a map $\varphi_P : \mathcal{X} \to \mathbb{R}^m_+$, such that for any $p, q \in \mathcal{X}$, we have:*

$$\frac{\log 1/\mathcal{S}(\varphi_P(p), \varphi_P(q))}{\log \mathbb{E}_{p',p'' \in P}[1/\mathcal{S}(\varphi_P(p'), \varphi_P(p''))]} \leq c \cdot \frac{\mathsf{d}(p,q)}{\mathbb{E}_{p',p'' \in P}[\mathsf{d}(p',p'')]}.$$

We primarily consider the target similarity measure $\mathcal{S}(x,y)$ being the *Sørensen-Dice coefficient*, denoted by $M(x,y) \triangleq \frac{2\sum_i \min\{x_i,y_i\}}{\sum_i x_i + y_i}$.[4] Other natural notions of similarities $\mathcal{S}$ include Jaccard coefficient and cosine similarity — however both yield equivalent UASE's up to a constant factor in distortion c (due to the relations between these similarities).

The above definition can be interpreted as an average embedding into a space with distance $\log 1/\mathcal{S}(x,y)$ for two vectors $x, y \in \mathbb{R}^m$ (replacing ℓ_1 in the Def. 8). In fact, for $\mathcal{S}$ being cosine similarity (or M, up to constants), the resulting distance is is called *Bhattacharyya distance* (which is formally not a distance), and is used in Statistics.

The definition above is not sufficient for our main results, and we need to extend it to handle a weighted mixture of (multiple) UASE, formally defined as follows:

Definition 10 ((c,m)-WASE; see complete Definition 116)**.** *We say a metric $(\mathcal{X}, \mathsf{d})$ admits a (c,m)-Weighted Average Similarity Embedding (WASE) into $\mathcal{S}$ if for every*

[4] It's easy to note that the definition is only interesting for specific similarities $\mathcal{S}$, as otherwise, for any $(\mathcal{X}, \mathsf{d})$, there always exists some $\mathcal{S}$ achieving $c = 1$.

$P \subseteq \mathcal{X}$ of size $n = |P|$, there exists a map $\varphi_P : \mathcal{X} \to \left(\mathbb{R}_+^{m/s}\right)^s$ for some $s \in \mathbb{N}$, such that for any $p, q \in \mathcal{X}$, we have:

$$\sum_{i\in[m/s]} \frac{\|\varphi_P(p)_i + \varphi_P(q)_i\|_1}{\|\varphi_P(p) + \varphi_P(q)\|_1} \cdot \min\left\{1 + 1/l_i, \frac{\log 1/\mathcal{S}(\varphi_P(p)_i, \varphi_P(q)_i)}{l_i}\right\} \le c \cdot \frac{\mathsf{d}(p,q)}{\mathbb{E}_{p',p''\in P}[\mathsf{d}(p',p'')]}, \tag{1.1}$$

where

$$l_i = \log 1/\,\mathbb{E}_{p',p''\sim\mathcal{D}_i}[\mathcal{S}(\varphi_P(p')_i, \varphi_P(p'')_i)], \qquad \mathcal{D}_i(x,y) \triangleq \frac{\|\varphi_P(x)_i\|_1 + \|\varphi_P(y)_i\|_1}{2n \cdot \|\varphi_P(P)_i\|_1}.$$

There are a few modifications from the UASE definition. The main extension is that the embedding ϕ_P is into m/s "outer coordinates", each one representing a UASE-like embedding $\phi_P(p)_i$; the contribution of the outer coordinates is weighted proportional to $\|\phi_P(p)_i + \phi_P(q)_i\|_1$. The second difference is that the "average contraction" is computed with respect to a non-uniform distribution $\mathcal{D}_i(p', p'')$ (in contrast, UASE uses uniform $p', p'' \in P$). Finally, the contribution is capped (used to handle points who are much more dissimilar than the average pair). All these modifications are introduced to enable us to construct such embeddings in the two concrete cases. A more in-depth discussion of the definitions is provided in Section 7.1.

One of our main contributions is to show that a WASE embedding yields efficient ANN algorithms (for similarity measure $\mathcal{S} = M$), stated somewhat informally next.

Theorem 11 (ANN from WASE; informal version of Theorem 148)**.** *Consider a metric $(\mathcal{X}, \mathsf{d})$ that admits a (c, m)-WASE into similarity $\mathcal{S} = M$. Suppose, for a dataset P of size n, we can compute the description of the function φ_P in time $nF_p(n)$, using space $nF_s(n)$, can compute $\varphi(q)$ for any given $q \in \mathcal{X}$ in time $F_q(n)$; and its "description*

complexity" is κ (formally defined in Definition 115). Then, for any $\epsilon > 0$, there exists an algorithm for ANN for P with the following performance:

- *Approximation $O(c/\epsilon)$;*
- *Query time of $O(n^{\epsilon} \cdot (F_q(n) + T_{\mathsf{d}})\kappa^{O(1)})$ where T_{d} is time to compute the distance d;*
- *Space of $O(n^{1+\epsilon} \cdot F_s(n)\kappa^{O(1)})$, in addition to storing the dataset;*
- *Preprocessing time of $O(n^{1+\epsilon} \cdot F_p(O(n))\kappa^{O(1)})$.*

The algorithm is based one *LSH Forest* where the embedding is being produced in each node of the forest, and then some carefully crafted hash function is applied on the embedding for each dataset point, as well as the query point. The full algorithm is described in Chapter 7.

We now describe our concrete results on WASE embeddings for two classic metrics. Combined with Theorem 11, we obtain improved ANN algorithms in the following well-studied settings.

Edit distance over length-k strings. The edit distance between two length-k strings x, y is the number of insertions/deletions/substitutions to transform x into y, and is a classic way to compare sequences, such as genomic data [148]. NNS under edit distance is a core task; e.g., see an application for DNA storage [183]. Despite receiving much attention over decades, NNS under edit distance has seen little progress.

We design a $\mathrm{poly}(\log k)$ distortion WASE embedding (Theorem 131), which together with Theorem 11, implies the following ANN data structure.

Theorem 12 (ANN for edit distance. See Chapter 7)**.** *For any $k \geq 1$ and $\epsilon \in (0,1)$, we can design an ANN data structure under the edit distance over Σ^k achieving $O(\frac{1}{\epsilon}(\log k)^{O(1)})$ approximation, $O(n^{\epsilon} \cdot k^{O(1)})$ query time, and $O(n^{1+\epsilon}k^{O(1)})$ space and preprocessing time.*

Our ANN algorithm improves over the $c = 2^{O(\sqrt{\log k \log\log k})}$-approximation ANN obtained from an embedding into ℓ_1 [175]. Improving over this embedding has been a long-standing open problem [153]. The lower bound on embedding stand at $\tilde{\Omega}(\log k)$ [134; 144; 19]. Even for average embedding, nothing nontrivial is known; see open question in [24]. Our result can be seen as progress on the embeddability of edit distance for the relaxed notion of average embeddings into similarity measures.

ℓ_p norm over $\mathbb{R}^d$. We design an $O(\log p)$ distortion WASE embedding for ℓ_p (Theorem 118), which, together with Theorem 11, yields the following ANN data structure.

Theorem 13 (ANN for ℓ_p. See Chapter 7)**.** *For any $d \geq 1$ and $\epsilon > 0$, $p > 1$, we can design an ANN data structure under $(\mathbb{R}^d, \ell_p)$ norm achieving $O(\frac{\log p}{\epsilon})$ approximation, $O(n^{\epsilon}d^{O(1)})$ query time, and $O(n^{1+\epsilon}d^{O(1)})$ space and preprocessing time.*

This improves over $\Theta(p)$-approximation ANN algorithm [145], based on an average embedding into ℓ_1; the $\Theta(p)$ bound is tight for such embeddings [165]. Previous results also include $O(2^p)$ approximation [167; 40], $O(\log\log d)$ approximation [8], and $O(\log p \cdot \log^{2/p} d)$ [27] (the latter result has approximation $< p$ for some values of p, but also has exponential preprocessing time). In a recent and independent work, [38] develop the notion of *average sketching* that yields a different trade-off: c-approximation with $n^{O(p/c)}$ space for any c at least a constant. Finally, a recursive metric embedding approach was introduced very recently [143], yielding poly(p) approximation.

We note that our bound of $c = \Theta(\log p)$ for ℓ_p is tight in a number of ANN data structure models. This follows from the lower bounds for ℓ_∞ [12; 129; 179; 57], noting that the same hard instance has an approximation gap of $\Omega(\log p)$ in ℓ_p norm.

Finally we remark that in the particular case of ℓ_∞, the approximation becomes $O(\log \log d)$ (by setting $p = \log d$), matching the approximation of the classic result from [120]. Thus this is a new (tight) algorithm for ℓ_∞, which is quite different from the original algorithm of [120]. Furthermore, our WASE embedding for ℓ_∞ is used to develop the edit distance WASE embedding.

1.3 Organization

This dissertation is split to two parts: Strings (Part I) and Nearest Neighbor Search (Part II).

In Part I, we showcase the aforementioned algorithm and analysis for the ED (Chapter 3), LIS (Chapter 4) and LCS (Chapter 5), and in Part II, we showcase our ANN constructions for LSH Forest for Bit-Sampling (Chapter 6) as well as the similarity embeddings which eventually obtain the results for ℓ_p and ED data-structures (Chapter 7).

We also present related work in Chapter 2.

Chapter 2

Related Work

Here we expand on prior work related to this thesis.

2.1 Edit distance and longest common subsequence

Focusing on constant-factor algorithms, a quantum algorithm for approximating edit distance was introduced in [50]. Some of the basic elements of the algorithmic approach are related to [70] (and the algorithm in this paper). Another recent related paper is [105], who obtain $3+\epsilon$ approximation in $\tilde{O}(n^{1.6})$ time; independently, the first author obtained a slightly worst time for the same approximation [10]. Similarly, independently, [71] and [10] extended the constant-factor edit distance algorithm from [70] to solve the text searching problem.

Progress on edit distance algorithms also inspired the first non-trivial algorithms for approximating the longest common subsequence (LCS) [109; 185; 186; 62], leading to a linear time, $n^{o(1)}$-approximation algorithm [29; 173]. Also of note is [186] which shows that a $O(1)$-factor approximation to edit distance implies a $2-\Omega(1)$ factor approximation to LCS over a binary alphabet in essentially the same time (in fact,

the additive approximate algorithms from [140; 55] are already enough for this application), which was later extended to the more general small alphabet setting [6; 115].

2.2 Streaming algorithms for string problems

Computing the length of LIS has been studied in the streaming model, where settling its complexity is a major open problem [177]. In this setting, the main question is to determine the minimum space required to estimate the length of LIS by an algorithm reading the sequence left-to-right. [107; 94] gave deterministic one-pass algorithms to $(1+\epsilon)$-approximate the LIS using $O(\sqrt{n})$ space, and matching lower bounds against deterministic algorithms were given by [94; 99]. These lower bounds are derived using deterministic communication complexity lower bounds and provably fail to extend to randomized algorithms [68]. However, no randomized algorithm requiring $o(\sqrt{n})$ space is known for this problem either. We note that our algorithm can be used in the streaming setting, yielding streaming complexity $O(n^{1/2+\epsilon})$ for approximation $n^{o(1)}$.

For ED, the main focus has been on obtaining polynomial time sketch and recovery algorithms that achieve the smallest possible sketch length for a parametrized threshold k. Jin, Nelson and Wu [45] proved that sketches must have length $\Omega(k)$. Progress has been made on the upper bound [128; 45; 136] culminating in near-linear sketch size [141] as well.

A streaming algorithm for LCS was also considered [34], also in the case of multiple sequences [98].

2.3 Sublinear-time edit distance algorithms

Sublinear-time algorithms for edit distance have originally developed in the early 2000s [41; 39; 30], and have recently drawn renewed attention [104; 137; 54; 103; 64]. Most recently, a truly sublinear-time algorithm with a *subpolynomial* approximation gap was found [60; 59], solving the $(k, k^{1+o(1)})$-gap edit distance problem in time $O(n/k + k^{2+o(1)})$. Another related line of work has been on computing edit distance for the semi-random models of input [20; 146; 51].

2.4 Dynamic and parallel algorithms

The LIS problem has also been studied recently in other settings, such as the MPC and the fully dynamic settings. [118] gave a $(1+\epsilon)$-approximation, $O(1/\epsilon^2)$-round MPC algorithm for LIS whenever the space per machine is $n^{3/4+\Omega(1)}$. In the dynamic setting, a sequence of works [160; 101] culminating in [138], gave the first exact dynamic LIS algorithm with sublinear update time, and also gave a deterministic algorithm with update time $n^{o(1)}$ and approximation factor $1 - o(1)$. [100] showed conditional lower bounds on update time for certain variants of the dynamic LIS problem.

For fully dynamic LCS, early work [36] has proposed algorithms with subquadratic update and query times under restricted models. More recent advances include [102] . Similarly, dynamic edit distance has been studied under both cell-update models and symbol-update models. For example, [73] achieved polylogarithmic updates for small alphabets in the cell-probe model. For edit distance, parallel (MPC) algorithms were developed in [50; 111].

2.5 Other string problems

There has been much more success on the problem of estimating the distance to monotonicity, which "complement" LIS (i.e., $d_m := n - \mathsf{LIS}$), in both the sublinear-time and the streaming settings. In the random-access setting, the problem was first studied in [5], and later in [189], who gave an algorithm that achieves $(1+\epsilon)$-approximation in time $\mathrm{poly}(1/d_m, \log n)$ for any constant $\epsilon > 0$. These algorithms have been also used, indirectly, for faster algorithms for estimating Ulam distance [15; 170] and smoothed edit distance [20].

In the streaming setting, several results were obtained which achieve $O(1)$-approximation of d_m using $\mathrm{polylog}(n)$ space [107; 94]. This culminated in a randomized $(1+\epsilon)$-approximation algorithm using $\mathrm{polylog}(n)$ space by Saks and Seshadhri [188], and a deterministic $(1+\epsilon)$-approximation algorithm using $\mathrm{polylog}(n)$ space by Naumovitz and Saks [169]. [169] also showed space lower bounds against $(1+\epsilon)$-approximation streaming algorithms of $\Omega(\log^2 n/\epsilon)$ (deterministic) and $\tilde{\Omega}(\log^2 n/\epsilon)$ (randomized).

Another classical problem is the pattern matching remains a classic problem with optimal linear-time algorithms [135; 53] and nearly optimal streaming solutions using fingerprinting and sketching [181]. Approximate matching under Hamming or edit distance constraints has seen advances in both cell-probe [73] and streaming models [82].

2.6 Data-dependent hashing

As previously mentioned, data-dependent hashing is an ubiquitous idea in practice, albeit with few, if any guarantees on correctness or performance. To list just a few

examples, these include PCA-trees [193; 156; 201] and its variants (called randomized kd-tree) [192; 163], spectral hashing [205], semantic hashing [190], and WTA hashing [206], and many more. We point an interested reader to the surveys of [204; 203].

There have also been efforts to obtain algorithms that are both data-dependent (in a practical way) and have theoretical guarantees. Most such examples include the algorithms that assume some additional structure in the dataset: such as some notion of *low intrinsic dimension* [130; 77; 142; 48; 125; 81; 87], or low dimensional data-set with high-dimensional noise [4]. Most relevant to us is the work of [89], which, while mostly focusing on the low intrinsic dimensional datasets, give a generic bound for the *worst-case* datasets as well. The bound of [89] however depends on the dataset in a non-obvious way, and it is not clear whether it obtains bounds that are better than (data-independent) LSH.

2.7 ANN for matching and string problems

Embeddings for EMD have been studied extensively for low- and high-dimensional spaces $\mathcal{X}$ [126; 134; 168; 14] (and many more results exist for related problems of fast evaluation, sketching, ANN; see, e.g., the references in the recent [127]). Overall, for high-dimensional spaces, $\mathsf{EMD}_{\mathcal{X}}$ when $\mathcal{X}$ is d-dimensional ℓ_1 has an ℓ_1 distortion of $\approx O(\log k \log d)$; recently, [127] also showed that, for ANN, one can improve the approximation to $\tilde{O}(\log k)$ via a data-dependent hashing algorithm.

For edit distance, as mentioned earlier, the best known ANN construction prior to this work was using the influential $c = 2^{O(\sqrt{\log k \log\log k})}$-approximation ℓ_1 embedding [175]. Improving over this has been a long-standing open problem [153], with

the lower bound still standing at $\tilde{\Omega}(\log k)$ [134; 144; 19].

Other prior ANN algorithms include [123] who achieves approximation $c = 3^\ell$ using space roughly $n^{k^{1/\ell}}$, and the line of work from [83; 72; 155] handling the case when the distance to the nearest neighbor is small (sub-logarithmic in n). Better algorithms are also known for the Ulam metric, which is the restriction of edit distance to non-repetitive strings [76; 15] (via embedding into ℓ_1 or other normed spaces).

ANN was also studied for LCS [92] and Fréchet [121]. More recent work include [63; 159].

Part I

String Problems

Chapter 3

Edit distance

Edit distance is a classic distance measure between sequences that takes into account the (mis)alignment of strings. Formally, edit distance between two strings of length n over some alphabet Σ is the number of insertions/deletions/substitutions of characters to transform one string into the other.

Being of key importance in several fields, such as computational biology and signal processing, computational problems involving the edit distance were studied extensively.

In this chapter we showcase an algorithm for approximating the edit distance between two strings of length n in time $n^{1+\epsilon}$ up to a constant factor, for any $\epsilon > 0$, proving Theorem 3, restated below:

Theorem 14 (Theorem 3, restated)**.** *There is a randomized algorithm that, for any $\epsilon > 0$, $n \geq 1$, and alphabet Σ, and any two strings $x, y \in \Sigma^n$, approximates the edit distance between x and y in expected $O(n^{1+\epsilon})$ time up to $f(1/\epsilon)$-factor approximation, for some universal non-decreasing function $f(\cdot)$.*

This result completes a research direction set forth in the breakthrough paper [70],

which showed the first constant-factor approximation algorithm with a (strongly) sub-quadratic running time. The recent results [140; 55] have shown near-linear time algorithms that obtain an additive approximation, near-linear in n (equivalently, constant-factor approximation when the edit distance value is close to n). In contrast, our algorithm obtains a constant-factor approximation in near-linear time for any input strings.

In contrast to prior algorithms, which are mostly recursing over smaller substrings, our algorithm gradually smoothes out the local contribution to the edit distance over progressively larger substrings. To accomplish this, we iteratively construct a distance oracle data structure for the metric of edit distance on all substrings of input strings, of length $n^{i\epsilon}$ for $i = 0, 1, \ldots, 1/\epsilon$. The distance oracle approximates the edit distance over these substrings in a certain average sense, just enough to estimate the overall edit distance.

While we do not derive the function $f(1/\epsilon)$ explicitly, we note that it is doubly exponential in $1/\epsilon$. We present a technical overview of our approach in Section 3.2, after setting up our notations in Section 3.1. The proof of the main theorem will follow in subsequent sections, in particular the top-level algorithm and its main guarantees are in Section 3.3.

3.1 Preliminaries: setup and notations

Fix a pair of strings $(x, y) \in \Sigma^n \times \Sigma^n$ for which we care to estimate the edit distance. We define $\mathtt{ed}_n(x, y)$ as half the number of insertions/deletions to transform one string into the other. Note that the standard edit distance (allowing substitutions) can be

reduced to this case (see, e.g., [196]). When length n is clear from the context, we omit the subscript.

S_w is the set of integer powers of 2 up to w: namely, $S_w = \{1, 2, 4, 8 \dots w\} \cup \{1/2, 1/4 \dots 1/2n\}$.

$[n]$ denotes the set $\{1, 2, 3, \dots n\}$ throughout the paper. Also, the notation $[a, b)$ means all integers $\{a, ...b-1\}$, whereas $[a, b] = \{a, ...b\}$.

We assume that $n, 1/\epsilon$ are powers of 2. Furthermore, we assume n is a sufficiently large integer power of $1/\epsilon$, and that n^ϵ is integer as well. These are obtainable via standard rounding of $n, 1/\epsilon$.

When describing intuitive parts, we sometimes use $O^*(f(n))$ to denote $f(n) \cdot n^{O(\epsilon)}$ (where ϵ is the small constant from the algorithm).

Finally, we use the standard Chernoff bound (below) as well as notion of *with high probability* (whp), meaning with probability at least $1 - n^{-C}$ for large enough constant $C > 1$.

Theorem 15 (Chernoff bound)**.** *Consider a sum of independent r.v. $X_1, \dots X_m \in [0, 1]$ for some $m > 1$. Then, if $\mathbb{E}[\sum X_i] = \Omega(\log n)$, then $\sum X_i \in [0.9, 1.1] \cdot \mathbb{E}[\sum X_i]$ w.h.p. (where implicit constant in Ω depends on C). Note that this implies that (without any condition), $\sum X_i \geq 0.9\, \mathbb{E}[\sum X_i] - O(\log n)$ w.h.p.*

3.1.1 Intervals

An interval is a substring $x[i : j] \triangleq x_i x_{i+1} \dots x_{j-1}$, for integer i, j where $i \leq j$ (i.e., starting at i and ending at $j - 1$, of length $j - i$). When part of the string is outside bounds of x, we assume those characters are a default character, say, $\$ \notin \Sigma$.

For $i \in [n]$, let $X_{i,w}$ ($Y_{i,w}$) denote the interval of x (y) of length w starting at position i. Let $\mathcal{W}_w, \mathcal{H}_w$ the set of all such $X_{i,w}$ and $Y_{j,w}$ strings respectively. We use $\mathcal{I}_w = \mathcal{W}_w \cup \mathcal{H}_w$ to denote all x and y axis intervals. When clear from context, we drop subscript w.

By convention, if $i \notin [1, n-w]$, we pad X_i/Y_i with the default character \$. Also $X_{\perp,w}$ ($Y_{\perp,w}$) and is a string of unique characters. In particular, for all distance functions $\tau_w(\cdot,\cdot)$ on two length-w strings in this paper, we define $\tau_w(X_{\perp,w}, \cdot) = w$; e.g., $\mathtt{ed}_w(X_{i,w}, \cdot) = w$.

Usually, by $I \in \mathcal{I}_w$ we refer not only to the corresponding substring but also to the "meta-information", in particular the string it came from, start position, and length (e.g., for $I = X_{i,w}$, the meta-information is x, i, w). This difference will be clear from context or stated explicitly.

In particular, the notation $I + j$, for an interval I and integer j, represents the interval j positions to the right; e.g., if $I = X_{i,w}$, then $I + j = X_{i+j,w}$.

Alignments. An alignment between x and y is a function $\pi : [n] \to [n] \cup \{\perp\}$, which is injective and strictly monotone on $\pi^{-1}([n])$. The set of all such alignments is called Π. Note that $\mathtt{ed}(x,y) = \min_{\pi\in\Pi} \sum_{i\in[n]} \mathtt{ed}_1(x_i, y_{\pi(i)})$ (recall that, by convention, $\mathtt{ed}_1(c, y_\perp) = 1$ for all $c \in \Sigma$). We define $\pi(i) \triangleq i$ when $i < 1$ and $i > n$ for convenience.

It is convenient for us to think of π as function from $\mathcal{I}_w \to \mathcal{I}_w$ (for fixed w), via the following extension. First, abusing notation, we think of $\pi : [n] \to [n] \cup \{\perp\}$ as $\pi : \mathcal{X}_w \to \mathcal{Y}_w \cup \{\perp\}$ where each interval is identified with its start position. Its

extension $\widehat{\pi} : \mathcal{I}_w \to \mathcal{I}_w \cup \{\perp\}$ is:

$$\widehat{\pi}[I] = \begin{cases} \pi[I] & I \in \mathcal{X} \\ \pi^{-1}[I] & I \in \mathcal{Y} \end{cases},$$

where $\widehat{\pi}[X_{i,w}]$ means $Y_{\pi(i),w}$, and $\widehat{\pi}[Y_{j,w}]$ means $X_{\pi^{-1}(j),w}$, with $\pi^{-1}(j) = \perp$ if there's no i with $\pi(i) = j$. Throughout this paper, we overload notation to use π for the extension $\widehat{\pi}$ as well. We also define $\overleftarrow{\pi}(i)$ as the minimum $\pi(j)$, $j \geq i$, which is defined ($\neq \perp$).

3.1.2 Interval distances

Our algorithms will use distances/metrics over intervals in $\mathcal{I}_w$. One important instance is the *alignment distance*, denoted $\mathtt{ad}_w(\cdot,\cdot)$. At a high level, $\mathtt{ad}_w(\cdot,\cdot)$ is a distance metric that approximates edit distance on length-w intervals (though a variant of it is not formally a metric). We discuss $\mathtt{ad}(\cdot,\cdot)$ metric in Section 3.3 as well as 3.8.

Definition 16 (Neighborhood)**.** *Fix $c \geq 0$ and $I \in \mathcal{I}_w$. The* c-neighborhood *of I is the set $\mathcal{N}_c(I) = \{J \in \mathcal{I}_w \mid \mathtt{ad}(I,J) \leq c\}$, i.e. all x and y intervals which are c-close to I in terms of their alignment distance.*

Definition 17 (Ball of intervals)**.** *A* ball of intervals *is a set of consecutive intervals in either $\mathcal{X}_w$ or $\mathcal{Y}_w$ (i.e., it's a ball in the metric where distance between X_i and X_j is $|i-j|$). The* smallest enclosing ball *of a set $\mathcal{S}$ (subset of $\mathcal{X}_w$ or $\mathcal{Y}_w$) is the minimal ball $\mathcal{B} \supseteq \mathcal{S}$.*

Average approximation for an optimal alignment. A common theme in our algorithm is constructing metrics on $\mathcal{I}_w$ approximating $\mathtt{ed}_w$ in a certain "average sense".

In particular, this differs from the standard notion of approximation in that the upper bound holds only on average, and for an optimal alignment $\pi \in \Pi$. Formally, we define:

Definition 18 (Align-approximation)**.** *Fix space* $(\mathcal{I}_w, \mathtt{d})$ *over* $\mathcal{I}_w$ *(which is often a metric space, but need not be). We say* $\mathtt{d}$ T-align-approximates $\mathtt{ed}$ *if the following holds:*

1. *For all* $I, J \in \mathcal{I}_w$*:* $\mathtt{d}(I, J) \geq \mathtt{ed}_w(I, J)$.
2. $\min_{\pi \in \Pi} \sum_{i \in [n]} \frac{1}{w} \mathtt{d}(X_i, Y_{\pi(i)}) \leq T \cdot \mathtt{ed}(x, y)$.

3.1.3 Operations on sets and the $*$ notation

By convention, applying numerical functions to a set refers to the sum over all set items; e.g., $f(S) = \sum_{i \in S} f(i)$. When applying set operators on other sets, we use the union; e.g., $\pi(\mathcal{S}) = \cup_{I \in \mathcal{S}} \pi(I)$ and $\mathcal{N}_c(\mathcal{S}) = \cup_{I \in \mathcal{S}} \mathcal{N}_c(I)$. Abusing notation, we use $f(S)$ even when S is not fully contained in the domain of f — in which case, we simply ignore elements outside the domain. Any exception to the above will be clearly specified.

We also use the notation $*$ as argument of a function, by which we mean a vector of all possible entries. E.g., $f(*)$ is a vector of $f(i)$ for i ranging over the domain of f (usually clear from the context). Similarly, $f(*_{\mathcal{R}})$ means a vector of $f(i)$ for i satisfying property $\mathcal{R}$. Overloading notation, sometimes $f(*_{\mathcal{R}})$ will also mean a vector of $f(i)$ for all i in the domain, with coordinates $i \notin \mathcal{R}$ being zeroed-out.

3.2 Technical overview

3.2.1 Prior work and main obstacles

As our natural starting point is the breakthrough $O(n^{12/7})$-time algorithm of [70] (and related [50]), we first describe their core ideas as well as the challenges to obtaining a near-linear time algorithm. In particular, we highlight two of their enabling ideas. At a basic level, their algorithm computes edit distance $\texttt{ed}_n(x, y)$ by computing $\texttt{ed}_w$ between various length-w *intervals* (substrings) of x, y recursively, and then uses edit-distance-like dynamic programming on intervals to put them back together. The main algorithmic thrust is to reduce the number of recursive $\texttt{ed}_w$ computations: e.g., if the intervals are of length w, and we only consider non-overlapping intervals, there are still $n/w \times n/w$ calls to $\texttt{ed}_w$, each taking at best $\Omega(w)$ time. Hence, [70] employ two ideas to do this efficiently: 1) use the triangle inequality to deduce distances between pairs of intervals for which we do not directly estimate $\texttt{ed}_w$, 2) two nearby x-intervals (e.g., consecutive) are likely to be matched into two nearby y-intervals (also consecutive) under the optimal edit distance alignment π. (The earlier quantum result [50] employed the first idea already, but relied on a quantum component instead of the second idea.) Indeed, these ideas are enough to reduce the number of recursive calls from $(n/w)^2$ to $\approx (n/w)^{1.5}$.

One big challenge in the above is that, in general, one has to consider all, overlapping intervals from x, y, of which there are n — since, in an optimal $\texttt{ed}_n$ alignment, an x-interval might have to match to a y-interval whose start position is far from an integer multiple of w. An alternative perspective is that if one considers only a restricted set of interval start positions, say every $s \leq w$ positions

in y, then one obtains an extra *additive* error of about $s \cdot n/w$ from the "rounding" of start positions in y. That's the reason that a bound of $(n/w)^{1.5}$ recursive calls did not transform into $n^{1.5}$ runtime in [70]: to compute edit distance when $\mathtt{ed} < n^{1-\Omega(1)}$, they employ a standard (exact) $O(n + \mathtt{ed}^2(x, y))$ algorithm [198; 164].

Recent improvements by [140; 55] showed how to reduce the number of recursive calls to $\approx n/w$, but some fundamental obstacles remained. The linear number of recursive calls was leveraged to obtain near-linear time but with an additive approximation only: when $\mathtt{ed}(x, y) \geq n^{1-\delta}$, the overall runtime is $n^{1+g(\delta)}$ for some increasing function g.

In particular, in addition to the aforementioned challenge, a new challenge arose: to be able to reduce to near-linear number of recursive $\mathtt{ed}_w$ calls, the algorithms from [140; 55] might miss a large fraction of "correct" matches. In particular this fraction is $\approx n^{-\delta}$, which results in an additive error of $\approx n^{1-\delta}$. To put this into perspective, for $w = \sqrt{n}$, if we allow an additive error $n^{1-\delta}$, then it suffices to analyze $b = n^{0.5+O(\delta)}$ intervals (which barely overlap) and misclassify $b \cdot n^{-\delta}$ of them. We use the following example to showcase the challenges:

A running example illustrating the challenges

Consider an instance where $\Delta = n^{-0.01}$ fraction of intervals in $\mathcal{X}_w$ are "sparse" — have a single cheap match in $\mathcal{Y}_w$ — and the rest of the intervals are dense *(they are close to many other intervals). Assume further that such sparse intervals are spread around in multiple "sparse sections" (sequences of consecutive sparse intervals).*

Note that if we can afford large additive errors, we can simply *ignore* all these sparse intervals (certifying them at max cost w) and output the distance based on the

dense intervals only, with at most $n\Delta = n^{0.99}$ additive approximation. To avoid this, one must first identify some sparse intervals (since the dense intervals do not provide sufficient information about the sparse sections). Even if we manage to find some of the sparse intervals efficiently, we still need to apply knowledge of the location of such intervals to deduce information on other intervals which might be in completely different areas in the string. We will return to this example later.

Below we describe the high-level approach to our algorithm, including how we overcome these obstacles. We note that, while our algorithm is based on the two key ideas from [70], the high-level algorithm departs from the general approach undertaken in [70; 140; 55]. That said, some algorithmic steps are similar to those developed in [140; 55]. We do not rely on previous results (for any distance regime) such as [198].

3.2.2 Our high-level approach

While there are many ideas going in overcoming the above challenges, one common theme is *averaging over the local proximity of intervals*. In particular, the algorithm proceeds by, and analyzes over, "average characteristics" of various intervals of x, y, in a "smooth" way. For example decisions for a fixed interval $I \in \mathcal{I}_w$, such as whether something is close, or something is matched, are done by considering the statistics collected on nearby intervals (to the left/right of I in the corresponding string). While we expand on our technical ideas below, this is the guiding principle to keep in mind.

Addressing the first challenge, we consider intervals (of fixed length w) at all n starting positions, i.e., the entire set $\mathcal{I}_w$. Note that recursion becomes prohibitive: we can't perform even n edit distance evaluations each taking $\Omega(w)$ time (w is set to be $\approx n^{1-\epsilon}$). Instead, our top-level algorithm iterates bottom–up over all interval lengths

$w = \gamma, \gamma^2, \ldots n$, where $\gamma = n^\epsilon$, and for each w computes a good-enough approximation to *the entire metric* $(\mathcal{I}_w, \mathtt{ed}_w)$. Recall that $\mathcal{I}_w = \mathcal{W}_w \cup \mathcal{H}_w$ consists of all w-length intervals (substrings); i.e., $|\mathcal{I}_w| = 2n$. The metric, termed $\mathfrak{D}_w(\cdot, \cdot)$, will be accessible via a distance oracle (fast data structure), with n^ϵ query time, and will approximate the distance between most of the pairs (an x-interval, y-interval) that participate in an optimal alignment in an average sense. Specifically, for π ranging over all alignments, we will have that $\min_\pi \frac{1}{w} \sum_i \mathfrak{D}_w(X_{i,w}, Y_{\pi[i],w}) = \Theta_\epsilon(\mathtt{ed}(x, y))$. Formally, we say $\mathfrak{D}_w$ $O_\epsilon(1)$-*align-approximates* $\mathtt{ed}$ (see Def. 18).

In each iteration, we build $\mathfrak{D}_w$ using $\mathfrak{D}_{w'}$, where $w' = w/\gamma$. Conceptually we do so in two phases. First, we build another metric on w-length strings, $(\mathcal{I}_w, \mathtt{ad}_w)$, accessible via a fast distance oracle, that uses $\gamma^{O(1)} = n^{O(\epsilon)}$ time and $\mathfrak{D}_{w'}$ oracle calls. Crucially, $\mathtt{ad}_w$ will similarly align-approximate $\mathtt{ed}$. Second, equipped with a fast oracle for $\mathtt{ad}_w$ (itself using $\mathfrak{D}_{w'}$), we build an "efficient representation" for the entire metric $(\mathcal{I}_w, \mathtt{ad}_w)$, while using only $n^{1+O(\epsilon)}$ calls to $\mathtt{ad}_w$ oracle. Naturally, this "efficient representation" will not be able to capture the entire $\mathtt{ad}_w$ metric (that would require $\gg n$ query complexity), but it will capture just enough to preserve the edit distance between x and y—again, formally, align-approximate $\mathtt{ed}$. Then we build an efficient distance oracle for this efficient representation, which will yield the desired metric $\mathfrak{D}_w$. Note that the final approximation to $\mathtt{ed}(x, y)$ is computed by (essentially) querying $\mathfrak{D}_n(X_{1,n}, Y_{1,n})$.

In particular, the "efficient representation" of $\mathtt{ad}_w$ is a weighted graph G_w with vertex set $\mathcal{I}_w$ and $n^{1+\epsilon}$ edges, such that the shortest path between $I, J \in \mathcal{I}_w$ approximates $\mathtt{ad}_w(I, J)$, again, in an average sense for an optimal alignment. In particular, the shortest path distance is non-contracting, and non-expanding on average for inter-

val pairs that "matter", i.e., which are part of the optimal alignment π corresponding to $\mathtt{ed}(x, y)$. An edge (I, J) of the graph G_w will always correspond to an explicit call to $\mathtt{ad}_w(I, J)$; and the main question in constructing G_w is deciding *which* $n^{1+\epsilon}$ pairs to compute $\mathtt{ad}_w$ for.

Once we have the graph G_w, we build a fast distance oracle data structure on it to obtain the metric $\mathfrak{D}_w$. In particular, our fast distance oracle is merely an embedding of the shortest path metric on G_w into ℓ_∞^d, where $d \approx |\mathcal{I}|^\epsilon$, incurring an approximation of $O(1/\epsilon)$, via [151]. We note that we cannot use some other common distance oracle, such as, e.g., [195; 78], because they do not guarantee that the resulting output is actually a metric, and in particular, that it satisfies the triangle inequality, which is crucial for us (as mentioned above). We remark that this particular step is somewhat reminiscent of the approach from [30], who similarly build an efficient representation for the metric $(\mathcal{I}_w, \mathtt{ed}_w)$ using metric embeddings. However, the similarity ends here: first [30] used Bourgain's embedding into ℓ_1, which incurs $\Theta(\log n)$ distortion, and second, more importantly, the construction of G_w was altogether different (incurring a much higher approximation).

Computing the graph G_w itself is the most algorithmically novel part of our approach, and is termed *Interval Matching Algorithm*, as it corresponds to matching intervals that are close in $\mathtt{ad}_w$ distance. This algorithmic part should be thought of as the analogue of the algorithm deciding for which pairs of intervals to (recursively) estimate the edit distance in [70].

We sketch the Interval Matching Algorithm next in this technical overview. We also sketch how to compute the $\mathtt{ad}_w$ distance in $n^{O(\epsilon)}$ time, which presents its own new challenges, especially to guarantee its metric properties.

3.2.3 Interval matching algorithm

The main task here is to efficiently compute a graph G_w that approximates $\mathtt{ad}_w$, in an average sense over an optimal alignment π. Specifically, to generate G_w, we iterate over all costs $c \in S_w$ (powers of 2), and for each such cost, we then generate a (sub-)graph $G_{w,c}$, with edges of weight $\Theta_\epsilon(c)$. The following is the main guarantee: for any fixed alignment π, for any pair $(I, \pi[I]) \in \mathcal{I}_w^2$ at a distance $\mathtt{ad}(I, \pi[I]) \leq c$, we generate a 1- or 2-hop path for it in $G_{w,c}$—except for $O_\epsilon(k_c)$ such pairs $(I, \pi[I])$ where k_c is the number of pairs with $\mathtt{ad}(I, \pi[I]) > c$ (i.e., the "error" increases by at most a constant factor). The union[1] of such graphs $G_{w,c}$ yields the final graph G_w. Below we focus on a single scale graph $G_{w,c}$, which is supposed to capture nearly all pairs $(I, \pi[I])$ where $\mathtt{ad}_w(I, \pi[I]) \leq c$. We refer to such a pair as a *π-matchable pair* $(I, \pi[I])$.

At its core, our algorithm can be thought of as a *partitioning* algorithm, where we partition $\mathcal{I}_w$ into sets of intervals, such that for nearly all π-matchable pairs $(I, \pi[I])$, both intervals belong to the same set. We start with a single set of intervals and we iteratively partition the set into progressively more refined partitions (consisting of smaller parts), with the goal of keeping π-matchable pairs $I, \pi[I]$ together. (This algorithm will use a significant amount of notation, and, while this high-level overview will mention only a fraction of them, the reader may refer to the Table 3 in Sec. 3.5 for some important definitions and formulas.)

In particular, the matching algorithm proceeds in $\approx 1/\epsilon$ steps. In each step t, for $\lambda = n^\epsilon$, we generate λ^t parts, each of size $\lesssim n/\lambda^t$. To construct a part, we sample

[1] When there are multiple edges (I, J) from different c's, we naturally take the minimum-weight edge—i.e., the smallest distance certificate.

a random interval A, termed *anchor*, and estimate $\mathtt{ad}_w(A, I)$ for all other intervals I in its part, generating a *cluster* of intervals at distance $O_\epsilon(c)$ from the anchor. The main desideratum is that the two intervals from a π-matchable pair $I, \pi[I]$ are either both close to A or both far from A, and hence always remain together (this is related to the triangle inequality idea from [70]). However, this cannot be guaranteed, and ensuring this desideratum is a major challenge for us, which we will address later. For now, in order to build intuition, we first develop our ideas under the following the assumption, that the desideratum holds:

Perfect Neighborhood Assumption (PNA):

any two intervals are at distance either $\leq c$ or $\omega(c)$ so that $\mathcal{N}_{O_\epsilon(c)}(I) = \mathcal{N}_c(I)$.

Every anchor will generate precisely one part, of target size. Notice that if the cluster is sufficiently large (i.e., $\gtrsim n/\lambda^t$), we are basically done, and in fact, this is where our algorithm "converges" (as will be described later). Otherwise, we use the cluster to construct one part (set) by taking the clustered intervals together with their *local extensions*: intervals around the clustered intervals (to the left/right of the clustered ones). The parameters are set up such that the resulting part has size $\lesssim n/\lambda^t$. Note that the iterative nature of the process helps ensure the runtime: As the size of parts decreases with step t, we can afford to use more anchors. In particular, at step t, we start with λ^{t-1} parts, each of size about n/λ^{t-1}, and hence, for each of λ^t anchors, we need to estimate $\mathtt{ad}$ distance to n/λ^{t-1} intervals (in its part), for an overall of $n\lambda$ distance computations.

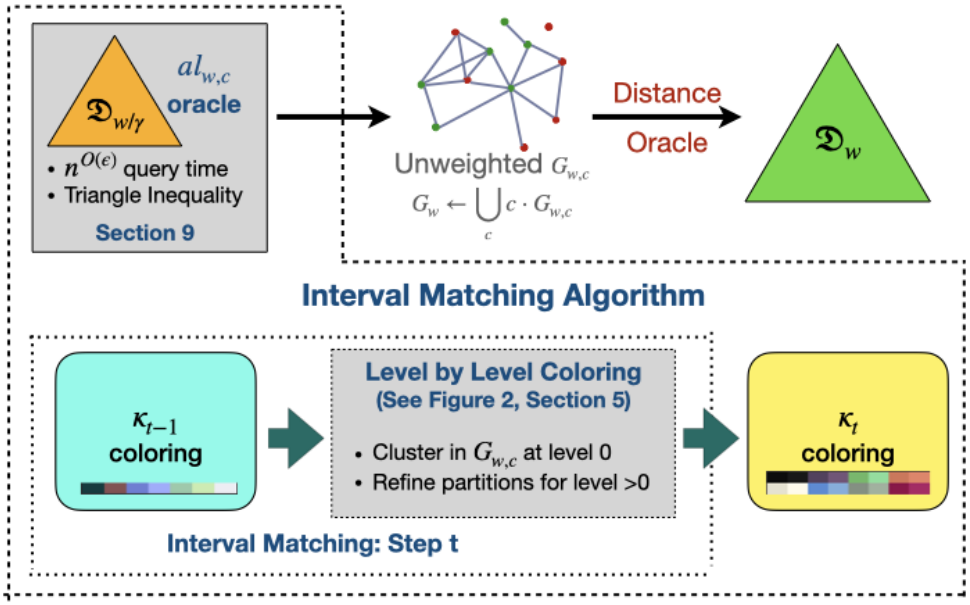


Figure 1: High Level Algorithm Scheme (Section 3.3)

A direct implementation of partitions as above however runs in various issues, yielding additive errors. In particular, it only guarantees to correctly partition "any fixed π-matchable pair with some probability"; instead of the needed "with some probability, all except a few π-matchable pairs are partitioned correctly" (akin to the "for each" vs "for all" guarantees). For the latter goal, bounding the "except a few" so that it's only a $O_\epsilon(1)$-factor approximation, we use the notion of *corruption*, defined later.

Colorings. To describe a partition, we use the slightly generalized concept of a *coloring*: a coloring κ is a mapping from each interval $I \in \mathcal{I}$ to a *distribution of colors* in a color-set ν, where a fixed color should be thought of as a part. We denote the mapping by $\mu_\kappa : \mathcal{I} \times \nu \to [0,1]$. For each interval I, we require $\|\mu_\kappa(I,*)\|_1 = 1$,

i.e., we think of the interval I as being split into fractions each assigned to a part: fraction $\mu_\kappa(I, \chi) > 0$ is assigned to color χ. While under the "perfect neighborhood assumption", standard partitions are sufficient (i.e., $\mu_\kappa \in \{0, 1\}$), fractional colorings will be crucial for removing the assumption later.

Most of the colors $\chi \in \nu$ correspond to a part constructed from a fixed anchor (i.e., its cluster of intervals together with the local extension) with the exception of two special "colors": $\nu = [\lambda^t] \cup \{\perp, \mathbf{u}\}$. First, the color $\perp$ that corresponds to the *already-matched* intervals, i.e., intervals for which we've already added a short path to their π-match in the graph $G_{w,c}$ (typically "dense" intervals that have already "converged"). Second, the **u**-color ("uncolored") consists of intervals which so far have failed to be captured in a part and remain "tbd" (here, the progress will be that such intervals gain a certain "sparsity" properties, improving the chances to be colored later). The color **u** will have the same size upper bound in each step as the regular colors, while the color $\perp$ can be arbitrarily large.

Coloring construction via potentials. To construct a new, more refined step-t coloring (from the step-$(t-1)$ coloring), our algorithm assigns *potential scores* to clustered intervals. Using these potential scores, we assign colors[2] to other nearby intervals in their proximity, as suggested above. The main intuition is that a π-matchable pair $(I, \pi[I])$ typically has a large set of other π-matchable $(J, \pi[J])$ in its respective proximity (i.e., to the left/right).

How large of a "proximity" a cluster can color depends on the size of the `ad`-neighborhood of A (its cluster size). To quantify this, we introduce the notion of *den-*

[2]We'll often just use the verb "color" to describe that process.

sity of an interval I of color χ, termed $\mathrm{d}(I,\chi)$: the measure μ of its $\mathtt{ad}$-neighborhood $\mathcal{N}_c(I)$ that share the color χ. If an interval I (and hence its aligned $\pi[I]$) is "dense" (large $\mathcal{N}_c$), then we have a higher probability to cluster such a pair to an anchor; but we can only afford a small extension for each one (i.e., each clustered interval is used to color few other proximal intervals). In contrast, "sparse" matches will be clustered with a small probability, but can be used to generate large extensions in their proximity.

In particular, to compute the new step-t coloring, we color the intervals gradually in levels, indexed by $l = 0, 1, \ldots, 1/\epsilon$, each level taking care of a density scale. In each level l, we define potentials ϕ and φ. First, for each anchor A (sampled using the coloring constructed in the previous step) corresponding to color χ, we allocate potential $\phi(I,\chi) \approx \frac{n}{d\cdot\lambda^t}$ to each clustered interval I (i.e., I at distance $O(c)$ from A in the same "part"[3] as A) of density d. Next, we define a derivative potential $\varphi(*,\chi)$ by splitting the allocated potential $\phi(I,\chi)$ across the **u**-colored intervals in a proximity ball of radius $\zeta = \zeta(l) \approx \beta^l$, where $\beta = n^\epsilon$, around each clustered interval I. At the end of each level, we augment $\mu_\kappa(I)$ by replacing some of the **u**-color mass with other colors, proportional to the potential vector $\varphi(I,*)$ (to be discussed further later). Overall, the following is a high-level diagram of algorithm updating a $(l-1)$-level coloring κ into an "amended" l-level coloring $\hat{\kappa}$ (all at the same step t):

$$\mu_\kappa \xrightarrow{\text{clustering}} \phi_\kappa \xrightarrow{\text{extension: splitting in the proximity ball}} \varphi_\kappa \xrightarrow{\text{minhash (under PNA)}} \mu_{\hat{\kappa}}$$

A more detailed diagram and a step-by-step coloring example (both require more detailed technical setup) are presented, respectively, in Section 3.4, Figure 2 and

[3] More precisely, to the fraction of I that shares a specific sampled color χ with A.

Figure 3.

In each level $l \geq 1$, our goal is to color all intervals of density in a certain range $[n^{-\epsilon}, 1] \cdot d_l$, where $d_l \approx \frac{n/\lambda^t}{\beta^l}$, for $\beta = n^\epsilon$, as long as there are sufficiently many intervals of that density range overall[4]. At level $l = 0$, corresponding to the highest density for step t, we generate a 2-hop path for each pair of intervals in A's cluster by "planting a star" in G; i.e., adding an edge between the anchor A and each corresponding clustered interval I. Then, we mark (fraction of) I as "already-matched" with the color $\perp$. In other levels, we color intervals found through extensions of clusters using φ.

The remaining case — "sparse" intervals that we did not yet color via an anchor cluster extension as above — will be addressed by the careful use of **u**-color, described next. We remark that, at the end of step t, there may be left some pairs of small densities which we still could not color, and are left **u**-colored, and we will show a bound on those as well. We now expand on the latter.

Controlling sparse sections: the u-color. In order to carry out the level-by-level coloring, we use the special color **u** (for un-colored). This color should be thought of as a "part" in the partition as well. At the beginning of a step, at level $l = 0$, all (fractions of) intervals which are not "already-matched" are assigned the **u** color, and (fractions of) intervals are moved from **u**-color to "standard" colors $\in [\lambda^t]$ as levels progress. The **u**-color helps with three aspects. First, it provides a way to track *sparse intervals* which cannot yet be colored and hence left pending for future levels. Second, if some sparse sections of intervals are never colored in the current step, then

[4]Notice there can be multiple matches in a ball, hence the quantity we care about (and bound) is the *relative density* which is the ratio between "global" and "local" densities.

these intervals will remain **u**-colored (and hence, at the end of the current step, form a part that is also bounded in size). Third, and more nuanced, it allows us to "group together" sparse sections of intervals that are far apart (in their starting index).

In particular, such grouping of far intervals is done using the aforementioned "proximity balls", formally defined via Λ-*balls*: $\Lambda^{\zeta}_{\kappa}(I)$ is the largest interval ball around I containing $\zeta \approx \beta^l$ **u**-colored ℓ_1-mass on both left and right of I. Note that Λ-balls can contain a significantly larger set of intervals than ζ (if the in-between intervals are mostly colored $\neq$ **u**). At the same time, the ball contains at most 2ζ mass of **u**-colored intervals, meaning that the potential $\phi(I)$ is distributed to a mass $\mu \leq 2\zeta$ of intervals, ensuring that, were I to be "corrupted" (e.g., $\pi[I] = \perp$, or the pair $(I, \pi[I])$ happens to be already separated), we will incur only an $O(1)$ factor of total *corrupted potential* to φ's of intervals in the Λ-ball of I.

To showcase the use of **u**-color, consider again the running example introduced in Section 3.2.1. In the early steps t, our algorithm will first color the dense intervals (i.e., via clustering/proximity balls, at lower levels l), leaving the sparse sections mostly unaffected, all colored in **u** (during such early steps, the dense intervals are partitioned into progressively smaller parts while most of the sparse ones remain **u**-colored). Now consider a step t where the part sizes so far are $\lesssim n/\lambda^{t-1} \approx \Delta n$ and we sample $\gtrsim 1/\Delta$ anchors. At the lower levels l, the dense intervals will be partitioned further (continuing the process from the previous steps) and assigned a color $\chi \in [\lambda^t]$. However, when we reach the high levels l, and $\zeta \approx \beta^l = n^{\epsilon l}$ is close to Δn, some fraction of the sparse intervals will be clustered. Furthermore, since at that point the dense intervals are already colored (and have little **u**-color), the Λ-balls around the clustered sparse intervals will be wide and cover most of the **u**-colored

sparse intervals. That allows us to finally partition the sparse intervals into smaller parts as well.

Keeping track of errors in analysis: Corruption. To measure and bound errors, in particular, π-matched interval that are separated, we use a formal notion of corruption. First we define what it means for a (interval, color) pair to be corrupted. Below, $F \gg 1$ is a "distortion" factor (used for the non-PNA case), which one can think of being $F = 1$ (enough for the PNA case).

Definition 19 (Corrupted pairs)**.** *Fix alignment $\pi \in \Pi$, interval $I \in \mathcal{I}$, distortion $F \geq 1$, scale $c \geq 1$, index κ, graph G on $\mathcal{I}$. For a color $\chi \in \nu$, we say (I, χ) is a* (F, κ, π, G, c)-corrupted pair *if any of the following holds:*

1. $\pi[I] = \perp$;
2. $\mathtt{ad}(I, \pi[I]) > c$;
3. $\chi \neq \perp$ *and* $\mu_\kappa(I, \chi) > F \cdot \mu_\kappa(\pi[I], \chi)$; *or*
4. $\chi = \perp$ *and* $(I, \pi[I])$ *are at hop-distance* > 2 *in* G.

For each interval $I \in \mathcal{I}$, we also define *corruption parameter* $\xi_F^{\kappa,\pi,G,c}(I) \in [0, 1]$ as follows:

$$\xi_F^{\kappa,\pi,G,c}(I) = \sum_{\chi:(I,\chi) \text{ is } (F,\kappa,\pi,G,c)\text{-corrupted pair}} \mu_\kappa(I, \chi). \tag{3.1}$$

In particular, an interval I is fully corrupted (in a coloring κ) if it does not have its π-matchable counterpart; and otherwise I is corrupted by the total ℓ_1 color-mass of $\mu_\kappa(I, *)$ where there is insufficient corresponding mass in $\mu_\kappa(\pi[I], *)$ (intuitively, the distribution of colors is too different). While our statements hold for any alignment in Π, we only care about a fixed optimal alignment π, a single graph $G = G_{w,c}$ and a fixed cost c. Hence, for ease of exposition, we say I is *F-corrupted* and the corruption is

$\xi_F^\kappa(I) \triangleq \xi_F^{\kappa,\pi,G,c}(I)$. Our main goal is to bound the total corruption $\xi_F^\kappa \triangleq \xi_F^\kappa(\mathcal{I})$, and in particular show it grows by at most a constant factor in any level/step. Our algorithm runs for a constant number of levels/steps, and hence finishes with corruption which is proportional to the number of intervals without a π-matchable counterpart (starting corruption), upper-bounded by $O_\epsilon(\frac{w}{c} \cdot \mathtt{ed}(x, y))$. Also, at the end of the interval matching algorithm, all intervals are "already matched", i.e., all mass is on $\mu_\kappa(\mathcal{I}, \perp)$, and hence the un-corrupted intervals have a 2-hop path to their π-match.

To bound the corruption growth, we also introduce the parameter $\rho(I)$, which measures the "local amount of corruption" of an $\mathbf{u}$-color of interval I, based on the nearby corrupted intervals. In particular, $\rho(I)$ is defined for a Λ-ball $\mathcal{B}$ around I as the ratio of the corruption to the $\mathbf{u}$-mass inside the Λ-ball $\mathcal{B}$ (formally defined in Section 3.5.1):

$$\rho(I) \triangleq \frac{\mu_\kappa(I,\mathbf{u})}{\mu_\kappa(\mathcal{B},\mathbf{u})} \cdot \xi_F^{\kappa'}(\mathcal{B}),$$

where κ is the current/amended coloring (obtained at the previous level of the current step), and κ' is the coloring from the previous step. One can observe that for any fixed ζ radius of Λ-balls, the sum of ρ over $\mathbf{u}$-colored intervals is proportional to the total sum of corruption.

Completing the algorithm under the perfect neighborhoods assumption (PNA). Once we compute the palettes φ of all intervals (as a function of the sampled anchors), we then use them to update μ for the next level. For illustrative purposes, we now complete the algorithm under the PNA, although our general algorithm will differ significantly from the PNA one. Recall that under PNA, all intervals are either at $\mathtt{ad}$-distance $\leq c$ or $\gg c$, and hence the intervals form equivalence classes according

to their c-neighborhood.

Under PNA, we are guaranteed that the uncorrupted π-matchable pairs will get similar potentials —in fact, $\phi(I)$ and $\phi(\pi[I])$ are precisely equal (and non-zero whenever they are clustered by an anchor). More importantly, if we consider the φ palettes of $I, \pi[I]$, which gather the contributions from clusters containing $I, \pi[I]$ in their proximity ball, then one can prove that (the average) ℓ_1 distance between the two φ palettes is bounded as a function of the "local corruption", namely $\rho(I)$ and $\rho(\pi[I])$.

Using the ℓ_1-distance property of φ, we can assign a single color $\chi \in \nu$ to each interval (i.e., $\mu_\kappa(I, *)$ has support one), obtaining disjoint partitions. To generate such a color (for each interval) we can use a random *weighted min-wise hash function* $h \sim \mathcal{H}$ for $\mathbb{R}^\nu$ (say, using [74]) and use it to partition the vectors $\varphi(I, *)$ of all $\mathbf{u}$-colored intervals I. Specifically, sample a minhash $h : \mathbb{R}^\nu \to \nu$ and set the updated coloring to be $\mu_{\widehat{\kappa}}(I, h(\varphi(I))) \leftarrow 1$ (the rest are 0) for all I for which $\mu_\kappa(I, \mathbf{u}) = 1$ at the end of the previous level.

For a glimpse of the analysis, recall that our overall goal is to make each part of the partition smaller (for runtime complexity) with only a constant-factor corruption growth (for correctness); also, we care only to partition areas with large mass of intervals with density in some range $[n^{-\epsilon}, 1] \cdot d_l$ (in a fixed level l). To control the size of parts, we cannot afford to assign the same color to too many intervals; hence we drop from φ all colors of potential below some fixed threshold $o(n^{-\epsilon})$, ensuring that each color appears in the φ palettes of at most $\frac{n^{1+O(\epsilon)}}{\lambda^{t-1}}$ intervals. For bounding the corruption, we note that minhash gives us a bound proportional to the Jaccard distance between $\varphi(I)$ and $\varphi(\pi[I])$, while the ρ bound we have is in ℓ_1 distance. Showing these bounds are close is where the $\mathbf{u}$-color plays a central role. First,

consider an interval I such that $\Omega(\epsilon)$ portion of its proximity ball Λ is composed of intervals of density $\in [n^{-\epsilon}, 1] \cdot d_l$, where Λ is of radius ζ. Then the palette $\varphi(I)$ has ℓ_1 mass $\Omega(\epsilon)$ whp after thresholding since: 1) some intervals in Λ will be clustered whp (they are dense enough), and 2) once clustered, the generated potential is large enough to pass the threshold (since they are not too dense). In this case, we are done (without using the color $\mathbf{u}$): the Jaccard distance is proportional to the ℓ_1 distance between $\varphi(I)$ and $\varphi(\pi[I])$, and hence the probability of separating I from $\pi[I]$ is bounded by the "local corruption" (which overall is bounded by the total corruption). Second, consider the case when $1 - o(\epsilon)$ portion of intervals in the Λ ball are outside the aforementioned density range. Then we add mass to $\varphi(\cdot, \mathbf{u})$ of intervals in the ball Λ, filling it up to reach $\|\varphi(I)\|_1 = \Omega(\epsilon)$ — this will increase the probability that such intervals are mapped (again) to $\mathbf{u}$ (this increases corruption by a factor ≤ 2). This process also guarantees Λ balls at the next level l have $(1 - o(\epsilon l)) \cdot \zeta$ mass of *sparse intervals*, i.e., of density $\lesssim \frac{n/\lambda^t}{\beta^l}$ (since $o(\epsilon)$ fraction of unmatched intervals of higher density were colored $\mathbf{u}$ at each level). One can then prove that, after running this process for $\approx 1/\epsilon$ levels, the set of intervals corresponding to each color, including $\mathbf{u}$, is of size $\lesssim n/\lambda^t$ only.

Since we could not directly extend the minhash construction to the general non-PNA case, we do not present this construction in the paper, but rather use it as an intuition for its "robust" version as we describe next.

3.2.4 Imperfect neighborhoods

To eliminate the perfect neighborhood assumption (PNA), we must rely on the weaker form of transitivity instead, from the triangle inequality: $\mathcal{N}_c(I) \subseteq \mathcal{N}_{2c}(J) \subseteq \mathcal{N}_{3c}(I)$ for

any $J \in \mathcal{N}_c(I)$. Note that the usual ideas to deal with such "weaker transitivity" do not seem applicable here. For instance, if we pick the threshold of "close" in the cluster construction to be uniformly random $\in [c, O(c)]$, there's still a constant probability of separating I from $\pi[I]$. One could instead apply the more nuanced metric random partitions, such as from [157], which would *partition* the metric $(\mathcal{I}_w, \mathtt{ad}_w)$ (thus putting us back into the perfect neighborhood assumption), with the probability of $I, \pi[I]$ ending up in the same part being $\geq n^{-\epsilon}$ — which has been useful in other contexts by repeating such partition $\approx n^{\epsilon}$ times. However, such a process results in a random partition retaining only $n^{1-\epsilon}$ π-matched pairs, which is not enough to reconstruct even those matched pairs (intuitively, the strings are "too corrupted", as if the edit distance is $(1 - o(1)) \cdot n$), making it inapplicable for our algorithm (here again, this challenge would be more manageable if additive approximation were allowed). Overall, dealing with imperfect neighborhoods proved to be a substantial challenge for us, and we develop several first-of-a-kind tools specifically to deal with it.

Eventually, we still sample a cost c_i from some ordered set $E_c = \{c_1, c_2, \ldots\} \subset [c, O_\epsilon(c)]$. Since we want that the cost c_i satisfies that $\mathcal{N}_{c_i}(\mathcal{N}_{c_i}(I)) \subseteq \mathcal{N}_{c_{i+1}}(I)$, we set E_c of costs to be exponentially-growing, i.e., $c_i = \Theta_\epsilon(c) \cdot 3^i$. The formal definition is in Section 3.4. A high-level comparison between the algorithms is provided in Table 1.

PNA Algorithm	**General Algorithm**
ϕ are uniformly distributed within each cluster.	ϕ are computed independently for each interval.
One radius[a] for each level.	Multiple radiuses for each level.
$\varphi(I, \mathbf{u}) \leftarrow \max\{0, 1 - \|\varphi(I, *_{\neq \mathbf{u}})\|_1\}$.	$\varphi(\mathbf{u})$ independently computed using pivots.
ℓ_1 distance for corruption.	`dd` distance for corruption.
Hard thresholding (randomized).	Soft thresholding (deterministic).
Coloring: $\mathbf{u} \to$ a single color by minhash.	Coloring: $\mathbf{u} \to$ multiple colors by normalization.

Table 1: High-level comparison between the PNA and general algorithms.

Distortion Resilient Distance. Relying purely on triangle inequality forces us to assign somewhat different potential scores to I and $\pi[I]$, hence we will quantify the ratio between the two, referring to it as "distortion".

While it may be tempting to try to keep the distortion to a constant factor, it turns out one cannot do that without introducing super-constant factor corruption growth (number of pairs with a large distortion), which is prohibitive for us. To control corruption, we allow the distortion (i.e., the multiplicative difference) between $\phi(I)$ and $\phi(\pi[I])$ for π-matchable pairs can be as high as n^α for some small positive constant $\alpha \ll \epsilon$. However, such a distortion makes it impossible to obtain a bound on ℓ_1 distance between φ's of a π-match, which is proportional to ρ. Instead, we deal with such distortion by employing a *distortion resilient* (robust) version of ℓ_1.

Definition 20. *Fix $p, q \in \mathbb{R}_+^k$. We define the F-*distortion resilient distance*,*

$$\mathtt{dd}_F(p, q) = \sum_{i: p_i > F \cdot q_i} p_i$$

.

This function allows us to define and control *corruption* of (interval, color) pairs by differentiating distortion (which captures multiplicative errors) from corruption (which captures additive ones). As part of our analysis, we will show several basic properties of the `dd` distance and develop `dd`-*preserving soft-transformations*, which will replace the hard thresholds from the minhash construction. Intuitively, `dd` replaces the use of the ℓ_1/Jaccard metric on the vectors ϕ and φ, which was a key enabler for using minhash under PNA. However, `dd` is not a metric in any reasonable sense (it's not even symmetric), rendering the minhash construction obsolete (e.g., it is unreasonable to expect any kind of LSH under `dd`).

Assigning potential to (interval, color) pairs. Since maintaining equivalence classes is essential for our construction, we analyze pairs of interval and colors in $\mathcal{I} \times \nu$ (which, combinatorially, can be thought of as "fractions of intervals"). Thus, when we increase the ϕ-potential of a clustered pair (I, χ'), we do so proportionally to its μ mass, meaning we set the potential to $\approx \mu(I, \chi')\frac{n}{d \cdot \lambda^t}$. Similarly, splitting the ϕ potential to $\mathbf{u}$-colored-pairs in Λ balls (i.e., assigning φ) is done in a pro-rated fashion, weighted according to respective $\mu(\cdot, \mathbf{u})$ masses.

Assigning u potential: pivot sampling. Having so far discussed assigning of the non-$\mathbf{u}$ colors in φ of (fractions of) intervals, we now discuss how to assign $\mathbf{u}$-color in φ_κ and, eventually, amended coloring $\mu_{\widehat{\kappa}}$. It may be tempting to merely subtract the assigned fraction of non-$\mathbf{u}$-color from the $\mathbf{u}$-color mass, but this would result in additive errors (for the color $\mathbf{u}$), while `dd` only allows for multiplicative distortion. In particular, we need $\mathbf{u}$ colors to agree up to a fixed distortion as well (to avoid more corruption), and hence we compute the new $\mathbf{u}$-color mass directly,

via a different technique for explicitly measuring *sparseness* (which is the central purpose of **u**-color). To accomplish this measurement, we developed a procedure called *pivot sampling*, which somewhat resembles the way we assign potentials for the non-**u** colors. First, we downsample $\mathcal{I} \times \nu$ into a smaller set of *pivots* $\mathcal{V}$. Second, we *approximate the density* of each pivot in $\mathcal{V}$, for each possible cost in E_c, thus generating θ*-potential* scores for each pivot. Third and last, such θ potential is split among the intervals in a Λ-ball in a similar fashion to how we split ϕ, generating $\varphi(\cdot, \mathbf{u})$ potentials to *intervals in sparse areas*. This rather involved process, specific to dealing with imperfect neighborhoods, requires much care to be able to control: (1) corruption of **u**-colors; (2) balance of palettes φ (as we describe next); (3) sparsity guarantees for **u**-color (part) at the end of the step t; and (4) computational efficiency of such sampling mechanism.

Amending a coloring in a level, using φ. While `dd` is a convenient analytical tool for bounding corruption, it lacks the basic properties to allow coordinated sampling between π-matchable pairs. Instead of sampling a color from $\varphi(I, *)$ (as was done under PNA), we add *all* colors in $\varphi(I, *)$ to the amended coloring $\mu_{\widehat{\kappa}}(I, *)$. To maintain a distribution of colors, we first combine φ with pre-existing non-**u**-colors in μ, and then normalize to have ℓ_1-mass of $\mu_\kappa(I, \mathbf{u})$, i.e., what "remains to be colored", thus ensuring that the overall amended $\mu_{\hat{\kappa}}(I, *)$ is a distribution. As in the PNA case, we need to bound extra *corruption from normalization* by ensuring that the palettes φ have constant norms. While this analysis for the PNA solution is immediate (by construction), here, instead, we employ several combinatorial arguments that analyze mass of pairs with certain density over certain set of costs, eventually showing that in

each level, we either add sufficient regular colors (corresponding to anchors/clusters) or **u**-colors to all intervals while maintaining guarantees (1)–(4) above.

Controlling the growth of distortion. Our arguments require that throughout the matching phase, the $\mathtt{dd}$ distortion F is bounded by $n^{o(\epsilon)}$ (in particular, to maintain control over the aforementioned soft-transformations). Many of our algorithmic steps generate extra distortion. To control both distortion (multiplicative error) and corruption (additive error), we parametrize maximum distortion $F = F(t, l)$ for each step/level a priori, and bound corruption ξ_F^κ at each step/level using the predetermined distortion parameter $F = F(t, l)$. The final approximation factor is a function of the maximum cost in $\frac{1}{c}E_c$ (which is further determined by the "base distortion" $F(1, 0) = n^\alpha$), together with the corruption factor we show in each step. At the end of the day, a distortion F bounded by $n^{o(\epsilon)}$ allows us to carry out the above arguments (i.e., some of the above arguments can only work under small distortion F).

3.2.5 The metrics $\mathtt{ad}_w$

We now briefly discuss the algorithm for computing the $\mathtt{ad}_w(I, J)$ distance, using oracle calls to $\mathfrak{D}_{w/\gamma}$ metric. This metric is used to compute distances when building the graph G_w, in the Interval Matching algorithm. Note that the latter makes $n^{1+O(\epsilon)}$ oracle calls to $\mathtt{ad}$, and hence the algorithm has to run in time $n^{O(\epsilon)} = \mathrm{poly}(\gamma)$.

Intuitively, $\mathtt{ad}_w(I, J)$ is meant to capture the following distance, which should be thought of as an extension of the edit distance over an alphabet with the metric

$(\mathcal{I}_{w'}, \mathfrak{D}_{w'})$ where $w' = w/\gamma$:[5]

$$\min_{\pi} \sum_{i \in [0,w)} \tfrac{1}{w'} \mathfrak{D}_{w'}(I + i, J + \pi(i)),$$

where π ranges over all alignments of indices of I to indices of J. One can show that essentially, if $\mathfrak{D}(I, J) = \mathtt{ed}(I, J)$, then, for $I = X_{i,w}, J = Y_{j,w}$, the above distance is between $\Omega(\mathtt{ed}(X_{i,w}, Y_{j,w}))$ and $O(\mathtt{ed}(X_{i,2w}, Y_{j,2w}))$.[6]

However, this distance function is hard to compute fast: not only it is as hard as computing edit distance on w-length strings, but even linear time (in $w \gg \mathrm{poly}(\gamma)$) is too much for us. In particular, it does not use the fact that $\mathfrak{D}_{w'}(I + i, J + \pi(i))$ captures the information of blocks of length w'. Hence, it is natural to approximate the above by considering a "rarefication" of the above sum as follows:

$$\tfrac{w}{\gamma} \cdot \min_{\pi} \sum_{i \in [0,\gamma)} \tfrac{1}{w'} \mathfrak{D}_{w'}(I + iw', J + \pi(iw')). \tag{3.2}$$

However, the latter will not satisfy the triangle inequality — which is crucial in the Interval Matching Algorithm — and in fact is not even symmetric: e.g., if the optimal $\pi(i) = i+1$, then $\mathtt{ad}_w(J, I)$ would be using $\mathfrak{D}$ on completely different arguments. This is especially an issue since $\mathfrak{D}$ may substantially over-estimate $\mathtt{ed}$ on some of the pairs (and hence "shift by one" can change the distance a lot).

Indeed, ensuring triangle inequality is the main challenge for defining and computing $\mathtt{ad}_w$ here. We manage to define an appropriate distance $\mathtt{ad}$, satisfying triangle inequality for "one scale only" metrics $\mathtt{ad}_{w,c}$, designed for distances in the range $\approx [c, \gamma c]$, which turns out to be enough for the Interval Matching algorithm.

[5] $I + i$ means the interval starting i positions to the right of the start of I; when, e.g., $I = Y_{\perp,w}$ then $Y_{\perp,w} + i = Y_{\perp,w}$.

[6] While we are not aware of an explicit proof of this statement, it is in the spirit of statements that appeared in, e.g., [175, Lemma 5], [20, Lemma 3.2], [21, Theorem 3.3].

First, we note that we have two different algorithms, corresponding to two distinct distance formulations: (i) for large distance regime, where $c > w/\gamma^2$; and (ii) for small distance regime, where $c \leq w/\gamma^2$. The reason there's a big difference between the two cases is that when $c > w/\gamma^2$, the alignment π may have a large displacement $|i - \pi(i)| \geq w/\gamma$, bigger than the length of "constituent" intervals for which we have the base metric $\mathfrak{D}_{w'}$. Hence, for the "large distance" regime, when $c \geq w/\gamma$, we uses a slightly different (and simpler) algorithm that runs in time $\approx \text{poly}(w/c)$, and hence is only good when c is sufficiently large.

Finally, we sketch the harder, $\text{poly}(\gamma)$-time algorithm, for not-so-large c. The idea is to allow alignment shifts in both intervals. More formally, let $T = \gamma^3$ and let $\mathcal{A}$ be the set of functions $A = (A_x, A_y)$ where $A_x, A_y : [-\gamma, \gamma] \to \{0, 1, \ldots, T-1\}$ are non-decreasing functions with $A_x[-\gamma] = A_y[-\gamma] = 0$ and $A_x[\gamma] = A_y[\gamma]$. We define the distance $\mathtt{ad}_{w,c}(I, J)$, to be, where θ is essentially $\approx c/w$ and $\mathfrak{S}_i = \{0, 1, \ldots, 3T - A_x[i] - A_y[i] - 1\}$:

$$\mathtt{ad}_{w,c}(I,J) \triangleq \min_{A \in \mathcal{A}} \left(A_x[\gamma] + A_y[\gamma]\right) \theta w' + \sum_{i \in [-\gamma,\gamma)} \frac{1}{T} \sum_{\Delta \in \mathfrak{S}_i} \mathfrak{D}_{w'}\left(I + w'(i + \theta(\Delta + A_x[i])), J + w'(i + \theta(\Delta + A_y[i]))\right).$$

Intuitively, ignoring Δ-sum (i.e., think $\Delta = 0$), we obtain an alignment of I to J where the starting positions (of w'-length intervals) are close to multiples of w' in *both* strings (as opposed to only one string, as in Eqn. (3.2)). While allowing such an alignment is enough for ensuring symmetry, it is still not enough to ensure triangle inequality. Consider intervals $I, J, K \in \mathcal{I}$ where we want to guarantee that $\mathtt{ad}(I, K) \leq \mathtt{ad}(I, J) + \mathtt{ad}(J, K)$. Apriori, there is no way to ensure an optimal alignment between $\mathtt{ad}(I, J)$ and $\mathtt{ad}(J, K)$ will use the same shifts and hence makes it hard to offset the

distance of large blocks in $\texttt{ad}(I, K)$. To solve the inconsistencies between the shifts, we use the Δ-sum over all possible shifts (this is yet another instance of "averaging it out"). The last definition can also be computed in poly(γ) time by a standard dynamic programming.

Nonetheless, the following issue remains: think of the case when $\texttt{ad}(I, J)$ and $\texttt{ad}(J, K)$ use the maximally-allowed values of the alignment (namely γ^3), in which case $\texttt{ad}(I, K)$ cannot use the natural composition of the two alignments (since it's out of bounds). To solve this issue, we upper bound each $\texttt{ad}_{w,c}$ with a maximal value $\gg c$ (which solves the triangle inequality issue), and define the (non-metric) distance $\texttt{ad}_w(\cdot, \cdot)$ as the summation over all costs $c \in S_w$ which can be upper-bounded by $\texttt{ad}_{w,c}$ (formal definition in section 3.8).

Finally, we remark that, at the end of the day, we cannot guarantee a per-pair upper bound on $\texttt{ad}(I, J)$, but only on average, and only when comparing $X_{i,w}$ against $Y_{\pi(i),w}$ (although the triangle inequality is true everywhere). This is, nonetheless, just enough for estimating $\texttt{ed}(x, y)$.

3.3 Top level algorithm

We now describe our "top-level" algorithm. We assume here that the first $(1-o(1))n$ positions of x and y are equal; we can remove this assumption by padding x, y with some fixed unique character \$, increasing the size of x, y by a factor of, say, $O(\log n)$.

Our algorithm consists of $\log_\gamma n$ iterations, where $\gamma = n^\epsilon$. For each $w = \gamma^i$, $i \in [\log_\gamma n]$, we construct the metric $(\mathcal{I}_w, \mathfrak{D}_w)$, which *align approximates* the metric $(\mathcal{I}_w, \texttt{ed}_w)$ (see Def. 18). Each iteration consists of two components: the *alignment*

distance algorithm, and the *interval matching algorithm*, described in later sections. Below we assume that n is a power of γ, which is without loss of generality (as we can increase n appropriately by padding the strings).

Alignment Distance algorithm. Assuming oracle access to $(\mathcal{I}_{w/\gamma}, \mathfrak{D}_{w/\gamma})$, the metric constructed at the previous iteration, our alignment algorithm is an oracle for computing the distance $\mathtt{ad}_w(\cdot,\cdot)$ on w-length intervals. In fact, we have $O(\log n)$ such distance measures, $\mathtt{ad}_{w,c}$, one for each target cost scale $c \in S_w = \{1, 2, 4, \ldots w\} \cup \{1/2, 1/4, \ldots, 1/2n\}$. Each such function $\mathtt{ad}_{w,c}(\cdot,\cdot)$ evaluation is an edit-distance-like dynamic programming of size $\mathrm{poly}(\gamma)$, and overall can be computed using $\mathrm{poly}(\gamma) = n^{O(\epsilon)}$ time and oracle calls to $\mathfrak{D}_{w/\gamma}$.

Note that this algorithm does not run directly, but instead is used as an oracle inside the matching algorithm described next.

Interval Matching algorithm. We construct a weighted graph G_w on $\mathcal{I}_w$, such that the shortest path distance in G_w approximates the $\mathtt{ad}_w$ distance on intervals. Again, this won't be achieved for all pairs of intervals, but only for interval pairs that "matter", i.e., that are in an optimal alignment for $\mathtt{ed}(x, y)$. The graph G_w is the union of edges of the graphs $G_{w,c}$, for $c \in S_w$, each of them align approximating $\mathtt{ad}_{w,c}$ at "scale c". Constructing the graphs $G_{w,c}$ is the heart of the matching algorithm.

Once we have the graph G_w, we build a fast distance oracle data structure on it, using the ℓ_∞ embedding of [151], and whose output is the desired metric $\mathfrak{D}_w$. Overloading the notation, we call $\mathfrak{D}_w$ both the distance oracle data structure as well as the metric it produces. In particular, [151] shows how one can embed any n-point metric into ℓ_∞ of dimension $d = O(\frac{1}{\epsilon} n^\epsilon \log n)$ while incurring $O(1/\epsilon)$ distortion only.

Once we have such an embedding, we can compute distance between two points by evaluating ℓ_∞ distance in dimension d. Furthermore the embedding itself can be computed by running d single-source shortest path (SSSP) computations. Hence, we obtain the following Theorem 21 as an immediate corollary of [151] together with a standard $\tilde{O}(m)$-time SSSP algorithm.

Theorem 21 (corollary of [151])**.** *For any $\epsilon \in (0, 1/2)$, given any weighted graph G on n nodes and m edges, we can build a distance oracle data structure with the following properties:*

- *supports distance queries: given u, v, output $\mathfrak{D}_G(u, v)$ which is a $(4/\epsilon)$-factor approximation to the shortest path distance between u, v in the graph;*
- *$\mathfrak{D}_G(u, v)$ is a (n-point) metric;*
- *runtime per query is $\tilde{O}_\epsilon(n^\epsilon)$;*
- *data structure uses $\tilde{O}_\epsilon(n^{1+\epsilon})$ space, and pre-processing time is $\tilde{O}_\epsilon(mn^\epsilon)$.*

Top-level algorithm is described in Algorithm 1. At the beginning, when $w = \gamma$, we use the metric $(\mathcal{I}_1, \mathfrak{D}_1)$, which is just the metric on all positions in x and y, where two positions are at distance 0 iff the positions contain the same character, and 1 otherwise. At the end, when $w = n$, we can extract the distance between x and y, which is our final approximation. The algorithm MatchIntervals, described in Section 3.4, returns an unweighted graph $G_{w,c}$. The full graph G_w for the scale w, is obtained by union of the graph edges of $G_{w,c}$ each scaled by $C_m c$, over all $c \in S_w$, together with some extra edges.

Algorithm 1 EstimateEditDistance(x, y, ϵ, n)

function EstimateEditDistance(x, y, ϵ, n)

Fix C_m to be the constant from Theorem 22.

$\gamma \leftarrow n^{\epsilon}$.

$\mathfrak{D}_1$ is a data structure that, given two positions into x and/or y, outputs 0 iff the characters in those positions are equal and 1 otherwise.

for $w \in \{\gamma, \gamma^2, \ldots, n\}$ **do**

Let $\mathcal{W}_w, \mathcal{H}_w$ be sets of all w-length intervals on x-axis and y-axis respectively, with $\mathcal{I}_w = \mathcal{W}_w \cup \mathcal{H}_w$.

for $c \in S_w$ **do**

$G_{w,c} \leftarrow$ MatchIntervals$(\mathfrak{D}_{w/\gamma}, c)$.

end for

$G_w \leftarrow \cup_{c \in S_w} C_m \cdot c \cdot G_{w,c}$ and add edges $(X_{i,w}, X_{i+1,w}), (Y_{i,w}, Y_{i+1,w})$ with unit cost for all i, as well as a special node $\perp$ that connects to every other node with an edge of cost $C_m \cdot w$.

$\mathfrak{D}_w \leftarrow$ data structure from Theorem 21 on graph G_w for approximation $4/\epsilon$.

end for

return $\mathfrak{D}_n(X_{i,n}, Y_{i,n})$ for a randomly chosen $i \in [n]$.

end function

3.3.1 Main guarantees

The guarantees of the algorithm follow from the following two central theorems.

Theorem 22 (MatchIntervals; see Sections 3.4, 3.5, 3.6, 3.7)**.** *Fix $\epsilon > 0$, $w, n \in \mathbb{N}$, and cost $c \in S_w$. Suppose $(\mathcal{I}_w, \mathtt{ad}_{w,c})$ is a metric, for which we have query access running in time $T_{\mathtt{ad}}$. Then, the algorithm* MatchIntervals *builds an undirected graph $G_{w,c}$, over intervals $\mathcal{I}_w$, such that:*

1. *For all edges* $(I, J) \in G_{w,c}$, *we have* $\mathtt{ad}_{w,c}(I, J) \le C_m \cdot c$, *where* $C_m = C_m(\epsilon)$ *is a constant.*

2. *For any alignment* $\pi \in \Pi$, *with high probability:* $|\{i \mid \mathrm{dist}_{G_{w,c}}(X_{i,w}, Y_{\pi(i),w}) > 2\}| \le C_m \cdot |\{i \mid \mathtt{ad}_{w,c}(X_{i,w}, Y_{\pi(i),w}) > c\}|$[7], *where* $\mathrm{dist}_{G_{w,c}}$ *is the hop-distance in* $G_{w,c}$.

3. *The runtime of the algorithm is, by expectation,* $O_\epsilon(T_{\mathtt{ad}} \cdot n^{1+O(\epsilon)})$.

As described above, using the algorithm from the above theorem, we build a graph G_w, which is the union of scale graphs $G_{w,c}$. Then we take $\mathfrak{D}_w$ to be the fast distance oracle of the shortest path on the graph G_w, using Theorem 23.

Next theorem says that, given access to $\mathfrak{D}_{w/\gamma}$, we can compute $\mathtt{ad}_{w,c}(I, J)$ for any two intervals $I, J \in \mathcal{I}$, which corresponds to the "natural extension" of $\mathfrak{D}$ from length-w/γ to length-w substrings.

Theorem 23 (alignment distance ad; see Section 3.8)**.** *Fix* w *and* $w' = w/\gamma$, *as well as constants* $C, C' \ge 1$, *and suppose we have a data structure for a metric* $\mathfrak{D}_{w'}$ *that* C*-align-approximates* ed, *while also* $\mathfrak{D}_{w'}(X_{i,w'}, X_{i+1,w'}) \le C'$ *for all* $i \in [n]$ *(and same for* Y *intervals). Then, for any* $c \in S_w$, *the algorithm from Section 3.8 defines a function* $\mathtt{ad}_{w,c}(\cdot,\cdot)$ *on* $\mathcal{I}_w \times \mathcal{I}_w$ *with the following properties:*

1. $(\mathcal{I}_w, \mathtt{ad}_{w,c})$ *is a metric;*

2. *For all* $X_i, Y_j \in \mathcal{I}_w$, $\mathtt{ad}_{w,c}(X_i, Y_j) \ge \min\{\mathtt{ed}_w(X_i, Y_j), c\sqrt{\gamma}\}$;

3. *Define* $\mathtt{ad}_w(X_i, Y_j) \triangleq \sum_{c \in S_w} c \cdot \mathbb{1}[\mathtt{ad}_{w,c}(X_i, Y_j) \ge c]$. *Then,* $\mathtt{ad}_w$ $O(CC')$*-align-approximates* ed.

[7] Recall from the preliminaries that $\mathtt{ad}_{w,c}(X_i, Y_\perp) = w$.

4. *For all* $I, J \in \mathcal{I}_w$, $\mathtt{ad}_{w,c}(I, J)$ *can be computed using* $O(\gamma^{O(1)})$ *time and queries to* $\mathfrak{D}_{w'}$.

We remark that $\mathtt{ad}_w$ is not guaranteed to be a metric, which is the reason why we use $\mathtt{ad}_{w,c}$ in the theorem statement. Also, the algorithm from Section 3.8 requires no further preprocessing.

3.3.2 Proof of Theorem 14

To prove Theorem 14, we just combine the above two theorems, 22 and 23. In particular, the inductive hypothesis is that, for $w = \gamma^i$, where $i \in [\log_\gamma n]$, the distance oracle data structure $\mathfrak{D}_w$ outputs a metric $\mathfrak{D}_w$ with the following properties, for some constant $C_w = C(\epsilon, \log_\gamma w)$, whp:

1. $\mathfrak{D}_w$ C_w-align-approximates $\mathtt{ed}$;

2. $\mathfrak{D}_w(X_{i,w}, X_{i+1,w}) \le 4/\epsilon$ (and same for Y intervals).

Base case: for $w = 1$, this is immediate by construction of $\mathfrak{D}_1$.

Now assume the inductive hypothesis for $w' = w/\gamma$ and we need to prove it for w. By inductive hypothesis, $\mathfrak{D}_{w'}$ satisfies hypothesis of Theorem 23, and hence we can apply it to obtain an oracle query to metrics $\mathtt{ad}_{w,c}$; each oracle query takes $O(\gamma^{O(1)})$ time. Let π be optimizer for $\min_\pi \sum_i \mathtt{ad}_w(X_{i,w}, Y_{\pi(i),w})$ (guaranteed to align-approximate $\mathtt{ed}$).

Define $\tau(\cdot, \cdot)$ to be the distance in the graph G_w constructed in the algorithm. We will build a fast distance oracle $\mathfrak{D}_w$ on the graph G_w (using Theorem 21), meaning that its output metric $\mathfrak{D}_w$ satisfies $\tau \le \mathfrak{D}_w \le 4\tau/\epsilon$.

To prove the first property of the inductive hypothesis, consider any two intervals I, J, and the shortest path $v_1, \dots v_k$ between them, where $v_1 = I$ and $v_k = J$. We have that (v_i, v_{i+1}) is an edge in some graph G_{w,c_i}, or is an extra edge of cost 1; call E the set of the latter i's. Hence the cost $\tau(I, J) = \sum_{i \notin E} C_m c_i + |E|$. For $i \notin E$, by Theorem 22 and Theorem 23, we have that $C_m c_i \geq \mathtt{ad}_{w,c_i}(v_i, v_{i+1}) \geq \mathtt{ed}_w(v_i, v_{i+1})$ (note that the other part of the min cannot happen as $C_m \ll \sqrt{\gamma}$). Also, for $i \in E$, we have that $1 = \tau(v_i, v_{i+1}) \geq \mathtt{ed}_w(v_i, v_{i+1})$ as $\mathtt{ed}_w(X_j, X_{j+1}) \leq 1$ for all j (and same for Y's). Hence $\mathfrak{D}(I, J) \geq \tau(I, J) = C_m \sum_{i \notin E} c_i + |E| \geq \mathtt{ed}_w(v_1, v_2) + \ldots + \mathtt{ed}_w(v_{k-1}, v_k) \geq \mathtt{ed}_w(I, J)$.

Next, we note that $\tau(I, J)$ is upper bounded by $2C_m \min_{c \in S_w} c + \sum_{c \in S_w} 4 \cdot C_m c \cdot \mathbb{1}[\mathrm{dist}_{G_{w,c}}(I, J) > 2] = C_m/n + 4C_m \sum_{c \in S_w} c \cdot \mathbb{1}[\mathrm{dist}_{G_{w,c}}(I, J) > 2]$. Hence:

$$\sum_{i \in [n]} \tau(X_{i,w}, Y_{\pi(i),w}) \leq C_m + 4C_m \sum_{c \in S_w} \sum_{i \in [n]} c \cdot \mathbb{1}[\mathrm{dist}_{G_{w,c}}(X_{i,w}, Y_{\pi(i),w}) > 2].$$

For fixed $c \in S_w$, we have that, by Theorem 22:

$$\sum_{i \in [n]} \mathbb{1}[\mathrm{dist}_{G_{w,c}}(X_{i,w}, Y_{\pi(i),w}) > 2] \leq C_m \cdot \sum_{i} \mathbb{1}[\mathtt{ad}_{w,c}(X_{i,w}, Y_{\pi(i),w}) > c].$$

Therefore, for every π:

$$\begin{aligned} \sum_{i \in [n]} \tau(X_{i,w}, Y_{\pi(i),w}) &\leq O_\epsilon \left(1 + \sum_{c \in S_w} \sum_{i \in [n]} c \cdot \mathbb{1}[\mathtt{ad}_{w,c}(X_{i,w}, Y_{\pi(i),w}) > c] \right) \\ &\leq O_\epsilon \left(1 + \sum_{i \in [n]} \mathtt{ad}_w(X_{i,w}, Y_{\pi(i),w}) \right). \end{aligned}$$

Take π to be the optimizer of the right-hand-side summation. Using Theorem 23 again, together with the inductive hypothesis for $w' = w/\gamma$, we conclude (where

constant depends on ϵ and $\log_\gamma w$):

$$\begin{aligned}\sum_{i\in[n]} \mathfrak{D}_w(X_{i,w}, Y_{\pi(i),w}) &\leq O_\epsilon\left(\sum_{i\in[n]} \tau(X_{i,w}, Y_{\pi(i),w})\right)\\ &\leq O_\epsilon\left(1+\sum_{i\in[n]} \mathtt{ad}_w(X_{i,w}, Y_{\pi(i),w})\right)\\ &\leq O_\epsilon\left(\min_{\pi'}\sum_{i} \mathtt{ed}_w(X_{i,w}, Y_{\pi'(i),w})\right),\end{aligned}$$

since we can assume wlog that $\mathtt{ed}(x,y) \geq 1$ (checking the opposite is immediate), thus completing the proof of the inductive hypothesis.

The second property is immediate by construction of the graph G_w and the fact that approximation of the distance oracle $\mathfrak{D}_w$ is taken to be $4/\epsilon$.

Now we argue that the final output produced by the top-level algorithm is a constant factor approximation. Consider the $\mathfrak{D}_w$ guarantees for $w = n$, and fix the minimizing π, and constant $C_n = C_w$. For a random index $i \in [n]$, with probability at least 0.9, we have that: 1) $i \in [1, n - o(n)]$, 2) $\pi(i) \neq \perp$ (by our assumption on x, y from the start of the section), and $\mathfrak{D}(X_{i,n}, Y_{\pi(i),n}) \leq O(C_n) \cdot \mathtt{ed}(x,y)$ (by Markov). Furthermore note that $|i - \pi(i)| \leq C_n \cdot \mathtt{ed}(x,y)$, and hence, $\mathfrak{D}(X_{i,n}, Y_{i,n}) \leq \mathfrak{D}(X_{i,n}, Y_{\pi(i),n}) + \frac{4}{\epsilon} \cdot |i - \pi(i)| \leq O(C_n/\epsilon) \cdot \mathtt{ed}(x,y)$. Also, since $i \leq n - o(n)$, we have that $\mathfrak{D}(X_{i,n}, Y_{i,n}) \geq \mathtt{ed}(X_{i,n}, Y_{i,n}) = \mathtt{ed}(x,y)$.

Concluding, the algorithm produces a $O(C_n/\epsilon)$ approximation to $\mathtt{ed}(x,y)$, with probability ≥ 0.9. Note that C_n is a constant, depending on ϵ, as we have only a constant number of iterations, each incurring a constant factor approximation.

The runtime guarantee follows from time guarantees of Theorems 21, 22, and 23. In particular, by Theorems 21 and 23, the runtime $T_{\mathtt{ad}}$ is $O(\gamma^{O(1)}) \cdot n^{O(\epsilon)} = n^{O(\epsilon)}$. Hence runtime in the matching algorithm, for every fixed w, is $n^{1+O(\epsilon)}$ by expectation. Thus

the graph G_w has size $n^{1+O(\epsilon)}$ and the preprocessing time of Theorem 21 is $n^{1+O(\epsilon)}$ by expectation as well. Overall, we have a constant number of w's to consider, and thus we obtain an expected runtime of $n^{1+O(\epsilon)}$.

3.4 Interval matching algorithm

In this section we describe our main interval matching algorithm, used to prove Theorem 22. The correctness and runtime complexity analysis will follow in Sections 3.5 and 3.7 respectively.

Our matching algorithm iterates over a constant number of steps t, each iterating over a constant number of levels l. The following MatchIntervals algorithm is the main loop over steps. We note that, in order to maintain high probability statements, in each step we output $O(\log n)$ output colorings for each input coloring.

Algorithm 2 Matching Algorithm

Input: Base cost $\mathfrak{c}$

Output: A matching graph $G = G_{w,\mathfrak{c}}$.

function MATCHINTERVALS($\mathfrak{c}$)
  $K_0 \leftarrow$ {initialize a new coloring of a single color **u** assigned with mass 1 to all $\mathcal{I}$}.
  $G \leftarrow$ unweighted and undirected graph with nodes $I \in \mathcal{I}_w$ and no edges.
  for $t = 1, 2, \ldots$ **do**
    $K_t \leftarrow \emptyset$.
    for $\kappa' \in K_{t-1}$ such that $\mu_{\kappa'}(\mathcal{I}, \nu \setminus \perp) > 0$ **do**
      for $O(\log n)$ times **do**
        $(G, K_t) \leftarrow (G, K_t) \uplus^a$ MATCHSTEP($\mathfrak{c}, \kappa', t$).
      end for
    end for
    break if $K_t = \emptyset$.
  end for
  return G.
end function

[a] By $\uplus$, we refer to the coordinate-wise union, meaning we add the output edges to G and coloring κ to K_t.

It remains to describe the MATCHSTEP algorithm, which iterates over a constant number of levels. In each such level, the algorithm updates the coloring κ under construction, to obtain the "amended" coloring $\widehat{\kappa}$, while also using the "step input coloring" κ', obtained at the end of the previous step. At the end of the iteration over the levels, the algorithm MATCHSTEP produces a number of edges to add to the graph as well as a coloring. The pseudo-code for MATCHSTEP is presented in Alg. 8,

after a detailed description of the mechanisms and subroutines of MatchStep.

3.4.1 Components of MatchStep: setup and notations

We first introduce basic notions used in our algorithm. To help the reader keep track of the many definitions and notations, we summarize them in Table 2 and Table 3 for quick reference (the latter table is in Section 3.5). To avoid confusion, we use $\mathfrak{c}$ to denote c from the theorem statement. Also, to simplify notation, we refer to the oracle $\mathtt{ad}_{w,\mathfrak{c}}$ simply as $\mathtt{ad}$. Note that $\mathtt{ad}$ algorithm uses the data structure $\mathfrak{D}_{w/\gamma}$, which we assume all our algorithms in this section have access to.

The set of costs $E_{\mathfrak{c}}$. For each fixed *base cost* $\mathfrak{c} \in S_w$, we use a fixed set of costs $E_{\mathfrak{c}} \subset \mathfrak{c} \cdot [1, O_\epsilon(1)]$. The set $E_{\mathfrak{c}}$ is defined as $\{c_1, c_2, \ldots c_{1/\eta}\}$, where $c_i = \frac{\mathfrak{c}}{\epsilon \cdot \alpha} \cdot 3^i$, for constants $\alpha = \epsilon^{5/\epsilon}, \eta = \alpha^2\epsilon^3$ (whose significance will be clear later), noting that $1/\alpha, 1/\eta$ are integers by constraints on ϵ.

Colorings. Recall our definition of *coloring* κ over color-set ν as a mapping from intervals to *distribution of colors* in ν, denoted by $\mu_\kappa : \mathcal{I} \times \nu \to [0, 1]$.

Our construction analyzes pairs of intervals and colors $(I, \chi) \in \mathcal{I} \times \nu$. For a set $\mathcal{R} \subseteq \mathcal{I} \times \nu$ of (interval, color) pairs, $\mu_\kappa(\mathcal{R})$ is the ℓ_1 mass of colors in $\mathcal{R}$, i.e. $\mu_\kappa(\mathcal{R}) \triangleq \sum_{(I,\chi)\in\mathcal{R}} \mu_\kappa(I, \chi)$. We often use the shorthand *pairs* when referring to (interval, color) pairs.

We use colorings to partition $\mathcal{I}$ into smaller (overlapping) parts. For a color χ, we denote its part by $\mathcal{P}_\kappa^\chi \triangleq \{I \in \mathcal{I} \mid \mu_\kappa(I, \chi) > 0\}$.

Finally, we equip each μ_κ with a data structure that allows efficient sampling from it (Theorem 49). The reader should henceforth consider that sampling takes

time proportional to the output size, up to $\text{poly}(\log n)$ factors.

Proximity balls Λ. For coloring κ and $\zeta \in \mathbb{N}$, we define $\Lambda_\kappa^\zeta(I)$ as the largest interval ball around I containing at most ζ $\mathbf{u}$-colored ℓ_1-mass on each of left and right of I, where $\mu_\kappa(I, \mathbf{u})$ is counted on both sides. An exception to the above is for $\zeta = 0$, when $\Lambda_\kappa^0(I) = \{I\}$. We note that for all $I \in \mathcal{I}$, $\zeta \geq 1$, we have $\mu_\kappa(\Lambda_\kappa^\zeta(I), \mathbf{u}) \leq 2\zeta$.

Extending the data structure for μ_κ above, we also use the data structure from Theorem 50 to be able to compute the boundaries of any $\Lambda_\kappa^\zeta(I)$, given I, ζ, in $\text{poly}(\log n)$ time.

Interval and Pair densities. For $I \in \mathcal{I}$, κ, color $\chi \in \nu$, interval set $\mathcal{S} \subseteq \mathcal{I}$, and cost $\hat{c} \in E_\mathfrak{c}$, we define a density parameter $\mathrm{d}_\kappa(I, \chi, \mathcal{S})_{\hat{c}} \triangleq \mu_\kappa(\mathcal{N}_{\hat{c}}(I) \cap \mathcal{S}, \chi)$. We use the shorthand $\mathrm{d}_\kappa(I, \chi)_{\hat{c}} \triangleq \mathrm{d}_\kappa(I, \chi, \mathcal{I})_{\hat{c}}$. We also define the density vector $\mathrm{d}_\kappa(I, \chi, \mathcal{S}) \in \mathbb{R}_+^{E_\mathfrak{c}}$ such that:

$$\mathrm{d}_\kappa(I, \chi, \mathcal{S}) \triangleq [\mathrm{d}_\kappa(I, \chi, \mathcal{S})_{\hat{c}}]_{\hat{c} \in E_\mathfrak{c}}$$

We also define *relative density* as follows:

Definition 24 (Relative density.)**.** *Fix $I \in \mathcal{I}$, κ, color χ and an interval set $\mathcal{S} \ni I$. The relative density $\mathrm{rd}_\kappa(I, \chi, \mathcal{S}) \in [1, \infty]^{E_\mathfrak{c}}$ is the density of I w.r.t. each cost $\hat{c} \in E_\mathfrak{c}$, divided by such density restricted to the set $\mathcal{S}$, i.e., $\mathrm{rd}_\kappa(I, \chi, \mathcal{S}) \triangleq \mathrm{d}_\kappa(I, \chi) \oslash \mathrm{d}_\kappa(I, \chi, \mathcal{S})$.*[8]

By convention, we set the relative density of empty colors to 1 (or ∞ if empty in $\mathcal{S}$ only). Note that $\mathrm{rd}_\kappa(I, \chi, \mathcal{S}) \geq 1$ and is monotonically decreasing in $\mathcal{S}$. Both are important properties that will be used when we bound density mass in "growing

[8] $\oslash$ denotes the Hadamard coordinate-wise division.

balls of intervals".

Our main matching algorithm uses estimates of densities of (intervals, color) pairs. In order to estimate the densities fast, we use standard sampling, as implemented by algorithms ApproxDensity and ApproxRelativeDensity (Alg. 3), whose guarantees are as follows.

Lemma 25 (Approximating Densities, proof in Section 3.6)**.** *Fix interval $I \in \mathcal{I}$, interval ball $S \subseteq \mathcal{I}$, color χ in coloring κ, and cost $c \in E_{\mathtt{c}}$. Then, recalling that $T_{\mathtt{ad}}$ is the runtime of a $\mathtt{ad}(\cdot,\cdot)$ oracle call:*

1. *For any given minimal density $\mathrm{d}_m > 0$, the algorithm* ApproxDensity *outputs $\widehat{\mathrm{d}} \in [\Omega(1), 1] \cdot \max\{\mathrm{d}_\kappa(I, \chi, \mathcal{S})_c, \mathrm{d}_m\}$ whp, in time $T_D = T_{\mathtt{ad}} \cdot \tilde{O}\left(\frac{\mu_\kappa(\mathcal{S},\chi)}{\mathrm{d}_m} + 1\right)$.*

2. *Assume $I \in \mathcal{S}$. For any given minimal relative density $\mathrm{rd}_m \geq 1$, the algorithm* ApproxRelativeDensity *outputs $\widehat{\mathrm{rd}} \in [\Omega(1), 1] \cdot \max\{\mathrm{rd}_\kappa(I, \chi, \mathcal{S})_c, \mathrm{rd}_m\}$ whp, in time*

$$T_{RD} = T_{\mathtt{ad}} \cdot \tilde{O}\left(\frac{1}{\mathrm{rd}_m} \cdot \frac{\mu_\kappa(\mathcal{I},\chi)}{\mu_\kappa(I,\chi)} + \frac{\mu_\kappa(\mathcal{S},\chi)}{\mu_\kappa(I,\chi)}\right).$$

Table 2: Summary of matching algorithm setup notations.

Notation	Description
Intervals	
$\mathcal{X}, \mathcal{X}_w$	space of all w-length intervals of x.
$\mathcal{Y}, \mathcal{Y}_w$	space of all w-length intervals of y.
$\mathcal{I}, \mathcal{I}_w$	space of all w-length intervals $= \mathcal{X} \cup \mathcal{Y}$.
Distances	
$(\mathcal{I}, \mathtt{ad}_w), (\mathcal{I}, \mathtt{ad}_{w,\mathfrak{c}})$	alignment-distances. $\mathtt{ad}_w$ align-approximates $\mathtt{ed}$. $\mathtt{ad}_{w,\mathfrak{c}}$ is a metric.
$(\mathbb{R}^d_+, \mathtt{dd}_F)$	distortion resilient distance for $F > 1$. $\mathtt{dd}_F(p,q) = \sum_{i:p_i > F \cdot q_i} p_i$
Costs, Neighborhood	
$\mathfrak{c}$	the current base cost for which we are building the current graph $G_{w,\mathfrak{c}}$
S_w	set of possible base costs $\{1/2n, \ldots 1/2, 1, 2, 4, \ldots w\}$
$E_\mathfrak{c}, c_i$	$c_i = \frac{\mathfrak{c}}{\alpha \cdot \epsilon} \cdot 3^i$ and $E_\mathfrak{c} = \{c_1, \ldots c_{1/\eta}\}$, for a small constant η, dependent on ϵ only
$\mathcal{N}_c(I)$	neighborhood of radius (cost) c: the set of all $J \in \mathcal{I}$ with $\mathtt{ad}(I,J) \leq c$
Colorings	
λ	$= n^\epsilon$
ν_t	$[\lambda^t] \cup \{\mathbf{u}, \perp\}$ color set
κ'	input coloring obtained from the step $t-1$
κ	current state of output coloring (at step t, level l)
$\hat{\kappa}$	new state of output coloring being constructed (for step t, level $l+1$)
$\mu_\kappa(I, \chi)$	mass of (I, χ) in κ; $\mu_\kappa(I, *)$ is a probability distribution
$\mathcal{P}^\chi_\kappa$	the support of $\mu_\kappa(*, \chi)$ (is a set of intervals).
Densities, Proximity Balls	
$\mathrm{d}_\kappa(I, \chi, \mathcal{S})_{\hat{c}}$	$= \mu_\kappa(\mathcal{N}_{\hat{c}}(I) \cap \mathcal{S}, \chi)$ is the $\hat{c}$-density of (I, χ) in interval set $\mathcal{S}$
$\mathrm{d}_\kappa(I, \chi, \mathcal{S})$	$= \mathrm{d}_\kappa(I, \chi, \mathcal{S})_*$ is a vector of densities, where $*$ ranges over all costs in $E_\mathfrak{c}$
$\mathrm{rd}_\kappa(I, \chi, \mathcal{S})$	$= \mathrm{d}_\kappa(I, \chi, \mathcal{I}) \oslash \mathrm{d}_\kappa(I, \chi, \mathcal{S})$ is a vector obtained by coordinate-wise division
$\Lambda^\zeta_\kappa(I)$	largest interval ball around I containing $\leq \zeta$ $\mathbf{u}$-colored ℓ_1-mass on each side of I

Algorithm 3 Matching Phase: Approximating Densities

Input: interval I, color χ, interval ball $\mathcal{S}$, cost c, additive parameter d_m / rd_m and access to μ_κ

Output: $\Theta(\max\{\mathrm{d}_\kappa(I,\chi,\mathcal{S})_c, \mathrm{d}_m\})$ and $\Theta(\max\{\mathrm{rd}_\kappa(I,\chi,\mathcal{S})_c, \mathrm{rd}_m\})$ respectively

1: **function** ApproxDensity$(I, \chi, \mathcal{S}, \mathrm{d}_m, c, \mu_\kappa)$
2: $\quad\gamma \leftarrow O\left(\frac{\log n}{\mathrm{d}_m}\right)$
3: $\quad\mathcal{S}' \leftarrow$ Sample each $I' \in \mathcal{S}$ independently with probability $\min\{\gamma \cdot \mu_\kappa(I', \chi), 1\}$.
4: $\quad\mathcal{S}^* \leftarrow \mathcal{N}_c(I) \cap \mathcal{S}'$ by computing $\mathtt{ad}(I, I')$ for all $I' \in \mathcal{S}'$ and keeping all c-matches.
5: $\quad\widetilde{\mathcal{S}^*} \leftarrow \{I' \in \mathcal{S}^* \mid \gamma \cdot \mu_\kappa(I', \chi) > 1\}$.
6: $\quad$**return** $\Theta\left(\max\left\{\mu_\kappa(\widetilde{\mathcal{S}^*}, \chi) + \frac{1}{\gamma} \cdot |\mathcal{S}^* \setminus \widetilde{\mathcal{S}^*}|, \mathrm{d}_m\right\}\right)$. $\quad\triangleright\ \Theta \leftarrow$ rescaling to get $[\Omega(1), 1]$-approx.
7: **end function**
8: **function** ApproxRelativeDensity$(I, \chi, \mathcal{S}, \mathrm{rd}_m, c, \mu_\kappa)$
9: $\quad\hat{d} \leftarrow$ ApproxDensity$(I, \mathcal{S}, \mu_\kappa(I, \chi), c, \mu_\kappa)$.
10: $\quad\hat{D} \leftarrow$ ApproxDensity$(I, \mathcal{I}, \mathrm{rd}_m \cdot \mu_\kappa(I, \chi), c, \mu_\kappa)$.
11: $\quad$**return** $\Theta\left(\max\left\{\frac{\hat{D}}{\hat{d}}, \mathrm{rd}_m\right\}\right)$. $\quad\triangleright\ \Theta \leftarrow$ rescaling to get $[\Omega(1), 1]$-approx.
12: **end function**

Soft transformations. We define a couple of "soft" transformations used by the algorithm: soft thresholding, and soft quantile. Their purpose is to replace "hard" thresholds, thus balancing complexity vs correctness.

The *soft thresholding* transformation helps us with preserving sparsity of palettes. For $\delta \in (0, 1]$ and $\gamma > 0$, define $T^q_{\delta,\gamma} : \mathbb{R}^d_+ \to \mathbb{R}^d_+$ to be the transformation:

$$T^q_{\delta,\gamma}(x)_i \triangleq \begin{cases} x_i & x_i \geq \gamma \\ 0 & x_i < \delta\gamma \\ \gamma \cdot (\frac{x_i}{\gamma})^q & \text{otherwise.} \end{cases}$$

The basic intuition is that $T^q_{\delta,\gamma}$ softens the threshold at about γ to decay (polynomially) between γ and $\delta\gamma$, when it becomes 0. Our algorithms use a few thresholding transformations with different parameters.

Also, define the *soft quantile* transformation $Q_{\delta,s,F} : \mathbb{R}^d_+ \to \mathbb{R}$, for parameters $\delta, s \in [0,1]$, and $F \geq 1$:

$$Q_{\delta,s,F}(x) \triangleq \max_{J \subseteq [d]} a_{|J|} \cdot \min_{j \in J} x_j; \qquad \text{where } a_i = \begin{cases} 1 & i \geq s \cdot d \\ 0 & i < (s-\delta)d \\ 1/F^{sd-i} & \text{otherwise.} \end{cases}$$

We use one such transformation: $Q_l(x) \triangleq Q_{\frac{\epsilon}{4}, 1-\frac{(l+1)\epsilon}{2}, n^{O(\alpha^2)}}(x)$. The intuition is that Q_l is a smoothing between $\frac{(l+1)\epsilon}{2}$ fractional rank element from x (in sorted order), to $\frac{(l+1)\epsilon}{2} + \epsilon/4$ fractional rank.

We discuss and prove the properties of these soft transformations in Section 3.5.3.

3.4.2 Components of MatchStep: main coloring procedure

To amend the coloring κ, we compute a set of potential scores, ϕ, φ, and θ. In particular, we sample *anchor* intervals to define ϕ potential scores. Then, such scores are divided over Λ-balls centered at the anchors and `ad`-close intervals, to generate φ palettes to all other intervals. We also sample *pivots* to define θ potential scores, and use it to augment φ. Lastly, we use φ palettes to amend coloring $\kappa \to \widehat{\kappa}$ for the next level. We describe this procedure in detail next.

Steps and levels. In each step t, for a given input coloring κ', we produce $O(\log n)$ output colorings. The goal for step t is that for each interval $I \in \mathcal{I}$, either: (1) we cluster it together with $\pi[I]$ and mark them as "already matched", or (2) color I similarly to $\pi[I]$ up to some *bounded distortion* (while also ensuring the color parts are decreasing with t). Here, our goal is to efficiently assign colors to intervals in $\mathcal{I}$ (which can be thought of as overlapping parts in a partition), so that we can compare sampled anchors to some limited number of other intervals that share the same color. Overall, as t increases, the number of colors in the coloring grows and the size of each color (number of intervals of that color) becomes smaller, allowing us to increase the number of sampled anchors.

We maintain the following set of *coloring properties* for a coloring κ at a step t, which uses the color-set $\nu_t = [\lambda^t] \cup \{\mathbf{u}, \perp\}$, analyzed and proved in later sections:

- *Sparsity:* Each color $\chi \in \nu \setminus \{\perp\}$ will be non-zero for (i.e., shared by) few intervals: $|\mathcal{P}_\kappa^\chi| = O(n^{1+O(\epsilon)}/\lambda^t)$. See Lemma 30.

- *Correctness:* Coloring κ will have *bounded corruption* (as per Eqn. (3.1)), meaning "accumulated error" is bounded by $O_\epsilon(1)$ times the "original error", $O_\epsilon(\mathtt{ed}(x, y))$. See Lemma 27.

At each step, our algorithm iterates over *levels* $l \in \{0, \ldots, \log_\beta n\}$, where $\beta = n^\epsilon$. Intuitively, each level takes care of different *density regimes* of pairs: lower levels will correspond to *high-density pairs* and high levels to *low-density pairs*. The lowest level $l = 0$ is special: here our goal will be to cluster and match high density pairs, and mark them as *already-matched*. In the subsequent levels, our goal is to color balls of intervals which have large mass of pairs of the corresponding densities.

Anchor Sampling. In each level $l \geq 0$, for each output color $\chi \in \nu_t \setminus \{\mathbf{u}, \perp\}$, we sample an *anchor*, which is an interval, color pair $(A, \chi') \in \mathcal{I} \times \nu_{t-1} \setminus \{\perp\}$, from the distribution $\frac{\mu_{\kappa'}(*,*)}{|\mathcal{I}|}$ (if sampled $\chi' = \perp$, we skip this anchor/χ). Each such anchor is compared to all other intervals in $\mathcal{P}_{\kappa'}^{\chi'}$ to form a family of $O_\epsilon(1)$ clusters $\mathcal{C}_\chi$. In level $l = 0$, we add a graph-G edge from A to each clustered interval, and mark such intervals as *already-matched* with $\perp$-color. In the other levels $l \geq 1$, such clusters are extended to other intervals in the clusters' *proximity*, adding χ to their palettes φ, as described in detail below.

Clustering. For each sampled pair (A, χ'), we compute the `ad` distance between A and all other intervals $I \in \mathcal{P}_{\kappa'}^{\chi'}$, using the `ad` oracle. We also sample a cost $\hat{c} \sim E_{\mathfrak{c}}$ uniformly at random. Next consider subsets $\mathcal{A}_j \subseteq \mathcal{P}_{\kappa'}^{\chi'}$ defined as $\mathcal{A}_j = \mathcal{N}_{\hat{c}+\mathfrak{c}\cdot j}(A) \cap \mathcal{P}_{\kappa'}^{\chi'}$, for $j \in \{0, \ldots, j_{\max}\}$ where $j_{\max} = O(1/\alpha)$. While the use of such sampling process will be shown later, the important *clustering property* to note here is that for $j < j_{\max}$, we have $\mathcal{N}_{\mathfrak{c}}(\mathcal{A}_j) \cap \mathcal{P}_{\kappa'}^{\chi'} \subseteq \mathcal{A}_{j+1}$. The formal clustering algorithm ClusterAnchor is presented in Alg. 4.

Algorithm 4 Matching Phase: Clustering around random anchors

Input: Anchor pair (A, χ'), interval set $\mathcal{R}$, base cost $\mathfrak{c}$

Output: "slowly-growing" clusters around A: $\{(\mathcal{A}_{\hat{c}+\mathfrak{c}j}, \hat{c}+\mathfrak{c}j, j, d_{\mathcal{A}})\}_{j=0\ldots j_{\max}}$ for a random cost $\hat{c} \sim E_{\mathfrak{c}}$, where $d_{\mathcal{A}}$ is an upper-bound for $\{d_{\kappa'}(I,\chi')_{\hat{c}+\mathfrak{c}j} \mid j \in \{0,\ldots j_{\max}\}, I \in \mathcal{A}_{\hat{c}+\mathfrak{c}j}\}$

1: **function** ClusterAnchor($A, \mathcal{R}, \chi', \mathfrak{c}$)
2: Sample $\hat{c} \sim E_{\mathfrak{c}}$ uniformly at random.
3: For all $I \in \mathcal{R}$, compute $c_{A,I} \triangleq \mathtt{ad}(A, I)$.
4: Below, for a parameter τ, we use notation $\mathcal{A}_\tau = \{I : c_{A,I} \leq \tau\}$.
5: **return** $\{(\mathcal{A}_{\hat{c}+\mathfrak{c}j}, \hat{c}+\mathfrak{c}j, j, \mu_{\kappa'}(\mathcal{A}_{3\hat{c}}, \chi')) \mid j = 0, 1, \ldots, j_{\max}\}$.
6: **end function**

Coloring: assigning ϕ potential to clusters. Next we assign potentials ϕ to the clustered intervals in $\mathcal{C}_\chi$. Later, using potentials ϕ, we will assign φ potential to other nearby intervals in the proximity of the clustered intervals (as described in Section 3.2.3).

From the above clustering algorithm, for a fixed output color χ, and corresponding, sampled $A, \chi', \hat{c}$, we get a family of clusters $\mathcal{C}_\chi \triangleq \{(\mathcal{A}_j, c_{\mathcal{A}_j}, j, d_{\mathcal{A}})\}_j$ satisfying the clustering invariant from above. We then define potential $\phi_\kappa(I, \chi)$ in new coloring κ for all intervals $I \in \cup_j \mathcal{A}_j$, using Alg. 5.

Intuitively, for sampled $(A, \chi', \hat{c})$, we would like to distribute $O(n/\lambda^t)$ potential credits "equally" among $I \in \mathcal{P}^{\chi'}_{\kappa'} \cap \mathcal{N}_{\hat{c}}(A)$, namely $\phi_\kappa(I,\chi) = \frac{2n}{\lambda^t} \cdot \frac{\mu_{\kappa'}(I,\chi')}{\mathrm{d}_{\kappa'}(A,\chi')_{\hat{c}}}$ (note that this sums up to $2n/\lambda^t$ over all I, and to $2n$ over all anchors/χ's). This method however does not satisfy the necessary ϕ properties, requiring some adjustments. Before describing the adjustments, we state these necessary properties, termed ϕ *scoring invariants*, which we will guarantee:

Claim 26 (ϕ Scoring Invariants, proved in Section 3.6)**.** *Fix step t and level l. Fix a color χ in an output coloring κ, for which we have sampled an anchor pair $(A, \chi'') \in \mathcal{I} \times \nu_{t-1} \setminus \{\perp\}$ where χ'' is some color in κ', and sampled cost $\hat{c} \in E_{\mathfrak{c}}$. The ϕ scores from Alg. 5 satisfy the following invariants, conditioned on an event that holds whp:*

1. Correctness: *For any $I \in \mathcal{I}$ and distortion $F \geq 1$, if (I, χ'') is* not an *F-*corrupted pair *(for some fixed alignment π), then $\mathtt{dd}_{O(F \cdot n^{2\alpha})}(\phi_\kappa(I, \chi), \phi_\kappa(\pi[I], \chi)) \leq n^{-10}$.*

2. Maximal Contribution*: For all $(I, \chi') \in \mathcal{I} \times \nu_{t-1} \setminus \{\perp\}$, the expected potential contribution of (I, χ') to $\phi_\kappa(I, \chi)$ satisfies:*

$$\mathbb{E}_{A,\chi'',\hat{c}}\left[\phi_\kappa(I, \chi) \cdot \mathbf{1}[\chi' = \chi'']\right] = O\left(\tfrac{\mu_{\kappa'}(I,\chi')}{\lambda^t}\right).$$

3. Minimal Contribution/Balance*: For all $(I, \chi') \in \mathcal{I} \times \nu_{t-1} \setminus \{\perp\}$, the expected potential contribution of (I, χ') to $\phi_\kappa(I, \chi)$ at least equals its mass $\mu_{\kappa'}(I, \chi')$ on almost all costs in $E_{\mathfrak{c}}$: in particular, for all but $O(\eta/\alpha)$ fraction of costs $\hat{c} \in E_{\mathfrak{c}}$:*

$$\mathbb{E}_{A,\chi''}\left[\phi_\kappa(I, \chi) \cdot \mathbf{1}[\chi' = \chi'']\right] \geq \tfrac{\mu_{\kappa'}(I,\chi')}{\lambda^t}.$$

The first invariant ensures uncorrupted pairs (A, χ') add uncorrupted ϕ potential, in particular that I and $\pi[I]$ get similar potential ϕ. To guarantee the invariant, we use the weaker transitivity property, namely the clustering property that $\mathcal{N}_{\mathfrak{c}}(\mathcal{A}_j) \cap \mathcal{P}_{\kappa'}^{\chi'} \subseteq \mathcal{A}_{j+1}$, and hence that for uncorrupted pairs, $I \in \mathcal{A}_j \Rightarrow \pi[I] \in \mathcal{A}_{j+1}$ (this is the reason we have multiple clusters $\mathcal{A}_j$ to start with). In addition, we estimate densities with a lower-bound threshold of a factor $1/n^{O(\alpha)}$ from some upper-bound density of any interval in the cluster, so that all density estimates are within a $n^{O(\alpha)}$ bound

(this treshold also helps maintaining efficiency constraints). An additional caveat is that we cannot use this argument for $j = j_{\max}$. To fix this, our algorithm multiplies the potential of each $\mathcal{A}_j$ by an exponentially decreasing coefficient $\gamma_j = n^{-j\alpha}$, which ensures that each meaningful potential added to I generates a similar potential in $\pi[I]$.

The second invariant ensures that we do not assign too much mass to any pair (notably a corrupted one). The naïve assignment of ϕ would jeopardize this invariant because the corrupted intervals may be "ad-centers", i.e., slightly denser than their neighbors (for every cost), and hence receive more ϕ in expectation. To overcome this issue, we estimate the density $\mathrm{d}_{\kappa'}(I, \chi')_{c_{\mathcal{A}}}$ of each $I \in \mathcal{A}$, denoted $\widehat{d_{I,\chi'}}$, and add potential proportional to $1/\widehat{d_{I,\chi'}}$ (instead of $1/\mathrm{d}_{\kappa'}(A, \chi')_{\hat{c}}$).

The third invariant guarantees we assign enough potential overall at each step and its importance will become clear later once we discuss the balance of colors.

The formal algorithm AssignPhiPotential appears in Alg. 5.

Algorithm 5 Matching Phase: assign potential to Intervals in $\mathcal{X}, \mathcal{Y}$

Input: Output color χ, cluster $(\mathcal{A}, \chi')$, density bound $d_{\mathcal{A}}$, cost c, and a "decaying" parameter j.

Output: Assign ϕ potential to all $I \in \mathcal{A}$ for color χ.

function AssignPhiPotential($\chi, \mathcal{A}, d_{\mathcal{A}}, \chi', c, j, \mu_{\kappa'}$)
  $\gamma \leftarrow n^{-\alpha \cdot j}$.
  for $I \in \mathcal{A}$ **do**
    $\widehat{d_{I,j}} \leftarrow$ ApproxDensity($I, \chi', \mathcal{I}, d_{\mathcal{A}} \cdot n^{-\alpha}, c, \mu_{\kappa'}$).
    $\phi_\kappa(I, \chi) \leftarrow \phi_\kappa(I, \chi) + \gamma \cdot \frac{2n}{\lambda^t} \cdot \frac{\mu_{\kappa'}(I, \chi')}{\widehat{d_{I,j}}}$.
  end for
end function

Coloring: assigning φ scores to intervals. Given ϕ potentials, we assign φ scores to other non-clustered intervals in the proximity of the clustered ones. We will first provide a mechanism for level $l > 0$, and deal with the special case of $l = 0$ later on. Intuitively, we would like to assign potential φ to each interval within the ball (in index distance) of fixed radius centered at any clustered interval. We use multiple radiuses since any radius for interval I might need to be matched with a slightly different radius of $\pi[I]$. In addition, sometimes we need to group together far sections, and hence we define ball radiuses with respect to fixed $\mathbf{u}$-*color* mass (in the current coloring κ).

Specifically, let $Z_l = \beta^l \cdot \{1, 2, 4, 8, \ldots, n/\beta^l\}$ be an exponentially growing set of radiuses. Then for each $\zeta \in Z_l$, we define potential score vector $\varphi_\kappa^\zeta(\cdot,\cdot) : \mathcal{I} \times [\lambda^t] \cup \{\mathbf{u}\} \to [0,1]$ as follows. For $\chi \in [\lambda^t]$, we define:

$$\varphi_\kappa^\zeta(I,\chi) = T^{\mathbf{c}}\left(\mu_\kappa(I,\mathbf{u}) \cdot \tfrac{\beta^{2l}}{\zeta^2} \cdot \tfrac{\phi_\kappa(\Lambda_\kappa^\zeta(I),\chi)}{\beta^l}\right) \tag{3.3}$$

where $T^{\mathbf{c}} = T^{\Theta(\epsilon^{-1})}_{\beta^{-1},\beta^{-3}}$. We define $\varphi(\cdot,\mathbf{u})$ later (using potentials θ).

Intuitively, the $\frac{\beta^{2l}}{\zeta^2}$ factor decays the overall contribution fast enough to have a constant factor corruption overall when summing over all radiuses, but slow enough so that the distortion doesn't grow too much between similar size radiuses.

We will guarantee the following φ *scoring properties* for all $I \in \mathcal{I}$ except for intervals whose proximity is sufficiently corrupted (and hence do not need guarantees):

- *Correctness:* for any $\zeta \in Z_l$, the contribution of ϕ-potential from F-uncorrupted pairs to $\varphi_\kappa^\zeta(I, *_{\neq\mathbf{u}})$ matches, up to $F^{O(1/\epsilon)}$ factor distortion, the contribution to $\varphi_\kappa^{\zeta'}(\pi[I], *_{\neq\mathbf{u}})$, for slightly larger ζ', up to an extra additive error $O(n^{-9})$. See

Lemma 31.

- *Complexity:* For any color χ, the number of $I \in \mathcal{I}$ with $\varphi_\kappa^\zeta(I,\chi) \neq 0$ is $\frac{n\beta^{O(1)}}{\lambda^t}$. This will follow from the fact that thresholding $T^{\mathbf{c}}$ ensures that $\varphi_\kappa^\zeta(I,\chi) = 0$ whenever $\mu_\kappa(I,\mathbf{u}) \cdot \phi_\kappa(\Lambda_\kappa^\zeta(I),\chi) < \zeta^2/\beta^{l+O(1)}$. Implicitly in Lemma 30.

Coloring: assigning the u-colored $\varphi(\cdot,\mathbf{u})$. We recall that the $\mathbf{u}$-color is used to 1) efficiently partition intervals of any density together with their corresponding matches in π, and 2) "group" together sparse sections that might be far apart.

We assign $\varphi(\cdot,\mathbf{u})$ in a slightly different manner than $\varphi(\cdot,*_{\neq\mathbf{u}})$. First, we sample a number of random pairs $\mathcal{V}$ termed *pivots*, directly estimate their densities, and assign sparsity θ scores. We then use θ scores to assign $\varphi(\cdot,\mathbf{u})$ scores to nearby intervals in a similar manner to how we used ϕ to assign $\varphi(\cdot,*_{\neq\mathbf{u}})$. Our $\mathbf{u}$-coloring procedure will have the following $\mathbf{u}$-*coloring guarantees*:

- *Correctness:* The distortion between uncorrupted $(I,\mathbf{u})$ and $(\pi[I],\mathbf{u})$ is bounded (as for the non-$\mathbf{u}$ colors). See Lemma 31.

- *Sparsity:* At level l, for any interval $I \in \mathcal{I}$, for $\delta = \varphi_\kappa^{\beta^l}(I,\mathbf{u})$, there is a set of pairs $\mathcal{D}$ in the proximity of $\Lambda_\kappa^{\beta^l}(I)$ with $\mu_{\kappa'}(\mathcal{D}) = \Omega(\delta\beta^l)$ such that all pairs in $\mathcal{D}$ have relative density (w.r.t. to the Λ-ball) $\leq n^{O(\alpha)}\beta\frac{n}{\beta^l\lambda^t}$ on at least $\approx 1 - \frac{(l+1)\epsilon}{2}$ fraction of costs $\in E_{\mathbf{c}}$. The exact property will be described in Claims 46 and 47.

- *Balance:* for every interval $I \in \mathcal{I}$, we will have $\|\mu_\kappa(I,*_{\neq\mathbf{u}}) + \varphi_\kappa(I,*)\|_1 = \Omega_\epsilon(1)$. This ensures that we can re-normalize the coloring at level l with only $O_\epsilon(1)$-factor corruption blow-up. See Lemma 32.

The high-level idea is as follows. Consider an interval I. If there is a set of pairs (J, χ') in I's proximity of total mass $\delta\beta^l$, where for each such pair the relative density $\mathrm{rd}_{\kappa'}(J, \chi', \Lambda_\kappa^{\beta^l}(J))_c$ is at most $\Theta^*(\frac{n}{\lambda^t \cdot \beta^l})$, for sufficiently many costs $c \in E_{\mathfrak{c}}$, then we have the "sparsity guarantee" we need for keeping I **u**-colored for the next level with δ mass. However, if δ is small enough, then we expect to find sufficiently many intervals of the "right" density to color I with non-**u** colors to obtain the "balance" guarantee as above.

We assign $\varphi(\cdot, \mathbf{u})$ in three stages. First, we randomly sample a multi-set $\mathcal{V}$ consisting of $O^*(\frac{n}{\beta^l})$ pivot pairs by including a pair $(I, \chi') \in \mathcal{I} \times \nu \setminus \{\perp\}$ with probability $\mu_{\kappa'}(I, \chi') \cdot \beta^{-l}$, independently, $k = \beta^{O(1)}$ times (i.e., a sample (I, χ') can have multiplicity up to k). Second, for each pivot $(V, \chi') \in \mathcal{V}$, for each possible radius $\zeta \in Z_l$, we generate *potential sparsity score* $\theta_l^\zeta(V, \chi')$, which can be thought of as "the mass of colors which V is relatively sparse on, for many costs". To obtain that, we iterate over all costs $\hat{c} \in E_{\mathfrak{c}}$ and estimate an upper bound on $\widehat{\mathrm{rd}_{\kappa'}(V, \chi', \Lambda_\kappa^\zeta(V))_{\hat{c}}}$, using the approximation algorithm ApproxRelativeDensity.

To maintain near-linear runtime overall, we can afford at most $O^*(\beta^l)$ time for ApproxRelativeDensity per pivot (on average), and hence we set the "min threshold" rd_m parameter to $\Theta^*(\frac{n}{\lambda^t \cdot \beta^l})$. Also for the runtime bound, we would need that the local color-mass $\mu_{\kappa'}(\Lambda_\kappa^\zeta(V), \chi')$ is at most $O^*(\beta^l) \cdot \mu_{\kappa'}(V, \chi')$. When the latter condition doesn't hold, we do not need to do any testing as the relative density will be lower than the bound we care about on average across all potential local (V, χ') pairs.

We use the estimate of relative density to generate $\sigma_{V,\chi'} \in [0, 1]$ based on the number of costs with sparse relative density, using the soft transformations:

$$\sigma_{V,\chi'} = Q_l(T^{\theta}(\Gamma_{l,\zeta}(V,\chi'))),$$

where $\Gamma_{l,\zeta}(V,\chi')$ is a vector of dimension $|E_{\mathfrak{c}}|$ with

$$\Gamma_{l,\zeta}(V,\chi')_c = \min\left\{\widehat{\mathrm{rd}_{\kappa'}(V,\chi',\Lambda^{\zeta}_{\kappa}(V))_c}^{-1} \cdot \frac{n \cdot n^{4\alpha}}{\beta^l \cdot \lambda^t},\ 1\right\} \quad \text{and} \quad T^{\theta} = T^{O(1/\epsilon)}_{1/\sqrt{\beta},1}.$$

Then we define the *sparsity potential score* $\theta^{\zeta}_{\kappa}(V,\chi')$ for sampled pivots $V \in \mathcal{V}$. As mentioned above, for pivot pairs where we cannot efficiently estimate $\sigma_{V,\chi'}$, we set the score to 1. In particular, for $m_{\mathcal{V}}(V,\chi')$ denoting the multiplicity of (V,χ') in $\mathcal{V}$:

$$\theta^{\zeta}_{\kappa}(V,\chi') = m_{\mathcal{V}}(V,\chi') \cdot \begin{cases} \sigma_{V,\chi'} & \mu_{\kappa'}(\Lambda^{\zeta}_{\kappa'}(V),\chi') \le \lambda \cdot \beta^6 \cdot \beta^l \\ 1 & \text{otherwise.} \end{cases}$$

The algorithm for computing the θ potential, AssignThetaPotential, is presented in Alg. 6.

Finally, we assign $\varphi_{\kappa}(I,\mathbf{u})$ as a function of the estimated sparsity potential from $\Lambda^{\zeta}_{\kappa}(I)$, using the following formula, for each $\zeta \in Z_l$:

$$\varphi^{\zeta}_{\kappa}(I,\mathbf{u}) = T^{\mathbf{u}}\left(\frac{\beta^{2l}}{\zeta^2} \cdot \mu_{\kappa}(I,\mathbf{u}) \cdot \min\left\{\frac{1}{k} \cdot \sum_{\chi' \in \nu \setminus \{\perp\}} \theta^{\zeta}_{\kappa}(\Lambda^{\zeta}_{\kappa}(I),\chi'), 1\right\}\right), \tag{3.4}$$

where $T^{\mathbf{u}} = T^{1/\epsilon}_{1/\beta,\Omega_{\epsilon}(1)}$, and $k = \beta^{O(1)}$ is a coefficient which guarantees concentration for all *potential scores* not omitted by the transformation $T^{\mathbf{u}}$ as above.

Algorithm 6 Matching Phase: assign θ potential to pivot pair V, χ'.

Input: Pivot (V, χ'), multiplicity m, and level l

Output: Assign θ potential to (V, χ')

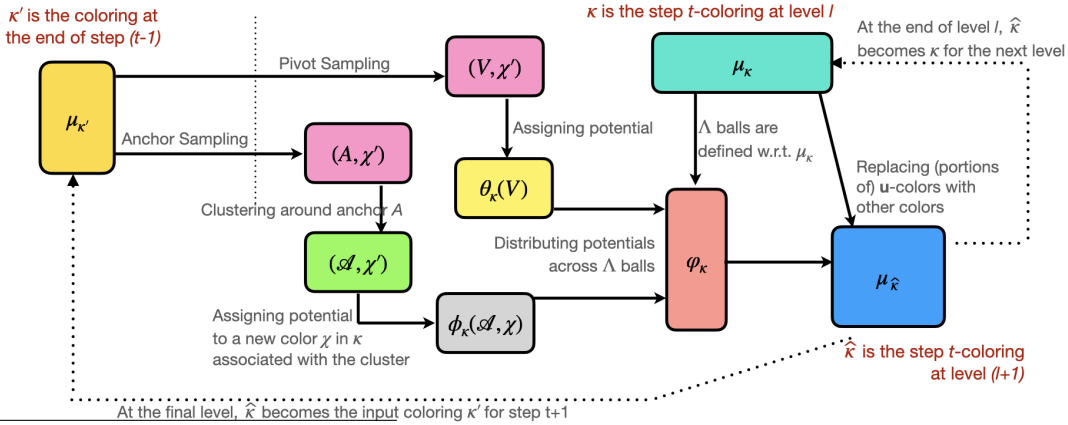
function AssignThetaPotential($V, \chi', m, l, \mu_{\kappa'}$)
  $\mathrm{rd}_m \leftarrow n^{4\alpha} \cdot \frac{n}{\beta^l \cdot \lambda^t}$.
  for $\zeta \in Z_l$ **do**
    if $\mu_{\kappa'}(\Lambda_\kappa^\zeta(V), \chi') > \lambda \cdot \beta^6 \cdot \beta^l$ **then**
      $\theta_\kappa^\zeta(V, \chi') \leftarrow m$.
    else
      for $\hat{c} \in E_\mathfrak{c}$ **do**
        $\widehat{\mathrm{rd}_{\kappa'}(V, \chi', \Lambda_\kappa^\zeta(V))_{\hat{c}}} \leftarrow$ ApproxRelativeDensity($V, \Lambda_\kappa^\zeta(V), \mathrm{rd}_m, \hat{c}, \mu_{\kappa'}$).
        $\Gamma_{l,\zeta}(V, \chi')_{\hat{c}} \leftarrow \min\left\{\mathrm{rd}_m / \widehat{\mathrm{rd}_{\kappa'}(V, \chi', \Lambda_\kappa^\zeta(V))_{\hat{c}}}, 1\right\}$.
      end for
      $\theta_\kappa^\zeta(V, \chi') \leftarrow m \cdot Q_l(T^\theta(\Gamma_{l,\zeta}(V, \chi')))$.
    end if
  end for
end function

Amending the measure $\mu_\kappa(I, *)$ to get $\mu_{\hat{\kappa}}(I, *)$. At the end of each level l, we update each measure $\mu_\kappa(I, *)$ by moving some of the $\mu_\kappa(I, \mathbf{u})$ mass according to the $\varphi(I, *)$ potential. We need to ensure that we still obtain a distribution at the end, and hence we do a certain normalization (rescaling) on φ. Since such a rescaling can increase the corruption, we need to ensure that the renormalization rescales the vector by a constant factor only. There is a caveat though, that the renomalization factor is small only when the added $\varphi(I, *)$ potential is large, which we can only guarantee when $\mu(I, *_{\neq \mathbf{u}})$ is small. As a result, we recolor using the sum of μ and φ according

to the following formula[9]:

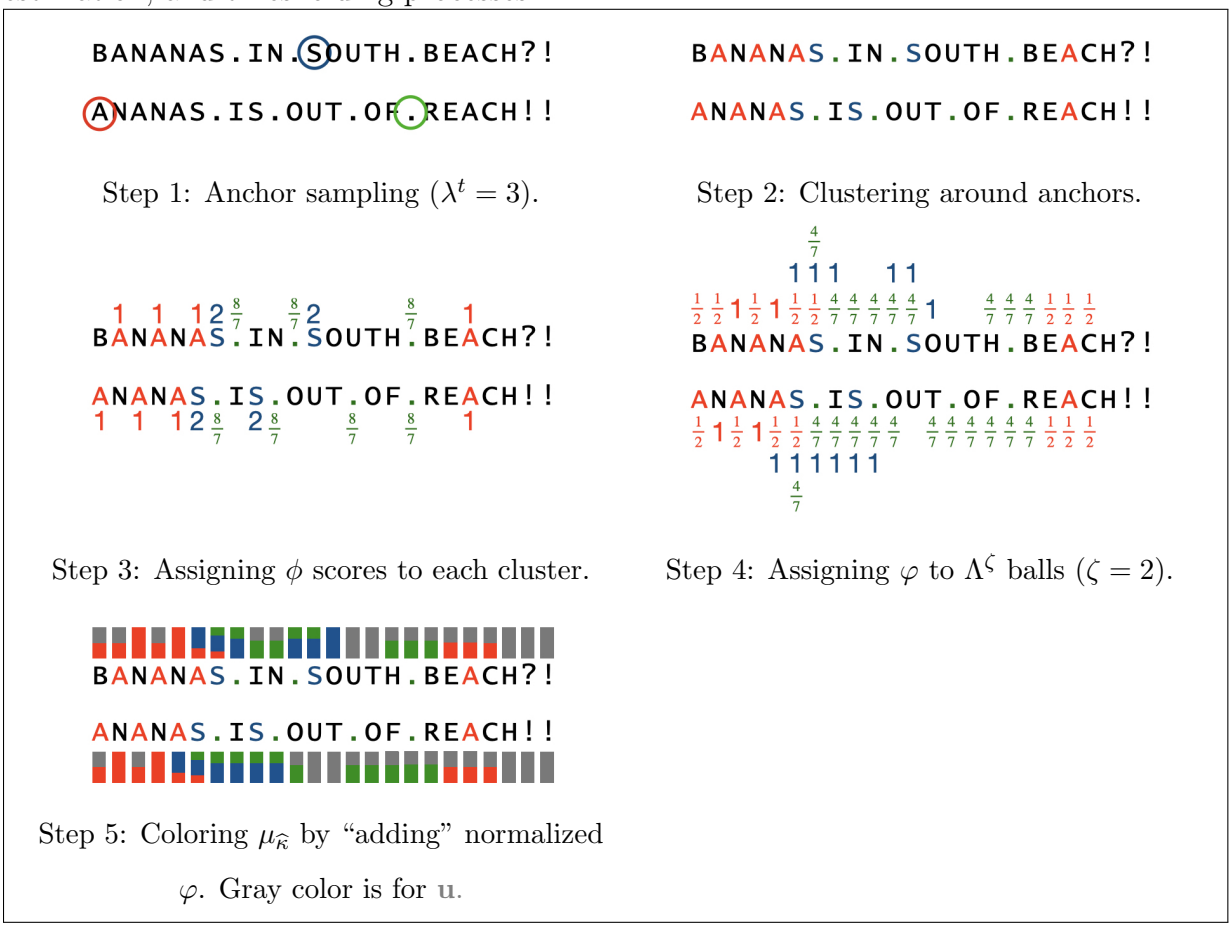
$$\mu_{\widehat{\kappa}}(I,*) = \mu_{\kappa}(I,*_{\neq \mathbf{u}}) + \mu_{\kappa}(I,\mathbf{u}) \cdot \frac{\mu_{\kappa}(I,*_{\neq \mathbf{u}}) + \sum_{\zeta \in Z_l} \varphi_{\kappa}^{\zeta}(I,*)}{\|\mu_{\kappa}(I,*_{\neq \mathbf{u}}) + \sum_{\zeta \in Z_l} \varphi_{\kappa}^{\zeta}(I,*)\|_1}.$$

Figure 2: The flow of coloring intervals at level l



[9]Note that in the formula, the vector $\mu_{\kappa}(I,*_{\neq \mathbf{u}})$ is considered to be the vector with $\mathbf{u}$-coordinate zero-ed out.

Figure 3: A step by step coloring example at level l. For simplification, all intervals start as **u**-colored, we use one $\zeta = 2$ radius, and omit the pivot sampling, density estimation, and thresholding processes.



Initial level ($l = 0$): marking as "already matched" $\perp$. For each step t, at level $l = 0$, we assign ϕ and θ potential as above and use them to mark intervals as "already matched" (with color $\perp$), instead of "regular coloring". Hence for any step t, we only have the colors **u** and $\perp$ in κ at the end of level $l = 0$, and the rest of colors come into play later, starting with level $l = 1$. We use the following potentials, where $Z_0 = \{0\}$:

$$\varphi^0_\kappa(I,\perp) = T^{\mathbf{c}}\left(\mu_\kappa(I,\mathbf{u})\cdot\phi_\kappa(I,\nu\setminus\{\mathbf{u},\perp\})\right). \tag{3.5}$$

$$\varphi^0_\kappa(I,\mathbf{u}) = T^{\mathbf{u}}\left(\mu_\kappa(I,\mathbf{u})\cdot\tfrac{\theta^0_\kappa(I,\nu\setminus\{\perp\})}{k}\right). \tag{3.6}$$

Overall, the idea here is similar to the general case: any matching from uncorrupted (I,χ') pairs in κ' will generate a 2-hop path in G between I and $\pi[I]$, and hence by projecting all of ϕ potential to $\perp$, we do not over-corrupt interval I (in expectation), and maintain the $\mathbf{u}$-*coloring guarantees*. We also note that the $T^{\mathbf{c}}$ transformation at level 0 is not required for correctness, but rather enables a more uniform analysis across levels.

The overall algorithm for computing amended coloring, AMENDCOLORING, is presented in Alg. 7.

Algorithm 7 Matching Phase: Color Intervals

Input: Coloring κ and level l

Output: Amended coloring $\widehat{\kappa}$ computed from μ_κ,ϕ_κ and θ_κ.

1: **function** AMENDCOLORING(κ, l)

2: Compute the balls $\Lambda^\zeta_\kappa(I)$ for all $I\in\mathcal{I}$ and $\zeta\in Z_l$ (using algorithm from Thm. 50).

3: Compute $\varphi^\zeta_\kappa(I,\mathbf{u})$ as per Eqn. (3.4),(3.6) for all $I\in\mathcal{I},\zeta\in Z_l$ (using data structure from Thm. 50).

4: Compute $\varphi^\zeta_\kappa(I,*_{\neq\mathbf{u}})$ as per Eqn. (3.3),(3.5) for all $I\in\mathcal{I},\zeta\in Z_l$ (using algorithm from Thm. 51).

5: Compute $\mu_{\widehat{\kappa}}(I) = \mu_\kappa(I,*_{\neq\mathbf{u}}) + \mu_\kappa(I,\mathbf{u})\cdot\frac{\mu_\kappa(I,*_{\neq\mathbf{u}})+\sum_{\zeta\in Z_l}\varphi^\zeta_\kappa(I,*)}{\left\|\mu_\kappa(I,*_{\neq\mathbf{u}})+\sum_{\zeta\in Z_l}\varphi^\zeta_\kappa(I,*)\right\|_1}$ for all $I\in\mathcal{I}$.

6: **return** $\widehat{\kappa}$.

7: **end function**

3.4.3 MatchStep algorithm: main levels loop

Finally, we describe the overall MatchStep algorithm, using the ingredients presented earlier. The main algorithm is MatchStep from Alg. 8. It uses couple more functions: InitColoring, in Alg. 9, and Adjust-**u** in Alg. 10.

Choice of Parameters. We fix the following parameters, as a function of n and ϵ.

- $\beta, \lambda \leftarrow n^{\epsilon}$, ensuring convergence in constant number of rounds while allowing *sparse* partitions.
- $k \leftarrow \beta^3 (\log n)^{\Theta(1)}$, an oversampling factor for concentration in pivot sampling.
- $\alpha \leftarrow \epsilon^{5/\epsilon}$, sufficiently small constant to control the blow-up of the distortion F (noting that the starting distortion is $F_0 = n^{\alpha}$).
- $\eta \leftarrow \alpha^2 \epsilon^3$, to ensure our set of costs is sufficiently large, avoiding blow-up from Q transformations.

Algorithm 8 Matching Step Algorithm

Input: base cost $\mathfrak{c}$, input coloring κ', step t.

Output: a matching graph G and an output colorings κ.

function MATCHSTEP($\mathfrak{c}, \kappa', t$)
  Initialize $\kappa \leftarrow$ INITCOLORING(κ', t).
  $G \leftarrow$ unweighted and undirected graph with nodes $I \in \mathcal{I}$ and no edges.
  for $l = 0, 1, \ldots$ **do**
    Break if $l > 0$ and $\min\{\mu_\kappa(\mathcal{X}, \mathbf{u}), \mu_\kappa(\mathcal{Y}, \mathbf{u})\} \le \beta^l$;
    $\phi_\kappa, \theta_\kappa \leftarrow \vec{0}$.
    for $\chi \in [\lambda^t]$ **do**
      Sample (A, χ') from the distribution $\frac{\mu_{\kappa'}(*,*)}{|\mathcal{I}|}$.
      $\mathcal{C}_\chi \leftarrow \begin{cases} \text{CLUSTERANCHOR}(A, \mathcal{P}^{\chi'}_{\kappa'}, \chi', \mathfrak{c}), & \text{if } \chi' \neq \perp \\ \emptyset, & \text{if } \chi' = \perp \end{cases}$
      for $(\mathcal{A}, c_\mathcal{A}, j, d_\mathcal{A}) \in \mathcal{C}_\chi$ **do**
        Add edge (A, I) to G for each $I \in \mathcal{A}$. ▷ Only important for level 0.
        ASSIGNPHIPOTENTIAL($\chi, \mathcal{A}, d_\mathcal{A}, \chi', c_\mathcal{A}, j, \mu_{\kappa'}$).
      end for
    end for
    $\mathcal{V} \leftarrow$ Subsample each $(V, \chi') \in \mathcal{I} \times \nu \setminus \{\perp\}$ with probability $\frac{\mu_{\kappa'}(V,\chi')}{\beta^l}$, independently k times.
    ASSIGNTHETAPOTENTIAL($V, \chi', m_\mathcal{V}((V, \chi')), l, \mu_{\kappa'}$) for all $(V, \chi') \in \text{supp}(\mathcal{V})$.
    $\kappa \leftarrow$ AMENDCOLORING(κ, l).
  end for
  Set $\mu_\kappa(I, \chi) = 0$ whenever $\mu_\kappa(I, \chi) \in (0, n^{-10})$ and renormalize each $\mu_\kappa(I, *)$ to a distribution.
  $\kappa \leftarrow$ ADJUST-$\mathbf{u}(\kappa)$.
  return (G, κ).
end function

The function InitColoring initializes an output coloring. It keeps all $\perp$ potentials from the input coloring κ' intact, since those are *already matched*, and sets the rest to $\mathbf{u}$.

Algorithm 9 Matching Phase - Init Coloring

Input: Input coloring κ', step t

Output: a new output coloring κ where all $\neq \perp$ mass is set to $\mathbf{u}$.

1: **function** InitColoring(κ', t)
2: Let $\nu \leftarrow [\lambda^t] \cup \{\mathbf{u}, \perp\}$.
3: Initialize new coloring κ over ν by setting $\mu_\kappa = 0$.
4: $\mu_\kappa(I, \perp) \leftarrow \mu_{\kappa'}(I, \perp)$ for $I \in \mathcal{I}$.
5: $\mu_\kappa(I, \mathbf{u}) \leftarrow \mu_{\kappa'}(I, \nu \setminus \perp)$ for $I \in \mathcal{I}$.
6: **return** κ.
7: **end function**

The function Adjust-$\mathbf{u}$ ensures the color $\mathbf{u}$ will have the same *sparsity guarantees* as the rest of colors in the end of step t. Such guarantees will be discussed in Section 3.7.1, but at high level, we would like to limit the number of intervals containing the $\mathbf{u}$ color. Since the only guarantee we have is for the minimal $\mathbf{u}$-mass between $\mathcal{X}, \mathcal{Y}$, we will remove all $\mathbf{u}$ color altogether if $\mu_\kappa(\mathcal{X}, \mathbf{u})$ and $\mu_\kappa(\mathcal{Y}, \mathbf{u})$ differ significantly, and otherwise, we use the $T^{\mathbf{u}}$ transformation to avoid the case where many intervals on both $\mathcal{X}$ and $\mathcal{Y}$ contain a tiny mass.

Algorithm 10 Matching Phase - Adjust $\mathbf{u}$

Input: Coloring κ at the end of step t with bounded $\mathbf{u}$-mass (in ℓ_1 sense).

Output: Adjusted coloring κ with bounded $\mathbf{u}$-support (in ℓ_0 sense).

function ADJUST-$\mathbf{u}(\kappa)$
  if $\frac{\mu_\kappa(\mathcal{X},\mathbf{u})}{\mu_\kappa(\mathcal{Y},\mathbf{u})} \notin [1/\beta, \beta]$ **then**
    $\mu_\kappa(I, \perp) \leftarrow \mu_\kappa(I, \perp) + \mu_\kappa(I, \mathbf{u})$ for $I \in \mathcal{I}$.
    $\mu_\kappa(I, \mathbf{u}) \leftarrow 0$ for $I \in \mathcal{I}$.
  end if
  $(\mu_\kappa(I, *_{\neq \mathbf{u}}), \mu_\kappa(I, \mathbf{u})) \leftarrow \frac{\left(\mu_\kappa(I,*_{\neq \mathbf{u}}), T^{\mathbf{u}}(\mu_\kappa(I,\mathbf{u}))\right)}{\left\|\left(\mu_\kappa(I,*_{\neq \mathbf{u}}), T^{\mathbf{u}}(\mu_\kappa(I,\mathbf{u}))\right)\right\|_1}$ for $I \in \mathcal{I}$.
  return κ.
end function

3.5 Correctness analysis of the interval matching algorithm

In this section, we prove correctness of the interval matching algorithm, namely Theorem 22, items 1 and 2. Item 3 (runtime) is proven in Section 3.7 later. Note that item 1 is immediate from the algorithm (as we only add edges if $\mathtt{ad}$ distance is $\leq O_\epsilon(\mathfrak{c})$). Hence we focus on item 2. To help the reader in the ensuing proofs, we collect important notations and definitions in Table 3 for quick reference.

Our central correctness lemma shows that the "corruption" in each level/step grows by at most a constant factor. Recall the notion of corruption from Def. 19: (I, χ) is F-corrupted pair if either: (1) $\pi[I] = \perp$; (2) $\mathtt{ad}(I, \pi[I]) > \mathfrak{c}$; (3) $\chi \neq \perp$ and $\mathtt{dd}_F(\mu_\kappa(I, \chi), \mu_\kappa(\pi[I], \chi)) > 0$; or (4) $\chi = \perp$ and $\mathrm{dist}_G(I, \pi[I]) > 2$. Also, recall from Eqn. (3.1) corruption per interval parameter $\xi_F^\kappa(I) = \sum_{\chi:(I,\chi) \text{ is } F\text{-corrupted pair}} \mu_\kappa(I, \chi)$ and the total corruption is defined as $\xi_F^\kappa = \xi_F^\kappa(\mathcal{I})$.

The following central lemma bounds corruption growth per level/step:

Lemma 27 (Corruption growth per level; proof in Section 3.5.2)**.** *Fix $\epsilon, \delta \in [0,1]$, and alignment $\pi \in \Pi$. Fix $F \in [n^{\alpha}, \beta^{o(\epsilon)}]$. Fix step t, level l, input coloring κ' (built at the previous step) and output coloring κ (being built in the current step). Then,* AmendColoring *at level l amends $\kappa \to \widehat{\kappa}$ such that $\xi_{\widehat{F}}^{\widehat{\kappa}} = O\left(\frac{\xi_F^{\kappa'} + \xi_F^{\kappa}}{\epsilon^{O(1/\epsilon)}}\right) + \tilde{O}_{\epsilon}\left(n^{-7}\right)$ with probability $1 - \epsilon^2$, where $\widehat{F} = F^{O(1/\epsilon^2)}$.*

Recalling that $\xi_F^{\kappa'}$ is the corruption at the end of the previous step, and ξ_F^{κ} is the corruption at the end of the previous level (in the current step), the above lemma bounds the multiplicative growth of the corruption of the amended coloring $\hat{\kappa}$, modulo a very small additive term.

In addition to this central lemma, we also need to bound the excess corruption incurred between steps, mainly from the Adjust-**u** procedure, which is captured by the following lemma.

Lemma 28 (Corruption growth by Adjust-**u**; proof in Section 3.5.3)**.** *Fix $\epsilon \in [0,1]$ and $F \in [n^{\alpha}, \beta^{o(\epsilon)}]$. The function* Adjust-**u** *amends $\kappa \to \widehat{\kappa}$ such that $\xi_{\widehat{F}}^{\widehat{\kappa}} = O\left(\xi_F^{\kappa}\right) + O\left(n^{-9}\right)$, where $\widehat{F} = F^{O(1/\epsilon^2)}$.*

While the rest of the section is devoted to proving the above lemmas, we first complete the proof of Lemma 22, item 2, which requires the following fact for preserving dd-distance on summations :

Fact 29. *For any $a_1, \ldots a_m, b_1 \ldots b_m \in \mathbb{R}_+^d$, we have that:*

$$\mathtt{dd}_{2F}(\sum_i a_i, \sum_i b_i) \leq 2 \sum_i \mathtt{dd}_F(a_i, b_i).$$

Proof. For $j \in [d]$, let $L_j \subseteq [m]$ be the set of coordinates where $a_{i,j} > F b_{i,j}$. If $\sum_{i \in L_j} a_{i,j} \leq \sum_{i \in [m] \setminus L_j} a_{i,j}$, then $\sum_i a_{i,j} \leq 2 \sum_{i \in [m] \setminus L_j} a_{i,j} \leq 2F \sum_i b_{i,j}$, and hence $\mathtt{dd}_{2F}(\sum_i a_{i,j}, \sum_i b_{i,j}) = 0$. Otherwise,

$$\mathtt{dd}_{2F}(\sum_i a_{i,j}, \sum_i b_{i,j}) \leq \sum_i a_{i,j} < 2 \sum_{i \in L_j} a_{i,j} = 2 \sum_i \mathtt{dd}_F(a_{i,j}, b_{i,j}).$$

Summing over all $j \in [d]$, we get:

$$\mathtt{dd}_{2F}(\sum_i a_i, \sum_i b_i) = \sum_j \mathtt{dd}_{2F}(\sum_i a_{i,j}, \sum_i b_{i,j}) \leq \sum_j \sum_i 2\, \mathtt{dd}_F(a_{i,j}, b_{i,j}) = 2 \sum_i \mathtt{dd}_F(a_i, b_i)$$

as needed. □

We also state the following complexity statement, bounding the size of parts $\mathcal{P}^{\chi}_{\kappa'}$, the set of intervals I with $\mu_\kappa(I, \chi) > 0$ (in "step input" coloring κ'). Its proof appears in Section 3.7.1.

Lemma 30 (Size of color parts)**.** *At each step t, for each color $\chi' \in \nu \setminus \{\perp\}$, we have that $|\mathcal{P}^{\chi'}_{\kappa'}| = n \cdot \tilde{O}_\epsilon(\beta^5 \cdot \lambda^{1-t})$ whp.*

An immediate corollary of Lemma 30 is that the total number of steps is bounded by $1/\epsilon + O(1)$ whp.

Proof of Lemma 22, item 2 using Lemma 27. Fix step t with input coloring κ'. Fix $F_t = n^{\alpha/\Theta(\epsilon)^{4t}}$. We first show that for each output coloring κ generated at each MatchStep call, we have with probability $1/2$, at the end of step t:

$$\xi^{\kappa}_{F_{t+1}} = \epsilon^{-O(1/\epsilon^2)} \cdot \xi^{\kappa'}_{F_t} + \tilde{O}_\epsilon(n^{-8}).$$

To do that, we use Lemma 27, to obtain that in each level $l \leq O(1/\epsilon)$ we have $O(\epsilon^{-O(1/\epsilon)} \cdot 4/\epsilon)$ factor growth in corruption with probability $1 - \frac{\epsilon}{4}$, and by the union bound we get overall blow-up $\epsilon^{-O(1/\epsilon^2)}$ with probability $3/4$ (as we have $O(1/\epsilon)$ levels). Observe also that removing pairs with mass $< n^{-10}$ in Line 19 introduces at most $O(F_{t+1} \cdot n^{-7}) < O(n^{-6})$ additive corruption, since there are $\leq n^3$ total non-zero pairs and each pair removed can generate at most $F_{t+1} \cdot n^{-10}$ corrupted mass. Finally, notice that by Fact 29, the excess corruption introduced by InitColoring in any new step is bounded by a factor of 2. Also, Adjust-$\mathbf{u}$ introduces excess $O(1)$-factor, and $O(1/n^9)$ additive corruption, and $\hat{F} = F^{O(1/\epsilon^2)}$ distortion growth (Lemma 28). Overall, we note that distortion grows as $O(F_t)^{O(1/\epsilon^4)} \leq F_{t+1}$.

Now, notice we generate $O(\log n)$ i.i.d colorings in each step t for each input coloring κ', hence, we must generate a "good" coloring κ with high probability as long as we started with at least one "good" coloring from the previous step. Now, for a fixed $\mathfrak{c}$, define $k_{\mathfrak{c}} \triangleq |\{i \mid \mathtt{ad}_{w,\mathfrak{c}}(X_{i,w}, Y_{\pi(i),w}) > \mathfrak{c}\}|$ and notice we start the MatchIntervals algorithm with $\xi^{\kappa}_{F_1} = k_{\mathfrak{c}}$ and generate $O_\epsilon(1)$ blow-up per step.

Finally, since MatchIntervals halts when $\mu_\kappa(\mathcal{I}, \perp) = 2n$ (this is the halting condition from Line 6), and by the corollary above the total number of steps is $O_\epsilon(1)$, we have $\xi^{\kappa}_{F_t} = O_\epsilon(k_{\mathfrak{c}} + 1/n^6)$ for any "good" coloring κ. Furthermore, for any π-matchable pair $I, \pi[I]$ with $\mathtt{ad}_{w,\mathfrak{c}}(I, \pi[I]) \leq \mathfrak{c}$, if the hop-path between $I, \pi[I]$ in G is more than 2, then this pair contributes 1 to $\xi^{\kappa}_{F_t} = O_\epsilon(k_{\mathfrak{c}} + 1/n^6)$. We conclude that all but $O_\epsilon(k_{\mathfrak{c}})$ π-matchable pairs have a 2-hop path in G as needed. □

3.5.1 Bounding corruption growth per level: two key lemmas

To prove our central correctness Lemma 27, we introduce the following two key Lemmas. We refer the reader to Table 3 for a quick recap of important quantities and formulas from our algorithm.

In particular, note that the amended coloring is obtained via the following formula:

$$\mu_{\widehat{\kappa}}(I,*) = \mu_\kappa(I,*_{\neq \mathbf{u}}) + \mu_\kappa(I,\mathbf{u}) \cdot \frac{\mu_\kappa(I,*_{\neq \mathbf{u}}) + \sum_{\zeta \in Z_l} \varphi_\kappa^\zeta(I,*)}{\|\mu_\kappa(I,*_{\neq \mathbf{u}}) + \sum_{\zeta \in Z_l} \varphi_\kappa^\zeta(I,*)\|_1}$$

Thus, in order to bound the growth of $\xi_{\widehat{F}}^{\widehat{\kappa}}$, we need to bound the quantities: $\mathtt{dd}(\varphi_\kappa^\zeta(I,*), \varphi_\kappa^{\zeta'}(\pi[I],*))$, for some ζ, ζ', as well as the ℓ_1 normalization from above. These two goals correspond to the two key lemmas.

To state the key lemmas, we introduce the following measures ρ of the corruption of intervals from nearby intervals. First, define $\mathcal{B}_\pi(\mathcal{S})$ for $\mathcal{S}$ being a subset of $\mathcal{X}$ or $\mathcal{Y}$ as the smallest enclosing ball around $\pi[\mathcal{S}]$ (ignoring the $\perp$'s). Now, fix an interval $I \in \mathcal{I}$, output coloring κ, and arbitrary coloring κ' (which can be either an input or output coloring). Fix distortion F, and set $\mathcal{S} \subseteq \mathcal{I}$ with $\mu_\kappa(\mathcal{S},\mathbf{u}) > 0$ that is a subset of $\mathcal{X}$ or $\mathcal{Y}$. We define "local" corruption measures $\rho_F^{\kappa,\kappa'}(I,\mathcal{S})$ and $\widetilde{\rho_F^{\kappa,\kappa'}}(I,\mathcal{S})$ as follows:

$$\rho_F^{\kappa,\kappa'}(I,\mathcal{S}) = \tfrac{\mu_\kappa(I,\mathbf{u})}{\mu_\kappa(\mathcal{S},\mathbf{u})} \cdot \xi_F^{\kappa'}(\mathcal{S}),$$
$$\widetilde{\rho_F^{\kappa,\kappa'}}(I,\mathcal{S}) = \tfrac{\mu_\kappa(I,\mathbf{u})}{\mu_\kappa(\mathcal{S},\mathbf{u})} \cdot \xi_F^{\kappa'}(\mathcal{B}_\pi(\mathcal{S})).$$

The first lemma below bounds the expected distortion-resistant corruption of $\varphi_\kappa^\zeta(I,\cdot), \varphi_\kappa^{\zeta'}(\pi[I],\cdot)$ for a fixed interval I, as a function of its ρ scores of the neighborhoods of $I, \pi[I]$. The lemma considers arbitrary ζ (radius around I), and assumes

there is a convenient ζ' (radius around $\pi[I]$) that is not much larger than ζ (otherwise there's a lot of error in this region already).

Lemma 31. *Fix step t, level $l \geq 0$, and alignment $\pi \in \Pi$. Fix input coloring κ' (obtained at the end of step $t-1$), and the current coloring κ (obtained at the end of level $l-1$ or from* InitColoring*). Fix $F \in [n^{\alpha}, \beta^{o(\epsilon)}]$. Consider $I \in \mathcal{I}$, where $(I, \mathbf{u})$ is not F-corrupted pair. For $\zeta \in Z_l$ when $l \geq 1$ or $\zeta = 0$ for $l = 0$, define $\mathcal{B}_I = \Lambda_{\kappa}^{\zeta}(I)$, $\mathcal{B}_I^{+2\zeta} = \Lambda_{\kappa}^{3\zeta}(I)$, and suppose there exists $\zeta' \in Z_l \cap [0, F^2\zeta]$ such that $\pi[\Lambda_{\kappa}^{3\zeta}(I)] \subseteq \Lambda_{\kappa}^{\zeta'/2}(\pi[I])$. Then, for $\widehat{F} \triangleq F^{\Theta(1/\epsilon^2)}$:*

1. *For $l \geq 1$, in expectation over the random choices of A, χ', c in the algorithm, we have*

$$\mathbb{E}\left[\mathtt{dd}_{\widehat{F}}\left(\varphi_{\kappa}^{\zeta}(I, *_{\neq \mathbf{u}}), \varphi_{\kappa}^{\zeta'}(\pi[I], *_{\neq \mathbf{u}})\right)\right] = O\left(\frac{\beta^l}{\zeta} \cdot \rho_F^{\kappa,\kappa'}(I, \mathcal{B}_I) + \frac{1}{n^8}\right). \tag{3.7}$$

2. *For $l \geq 0$, with high probability,*

$$\mathtt{dd}_{\widehat{F}}\left(\varphi_{\kappa}^{\zeta}(I, \mathbf{u}), \varphi_{\kappa}^{\zeta'}(\pi[I], \mathbf{u})\right) = O\left(\frac{\beta^l}{\max\{\zeta, 1\}} \cdot \rho_F^{\kappa,\kappa'}(I, \mathcal{B}_I^{+2\zeta}) + \frac{1}{n^9}\right). \tag{3.8}$$

The second key lemma argues that normalizing each vector $\mu(I, *)$ does not add more than constant corruption in each level.

Lemma 32. *At the end of level $l \geq 0$, for each coloring κ, we have $\left\|\mu_{\kappa}(I, *_{\neq \mathbf{u}}) + \sum_{\zeta \in Z_l} \varphi_{\kappa}^{\zeta}(I, *)\right\|_1 = (\Omega(\epsilon))^{l+1}$ for each $I \in \mathcal{I}$ with high probability.*

The proofs of Lemmas 31 and 32 are involved and appear in Section 3.6. We prove Lemma 22 using these two key lemmas in Section 3.5.2 after introducing a few useful facts.

3.5.2 Proof of Lemma 27 from key correctness lemmas: corruption growth per level

Before continuing with the correctness analysis, we establish some auxiliary statements.

Distortion Resilient Distance Properties. We show some properties of Distortion Resilient Distance from Def. 20. Recall that $\mathtt{dd}_F(p,q) = \sum_{i:p_i > F\cdot q_i} p_i$. First, we show multiplication by scalars.

Fact 33. *Fix $p, q \in \mathbb{R}^n_+$ and scalars $a, b \in \mathbb{R}_+$. We have $\mathtt{dd}_F(a\cdot p, b\cdot q) = a\cdot \mathtt{dd}_{F\cdot\frac{b}{a}}(p,q)$*

Proof.

$$\mathtt{dd}_F(a\cdot p, b\cdot q) = \sum_{i:\frac{ap_i}{bq_i} > F} ap_i = a \sum_{i:\frac{p_i}{q_i} > \frac{bF}{a}} p_i = a\cdot \mathtt{dd}_{F\cdot\frac{b}{a}}(p,q)$$

□

We also show a bound on the ℓ_1 normalization of vectors.

For two vectors $p, q \in \mathbb{R}^n_+$ and $F \geq 1$, define $\mathtt{dd}_F(p,q) = \sum_{i:p_i/q_i > F} p_i$.

Fact 34. *Fix $p, q \in \mathbb{R}^n_+$ and $a, b \in \mathbb{R}_+$ with $a \geq \|p\|_1$, $b \geq \|q\|_1$ and $a+b-\|p\|_1-\|q\|_1 = o(F)\cdot\min\{a,b\}$. We have,*

$$\mathtt{dd}_{3\cdot F^2}\left(\tfrac{p}{a},\tfrac{q}{b}\right) + \mathtt{dd}_{3\cdot F^2}\left(\tfrac{q}{b},\tfrac{p}{a}\right) \leq 4\cdot\left(\tfrac{\mathtt{dd}_F(p,q)}{a} + \tfrac{\mathtt{dd}_F(q,p)}{b}\right)$$

Proof. By fact 33, we have that,

$$\mathtt{dd}_{3\cdot F^2}\left(\tfrac{p}{a},\tfrac{q}{b}\right) + \mathtt{dd}_{3\cdot F^2}\left(\tfrac{q}{a},\tfrac{p}{b}\right) = \tfrac{1}{a}\cdot\mathtt{dd}_{3\cdot F^2\cdot\frac{a}{b}}(p,q) + \tfrac{1}{b}\cdot\mathtt{dd}_{3\cdot F^2\cdot\frac{b}{a}}(q,p)$$

Now, consider a case where $\frac{a}{b} \in [1/3F, 3F]$, then we have that, using the observation that $\mathtt{dd}_F(\cdot,\cdot)$ is decreasing in F:

$$\mathtt{dd}_{3\cdot F^2}\left(\tfrac{p}{a},\tfrac{q}{b}\right) + \mathtt{dd}_{3\cdot F^2}\left(\tfrac{q}{b},\tfrac{p}{a}\right) \leq \tfrac{1}{a}\cdot\mathtt{dd}_F(p,q) + \tfrac{1}{b}\cdot\mathtt{dd}_F(q,p)$$

Otherwise, assume w.l.o.g $a > 3Fb$. The statement guarantees gives us $a = \|p\|_1 (1 + o(1))$ and $b \geq \|q\|_1$, implying $\frac{\|p\|_1}{\|q\|_1} > (3 - o(1))F$, then on one hand we have that $\mathtt{dd}_F(p, q) \geq 0.6 \|p\|_1$ (by Fact 29), and hence the RHS of the claim is at least $4 \cdot 0.6 \cdot \frac{\|p\|_1}{a} > 2$. On the other hand, $\mathtt{dd}_{3F^2}\left(\frac{p}{a}, v\right) \leq \left\|\frac{p}{a}\right\|_1 \leq 1$ for any v, and similarly for q, and hence the LHS is at most 2 and the claim follows.

□

Alternative Counting for Λ balls. When bounding certain parameters, we use the following simple combinatorial claim:

Claim 35. *Fix coloring κ, parameter $\zeta \geq 1$ and function $f : \mathcal{I} \to \mathbb{R}_+$. Then,*

$$\sum_{I \in \mathcal{I}} \mu_\kappa(I, \mathbf{u}) \cdot f(\Lambda^\zeta_\kappa(I)) = \sum_{I \in \mathcal{I}} \mu_\kappa(\Lambda^\zeta_\kappa(I), \mathbf{u}) \cdot f(I) \leq 2\zeta f(\mathcal{I}).$$

Proof. The proof follows immediate by counting the contribution of each $f(I)$ to elements in $\Lambda^\zeta_\kappa(I)$. Formally:

$$\sum_{I \in \mathcal{I}} \mu_\kappa(I, \mathbf{u}) \cdot f(\Lambda^\zeta_\kappa(I)) = \sum_{I \in \mathcal{I}} \mu_\kappa(I, \mathbf{u}) \cdot \sum_{J \in \Lambda^\zeta_\kappa(I)} f(J) = \sum_{I \in \mathcal{I}} \sum_{J \in \mathcal{I}} \mathbf{1}[\mu(\mathrm{SEB}(I, J)^{10}, \mathbf{u}) \leq \zeta] \cdot \mu_\kappa(I, \mathbf{u}) f(J)$$

Now, by change of summation, such quantity equals

$$\sum_{J \in \mathcal{I}} \sum_{I \in \mathcal{I}} \mathbf{1}[\mu(\mathrm{SEB}(I, J), \mathbf{u}) \leq \zeta] \cdot \mu_\kappa(I, \mathbf{u}) f(J) = \sum_{J \in \mathcal{I}} f(J) \cdot \sum_{I \in \Lambda^\zeta_\kappa(J)} \mu_\kappa(I, \mathbf{u}) = \sum_{J \in \mathcal{I}} \mu_\kappa(\Lambda^\zeta_\kappa(J), \mathbf{u}) \cdot f(J).$$

And since $\mu_\kappa(\Lambda^\zeta_\kappa(J), \mathbf{u}) \leq 2\zeta$, we conclude:

[10] $\mathrm{SEB}(I, J)$ = smallest enclosing ball containing both intervals

Table 3: Summary of main notations and their definitions.

Parameters			
ϵ	Arbitrary small constant; complexity is $n^{1+O(\epsilon)}$.	α	$= \epsilon^{5/\epsilon}$. We define $F_0 = n^\alpha$.
λ	$= n^\epsilon$. Sample size growths by factor λ each step.	η	$= \alpha^2\epsilon^3$. The set of costs $E_{\mathfrak{c}}$ has size $1/\eta$.
β	$= n^\epsilon$. Base radius growths by β factor each level.	k	$= \beta^3(\log n)^{\Theta(1)}$, pivot oversampling factor.
$j_{\max}$	$= O(1/\alpha)$, number of costs per anchor.		
Transformations			
$T^q_{\delta,\gamma}(x)$	$= \begin{cases} x_i & x_i \geq \gamma \\ 0 & x_i < \delta\gamma \\ \gamma \cdot (\frac{x_i}{\gamma})^q & \text{Otherwise} \end{cases}$	$T^{\mathbf{c}}$	$= T^{\Theta(1/\epsilon)}_{1/\beta,1/\beta^3}$
		T^{θ}	$= T^{O(1/\epsilon)}_{1/\sqrt{\beta},1}$
		$T^{\mathbf{u}}$	$= T^{O(1/\epsilon)}_{1/\beta,\Omega_\epsilon(1)}$
$Q_{\delta,s,F}(x)$	$= \max_{J\subseteq[d]} a_{\lvert J\rvert} \cdot \min_{j\in J} x_j$ where $a_i = \begin{cases} 1 & i \geq s\cdot d \\ 0 & i < (s-\delta)d \\ 1/F^{sd-i} & \text{Otherwise} \end{cases}$		
$Q_l(x)$	$= Q_{\frac{\epsilon}{4},1-\frac{(l+1)\epsilon}{2},n^{O(\alpha^2)}}(x)$		
Potentials			
$\phi_\kappa(I,\chi)$	$= \mu_{\kappa'}(I,\chi')\frac{2n}{\lambda^t}\sum_{j=0}^{j_{\max}} \frac{n^{-\alpha j}\cdot \mathbb{1}[\mathtt{ad}(A,I)\leq \hat{c}+j\mathfrak{c}]}{\widehat{d_{I,j}}}$, basic potential from matching anchor $(A,\chi',\hat{c})$		
$\widehat{d_{I,j}}$	$= \Theta\left(\max\{d_{\kappa'}(I,\chi')_{\hat{c}+j\mathfrak{c}}, \frac{1}{n^\alpha}\cdot d_{\kappa'}(A,\chi')_{3\hat{c}}\}\right)$, approximated density for a matched interval I		
$\Lambda^\zeta_\kappa(I)$	largest interval ball containing $\leq \zeta$ of ℓ_1-mass of $\mathbf{u}$-color in μ on the left and right of I		
$\varphi^\zeta_\kappa(I,*_{\neq\mathbf{u}})$, $\zeta > 0$	$= T^{\mathbf{c}}\left(\mu_\kappa(I,\mathbf{u})\cdot\frac{\beta^{2l}}{\zeta^2}\cdot\frac{\phi_\kappa(\Lambda^\zeta_\kappa(I),*)}{\beta^l}\right)$		
$\Gamma_{l,\zeta}(V,\chi')$	a vector of costs in $E_{\mathfrak{c}}$ with $\Gamma_{l,\zeta}(V,\chi') = \min\left\{\widehat{\mathrm{rd}_{\kappa'}(V,\chi',\Lambda^\zeta_\kappa(V))}^{-1}\cdot n^{4\alpha}\cdot\frac{n}{\beta^l\cdot\lambda^t},1\right\}$		
$\sigma_{V,\chi'}$	$= Q_l(T^\theta(\Gamma_{l,\zeta}(V,\chi')))$		
$\mathcal{V}$	multiset of pivots; = subsampling $(V,\chi')\in\mathcal{I}\times\nu\setminus\{\perp\}$ k times with prob. $\mu_{\kappa'}(V,\chi')\cdot\beta^{-l}$		
$m_{\mathcal{V}}((V,\chi'))$	multiplicity of (V,χ') in $\mathcal{V}$		
$\theta^\zeta_\kappa(V,\chi')$	$= m_{\mathcal{V}}((V,\chi'))\cdot\begin{cases}\sigma_{V,\chi'} & \mu_{\kappa'}(\Lambda^\zeta_{\kappa'}(V),\chi')\leq\lambda\cdot\beta^6\cdot\beta^l \\ 1 & \text{Otherwise}\end{cases}$		
$Z_l, l>0$	$= \beta^l\cdot\{1,2,4,8,\ldots,n/\beta^l\}$, is the set of radiuses ζ		
Z_0	$= \{0\}$		
$\varphi^\zeta_\kappa(I,\mathbf{u})$, $\zeta>0$	$= T^{\mathbf{u}}\left(\mu_\kappa(I,\mathbf{u})\cdot\frac{\beta^{2l}}{\zeta^2}\cdot\min\left\{\frac{1}{k}\cdot\sum_{\chi'\in\nu\setminus\{\perp\}}\theta^\zeta_\kappa(\Lambda^\zeta_\kappa(I),\chi'),1\right\}\right)$		
$\varphi^\zeta_\kappa(I,\mathbf{u})$, $\zeta=0$	$= T^{\mathbf{u}}\left(\mu_\kappa(I,\mathbf{u})\cdot\frac{\theta^0_\kappa(I,\nu\setminus\{\perp\})}{k}\right)$		
$\varphi^0_\kappa(I,\perp)$	$= T^{\mathbf{c}}(\mu_\kappa(I,\mathbf{u})\cdot\phi_\kappa(I,\nu\setminus\{\mathbf{u},\perp\}))$ ($\varphi^l_\kappa(I,\perp)$ defined only for $l=0$ level)		
$\mu_{\widehat{\kappa}}(I,*)$	$= \mu_\kappa(I,*_{\neq\mathbf{u}}) + \mu_\kappa(I,\mathbf{u})\cdot\frac{\mu_\kappa(I,*_{\neq\mathbf{u}})+\sum_{\zeta\in Z_l}\varphi^\zeta_\kappa(I,*)}{\lVert\mu_\kappa(I,*_{\neq\mathbf{u}})+\sum_{\zeta\in Z_l}\varphi^\zeta_\kappa(I,*)\rVert_1}$ (always a distribution over χ's)		
Error measures			
$\xi^\kappa_F(I)$	$= \begin{cases}1 & \pi[I]=\perp \text{ or } \mathtt{ad}(I,\pi[I])>\mathfrak{c} \\ \mathtt{dd}_F(\mu_\kappa(I,*_{\neq\perp}),\mu_\kappa(\pi[I],*_{\neq\perp}))+\mu_\kappa(I,\perp)\mathbf{1}[\mathrm{dist}_G(I,\pi[I])>2] & \text{otherwise}\end{cases}$		
$\rho^{\kappa,\kappa'}_F(I,\mathcal{S})$	$= \frac{\mu_\kappa(I,\mathbf{u})}{\mu_\kappa(\mathcal{S},\mathbf{u})}\cdot\xi^{\kappa'}_F(\mathcal{S})$		
$\widetilde{\rho^{\kappa,\kappa'}_F}(I,\mathcal{S})$	$= \frac{\mu_\kappa(I,\mathbf{u})}{\mu_\kappa(\mathcal{S},\mathbf{u})}\cdot\xi^{\kappa'}_F(\mathcal{B}_\pi(\mathcal{S}))$, where $\mathcal{B}_\pi(\mathcal{S})$ is the smallest enclosing ball around $\pi[\mathcal{S}]$		

$$\sum_{I\in\mathcal{I}} \mu_\kappa(I,\mathbf{u})\cdot f(\Lambda_\kappa^\zeta(I)) = \sum_{I\in\mathcal{I}} \mu_\kappa(\Lambda_\kappa^\zeta(I),\mathbf{u})\cdot f(I) \le 2\zeta f(\mathcal{I}).$$

as needed.

□

We now proceed to the proof of Lemma 27. Recall that Lemma 27 states that, in a fixed level, the new coloring $\widehat{\kappa}$ (amended from κ) satisfies $\xi_{\widehat{F}}^{\widehat{\kappa}} = O\left(\frac{\xi_F^{\kappa'}+\xi_F^{\kappa}}{\epsilon^{2/\epsilon}\cdot\delta}\right) + \tilde{O}\left(n^{-8}\right)$ with probability $1-\delta$, where $\widehat{F} = F^{O(1/\epsilon^2)}$. For reader's convenience, we include a summary table of notations in Table 3.

Proof of Lemma 27 using Lemmas 31, 32. Fix distortion F, level l, and consider an interval $I \in \mathcal{I}\backslash\pi^{-1}[\perp]$ with $\mathtt{ad}(I,\pi[I]) \le c$. Let $m_I = \mu_\kappa(I,\mathbf{u})$ and $m_{\pi[I]} = \mu_\kappa(\pi[I],\mathbf{u})$. If either $(I,\mathbf{u})$ or $(\pi[I],\mathbf{u})$ are corrupted — meaning $\mathtt{dd}_F(m_I, m_{\pi[I]})+\mathtt{dd}_F(m_{\pi[I]}, m_I) > 0$ and hence $\mathtt{dd}_F(m_I, m_{\pi[I]})+\mathtt{dd}_F(m_{\pi[I]}, m_I) \ge \max\{m_I, m_{\pi[I]}\}$ — then $\xi_F^\kappa(\{I,\pi[I]\}) \ge \frac{m_I+m_{\pi[I]}}{2}$ (merely from $\mathbf{u}$-color). The new coloring $\widehat{\kappa}$, obtained at the end of level l, updates at most $m_I + m_{\pi[I]}$ total mass (at $I, \pi[I]$). Hence, we have, for $\xi^* = \xi_{2F}^{\widehat{\kappa}}(\{I,\pi[I]\}) - \xi_F^\kappa(\{I,\pi[I]\})$, and using that $\mathtt{dd}_F(a+b,c) \le \mathtt{dd}_F(a,c)+b$ for $a,b,c \ge 0$:

$$\xi^* \le \sum_{I'\in\{I,\pi[I]\}} \mathtt{dd}_{2F}(\mu_{\widehat{\kappa}}(I',*_{\neq\perp}), \mu_{\widehat{\kappa}}(\pi[I'],*_{\neq\perp})) + (\mu_{\widehat{\kappa}}(I',\perp) - \mu_\kappa(I',\perp)) \tag{3.9}$$

$$\overset{Fact\ 29}{\le} 2\sum_{I'\in\{I,\pi[I]\}} \mathtt{dd}_F(\mu_\kappa(I',*_{\neq\perp}), \mu_\kappa(\pi[I'],*_{\neq\perp})) + 3m_{I'} \tag{3.10}$$

$$\le O(\xi_F^\kappa(\{I,\pi[I]\})). \tag{3.11}$$

where for (9), we note that $(I',\perp)$ can only be corrupted in $\mu_{\widehat{\kappa}}$ if it was previously corrupted. Below, we consider intervals I such that both $(I,\mathbf{u})$ and $(\pi[I],\mathbf{u})$ are not F-corrupted, i.e., $\frac{m_I}{m_{\pi[I]}} \in [1/F, F]$.

To bound the new corruption $\xi_{\hat{F}}^{\widehat{\kappa}}(I)$, we analyze two cases: the base case of level $l = 0$, where our main goal is to match intervals and the corruption added is for *already matched* pairs (colored $\perp$) without a short path in G, and the general case of $l \geq 1$, and our goal is to move μ-mass from $\mathbf{u}$-color to $\chi \in [\lambda^t]$ color, in similar proportion for both I and $\pi[I]$. In both cases, we analyze the expectation of $\xi_{\widehat{F}}^{\widehat{\kappa}}$ as a function of $\xi_F^{\kappa'}, \xi_F^{\kappa}$. We note that throughout this section, we consider the probability space over the sampled anchors and pivots in the MatchStep algorithm, and all expectations are over that space.

Recall our coloring update procedure replaces the $\mathbf{u}$-mass of I by vector $\mu_\kappa(I, \mathbf{u}) \cdot \frac{\mu_\kappa(I,*_{\neq \mathbf{u}}) + \sum_{\zeta \in Z_l} \varphi_\kappa^\zeta(I,*)}{\left\| \mu_\kappa(I,*_{\neq \mathbf{u}}) + \sum_{\zeta \in Z_l} \varphi_\kappa^\zeta(I,*) \right\|_1}$.

Base case of level $l = 0$. At the beginning of the step, we have that $\mu_\kappa(I, \perp) = \mu_{\kappa'}(I, \perp)$ and $\mu_\kappa(I, \mathbf{u}) = \mu_{\kappa'}(I, \nu \setminus \perp)$. Furthermore, at level 0, we amend only the $\mathbf{u}$ and $\perp$ color potentials. For $\mathbf{u}$, we invoke Lemma 31 (2) and obtain (whp) that

$$\mathtt{dd}_{\widehat{F}}\left(\varphi_\kappa^0(I, \mathbf{u}), \varphi_\kappa^0(\pi[I], \mathbf{u})\right) = O\left(\rho_F^{\kappa,\kappa'}(I, \{I\}) + \tfrac{1}{n^9}\right) = O\left(\xi_F^{\kappa'}(I) + \tfrac{1}{n^9}\right).$$

For $\perp$, observe that $\varphi_\kappa^0(I, \perp) \leq \phi_\kappa(I, \nu \setminus \{\mathbf{u}, \perp\})$ at level 0. Now, consider an *F-uncorrupted pair* $(I, \chi') \in \mathcal{I} \times \nu \setminus \{\perp\}$ that has been clustered by an anchor A at color χ. By Claim 26 (1), we have $\mathtt{dd}_{O(F \cdot n^{2\alpha})}(\phi_\kappa(I, \chi), \phi_\kappa(\pi[I], \chi)) \leq n^{-10}$ whp. Notice also that whenever $\phi_\kappa(I, \chi) > 0$, we add an edge (A, I) to G which is uniquely identified by color χ. Therefore, one of the following must hold:

1. We add a 2-hop path between $I, \pi[I]$ in G; or
2. The total contribution of *F-uncorrupted pairs* to $\varphi_\kappa^0(I, \perp)$ is $O(n^{-9})$.

Focusing on corrupted pairs (I, χ'), note that there is $\xi_F^{\kappa'}$ corrupted mass in $\mathcal{I}$; i.e.,

total $\mu_{\kappa'}$ of (I,χ') with $\mathtt{dd}_F(\mu_{\kappa'}(I,\chi'),\mu_{\kappa'}(\pi[I],\chi'))>0$ is at most $\xi_F^{\kappa'}$. Recall that we sample λ^t anchors and by Claim 26 (2), each such anchor generates, in expectation, potential $O(\frac{\mu_{\kappa'}(I,\chi')}{\lambda^t})$.

Therefore, for each (I,χ') pair, the added potential over all sampled anchors is a r.v. with expectation $O(\mu_{\kappa'}(I,\chi'))$. Overall, we obtain the expected contribution of corrupted pairs to $\varphi_\kappa^0(I,\perp)$ is $O(\xi_F^{\kappa'}(I))$. Define $u_I=\mu_\kappa(I,*_{\neq\mathbf{u}})+\varphi_\kappa^0(I,*)$ and $Z_I\triangleq\mathbf{1}[\mathrm{dist}_G(I,\pi[I])>2]$. Notice that, as there are only $\mathbf{u}$ and $\perp$ colors to consider at $l=0$. Then, for $\widehat{\xi}=\sum_{j=I,\pi[I]}\mathbb{E}\left[\xi_{\widehat{F}}^{\widehat{\kappa}}(J)\right]$, we have:

$$\begin{aligned}
\widehat{\xi} &\overset{Fact\ 29}{\leq} Z_I\cdot\sum_{j=I,\pi[I]}\mathbb{E}[\mu_\kappa(J,\perp)]+\sum_{j=I,\pi[I]}\mathbb{E}\left[\mathtt{dd}_{\hat{F}/2}(m_J\cdot\tfrac{u_J}{\|u_J\|_1},m_{\pi[J]}\cdot\tfrac{u_{\pi[J]}}{\left\|u_{\pi[J]}\right\|_1})\right]\\
&\overset{Fact\ 33}{\leq} \xi_F^\kappa(\{I,\pi[I]\})+\sum_{j=I,\pi[I]}m_J\cdot\mathbb{E}\left[\mathtt{dd}_{\hat{F}/2F}(\tfrac{u_J}{\|u_J\|_1},\tfrac{u_{\pi[J]}}{\left\|u_{\pi[J]}\right\|_1})\right]\\
&\overset{Fact\ 34}{\leq} \xi_F^\kappa(\{I,\pi[I]\})+\max_{J=I,\pi[I]}\left\{\tfrac{4}{\|u_J\|_1}\right\}\cdot\sum_{J=I,\pi[I]}m_J\cdot\mathbb{E}\left[\mathtt{dd}_{\sqrt{\hat{F}/4F}}(u_J,u_{\pi[J]})\right]\\
&\overset{Fact\ 29}{\leq} \xi_F^\kappa(I)+\max_{J=I,\pi[I]}\left\{\tfrac{4}{\|u_J\|_1}\right\}\cdot\sum_{J=I,\pi[I]}\left(O(\xi_F^\kappa(J)+\xi_F^{\kappa'}(J))+E\left[\mathtt{dd}_{\sqrt{\hat{F}/4F}}(\varphi_\kappa^0(J,\mathbf{u}),\varphi_\kappa^0(\pi[J],\mathbf{u}))\right]\right)\\
&\leq \xi_F^\kappa(I)+\max\left\{\tfrac{1}{\|u_I\|_1},\tfrac{1}{\left\|u_{\pi[I]}\right\|_1}\right\}\cdot O\left(\xi_F^\kappa(\{I,\pi[I]\})+\xi_F^{\kappa'}(\{I,\pi[I]\})+\tfrac{1}{n^9}\right).
\end{aligned}$$

By Lemma 32, we have $\|u_I\|_1,\left\|u_{\pi[I]}\right\|_1=\Omega(\epsilon)$. Taking into account Eqn. (3.9) for corrupted $(I,\mathbf{u})$, we therefore have in expectation $\mathbb{E}[\xi_{\widehat{F}}^{\widehat{\kappa}}(\{I,\pi[I]\})-\xi_F^\kappa(\{I,\pi[I]\})]=O\left(\frac{1}{\epsilon}(\xi_F^\kappa(\{I,\pi[I]\})+\xi_F^{\kappa'}(\{I,\pi[I]\})+n^{-9})\right)$ overall for each interval. Summing over all intervals (including ones that map to $\perp$ or are at distance more than $\mathfrak{c}$), we conclude $\mathbb{E}[\xi_{\widehat{F}}^{\widehat{\kappa}}]=O\left(\frac{1}{\epsilon}(\xi_F^\kappa+\xi_F^{\kappa'}+n^{-8})\right)$.

General case. Consider level $l\geq 1$. Intuitively, the new corruption added to I is driven by 1) vectors φ that differ between I and $\pi[I]$ (in $\mathtt{dd}$ sense), and 2)

normalization. For a fixed $\zeta \in Z_l$, we shall bound the quantity $\mathtt{dd}_{F'}(\varphi_\kappa^\zeta(I), \varphi_\kappa^{\zeta'}(\pi[I]))$, for some $\zeta' \in Z_l, F' \geq F$, by the following quantity in expectation, up to a constant factor:

$$\rho_I^\zeta \triangleq \frac{\beta^l}{\zeta} \cdot \rho_F^{\kappa,\kappa'}(I, \Lambda_\kappa^{3\zeta}(I)) + \frac{1}{\log n} \cdot \widetilde{\rho_F^{\kappa,\kappa}}(I, \Lambda_\kappa^{3\zeta}(I)) \geq \frac{m_I}{6\zeta} \left(\frac{\beta^l}{\zeta} \cdot \xi_F^{\kappa'}(\Lambda_\kappa^{3\zeta}(I)) + \frac{1}{\log n} \xi_F^{\kappa}(\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I))) \right), \tag{3.12}$$

where the first error term will come from standard error analysis (bounded via Lemma 31), and the second error term comes from the case when ζ' is too large.

Specifically, for $\zeta' \in Z_l$ defined as the smallest such that $\pi[\Lambda_\kappa^{3\zeta}(I)] \subseteq \Lambda_\kappa^{\zeta'/2}(\pi[I])$, we would like to use Lemma 31, but for that we need ζ' to be upper bounded by $F^2\zeta$. Let L_ζ be the set of intervals I for which the required property is not satisfied: namely, the resulting $\zeta' > 100 \cdot F \cdot \zeta \cdot \log n$ (and $(I, \mathbf{u}), (\pi[I], \mathbf{u})$ are uncorrupted). We note that if $I \in L_\zeta$, we have $\mu_\kappa(\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I)), \mathbf{u}) \geq 50F \cdot \zeta \cdot \log n$, and since $\mu_\kappa(\Lambda_\kappa^{3\zeta}(I), \mathbf{u}) \leq 6\zeta$ (by definition of Λ-balls), this in turn implies most of $\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I)) \times \{\mathbf{u}\}$ is F-corrupted—since $\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I)) \times \{\mathbf{u}\}$ maps (under π) back into $\Lambda_\kappa^{3\zeta}(I) \times \{\mathbf{u}\}$ and specifically:

$$\begin{aligned} &\xi_F^\kappa(\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I)) \geq \mu_\kappa(\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I)), \mathbf{u}) - \mu_\kappa(\Lambda_\kappa^{3\zeta}(I), \mathbf{u}) \geq 50F \cdot \zeta \cdot \log n - F \cdot 6\zeta \\ &\overset{Eqn.\ (3.12)}{\Rightarrow} \rho_I^\zeta \geq 4Fm_I \geq 4m_{\pi[I]}. \end{aligned}$$

Overall $m_I + m_{\pi[I]} \leq O(\rho_I^\zeta)$, and hence, just like for the corrupted I's, the new corruption is bounded by $O(\rho_I^\zeta)$, and thus $O(\sum_{\zeta \in Z_l} \rho_I^\zeta)$ (removing the explicit dependence on a fixed ζ). Henceforth, we assume that $I \notin \cup_\zeta L_\zeta$, and thus $\zeta' = O(F \cdot \zeta \cdot \log n) \leq F^2\zeta$ and we can use Lemma 31.

By Lemma 31, we have that there exists $F' = F^{\Theta(1/\epsilon^2)}$, and hence $F' \geq (F\zeta'/\zeta)^{\Theta(1/\epsilon^2)} \cdot 2\log n$, such that:

$$\begin{aligned}
&\mathbb{E}\left[\mathtt{dd}_{\frac{F'}{2\log n}}\left(\varphi_\kappa^\zeta(I), \varphi_\kappa^{\zeta'}(\pi[I])\right)\right] \\
&\quad = \mathbb{E}\left[\mathtt{dd}_{\frac{F'}{2\log n}}\left(\varphi_\kappa^\zeta(I, *_{\neq \mathbf{u}}), \varphi_\kappa^{\zeta'}(\pi[I], *_{\neq \mathbf{u}})\right) + \mathtt{dd}_{\frac{F'}{2\log n}}\left(\varphi_\kappa^\zeta(I, \mathbf{u}), \varphi_\kappa^{\zeta'}(\pi[I], \mathbf{u})\right)\right] \\
&\quad = O(\rho_I^\zeta + n^{-8}),
\end{aligned}$$

noting that while part (1) is a "whp" statement, it also implies "in expectation" statement with extra $1/n^9$ term. We now aggregate the contribution from all ζ's, obtaining:

$$\begin{aligned}
\mathbb{E}\left[\mathtt{dd}_{F'}\left(\sum_{\zeta\in Z_l}\varphi_\kappa^\zeta(I), \sum_{\zeta\in Z_l}\varphi_\kappa^\zeta(\pi[I])\right)\right] &\overset{\text{Fact 33}}{=} \mathbb{E}\left[\mathtt{dd}_{\frac{F'}{|Z_l|}}\left(\sum_{\zeta}\varphi_\kappa^\zeta(I), |Z_l|\cdot\sum_{\zeta}\varphi_\kappa^\zeta(\pi[I])\right)\right] \\
&\overset{\text{Fact 29}}{\leq} 2\,\mathbb{E}\left[\sum_{\zeta}\mathtt{dd}_{\frac{F'}{2|Z_l|}}\left(\varphi_\kappa^\zeta(I), \sum_{\zeta}\varphi_\kappa^\zeta(\pi[I])\right)\right] \\
&\leq 2\sum_{\zeta}\mathbb{E}\left[\mathtt{dd}_{\frac{F'}{2\log n}}\left(\varphi_\kappa^\zeta(I), \varphi_\kappa^{\zeta'}(\pi[I])\right)\right] \\
&= O\left(\sum_{\zeta}(\rho_I^\zeta + n^{-8})\right).
\end{aligned}$$

Finally, we consider the contribution from normalization. Define $u_I = \mu_\kappa(I, *_{\neq \mathbf{u}}) + \sum_{\zeta\in Z_l}\varphi_\kappa^\zeta(I)$ and $v_I = \mu_\kappa(\pi[I], *_{\neq \mathbf{u}}) + \sum_{\zeta\in Z_l}\varphi_\kappa^\zeta(\pi[I])$. We split the vectors by the $\perp$ coordinate $u_I(\perp)$ and the rest $u_I(*_{\neq\perp})$. For the $\perp$ color, using Fact 32 to bound the effect of normalization, we have:

$$\frac{u_I(\perp)}{\|u_I\|_1} \leq O\left(\tfrac{1}{\epsilon^{l+1}}\right)\cdot u_I(\perp) = O\left(\tfrac{1}{\epsilon^{l+1}}\right)\cdot \mu_\kappa(I, \perp),$$

and since $(I, \perp)$ is only corrupted in $\widehat{\kappa}$ if it was corrupted in κ, we can bound the total contribution of $\perp$ corruption by:

$$\text{Contribution of } \perp \text{ to } \xi^{\widehat{\kappa}}_{\widehat{F}}(\mathcal{I}) = O\left(\tfrac{1}{\epsilon^{l+1}}\right) \cdot \xi^{\kappa}_{F}(\mathcal{I}). \tag{3.13}$$

For the other colors, we use Lemma 34 on vectors $u_I(*_{\neq\perp}), v_I(*_{\neq\perp})$ with $a = \|u_I\|_1, b = \|v_I\|_1$, noting that $a + b - \|u_I(*_{\neq\perp})\|_1 - \|v_I(*_{\neq\perp})\|_1 = \mu_\kappa(I,\perp) + \mu_\kappa(\pi[I],\perp) \le 2 \le o(F)\min\{a,b\}$ (using Lemma 32 on a, b again):

$$\begin{aligned}
&\mathbb{E}\left[\mathtt{dd}_{8F'^2}\left(\tfrac{u_I(*_{\neq\perp})}{\|u_I\|_1}, \tfrac{v_I(*_{\neq\perp})}{\|v_I\|_1}\right) + \mathtt{dd}_{8F'^2}\left(\tfrac{v_I(*_{\neq\perp})}{\|v_I\|_1}, \tfrac{u_I(*_{\neq\perp})}{\|u_I\|_1}\right)\right] \\
&\quad\le O\left(\tfrac{1}{\epsilon^{l+1}}\right) \cdot \mathbb{E}\left[\mathtt{dd}_{2F'}(u_I, v_I) + \mathtt{dd}_{2F'}(v_I, u_I)\right] &(3.14)\\
&\quad\overset{Fact\ 29}{=} O\left(\tfrac{1}{\epsilon^{l+1}}\right) \cdot \left(\xi^{\kappa}_F(I) + \xi^{\kappa}_F(\pi[I]) + \sum_{\zeta\in Z_l}(\rho^{\zeta}_I + \rho^{\zeta}_{\pi[I]} + n^{-8})\right). &(3.15)
\end{aligned}$$

We now compute the aggregate contribution $\xi^{\widehat{\kappa}}_{\widehat{F}}(\mathcal{I})$ over all intervals $I \in \mathcal{I}$, where $\widehat{F} = 8F'^2 \cdot F = F^{\Theta(1/\epsilon^2)}$. For this, we need to also bound the sum of all ρ^{ζ}_I quantities as a function of total corruption ξ^{κ}_F, which we do in the next claim.

Claim 36. *Fix $l > 0$ and assume $\mu_\kappa(\mathcal{Y}, \mathbf{u}), \mu_\kappa(\mathcal{X}, \mathbf{u}) \ge 1$. Then, $\sum_{I\in\mathcal{I}}\sum_{\zeta\in Z_l}\rho^{\zeta}_I = O(\xi^{\kappa'}_F + \xi^{\kappa}_F)$.*

Proof. Recall that $\rho^{\zeta}_I = \frac{\beta^l}{\zeta}\rho^{\kappa,\kappa'}_F(I, \Lambda^{3\zeta}_\kappa(I)) + \frac{1}{\log n}\widetilde{\rho^{\kappa,\kappa}_F}(I, \Lambda^{3\zeta}_\kappa(I))$. We bound the sum of each term separately. First, let $r_{\mathcal{X}} = \min\{3\zeta, \mu_\kappa(\mathcal{X}, \mathbf{u})\}$ and $r_{\mathcal{Y}} = \min\{3\zeta, \mu_\kappa(\mathcal{Y}, \mathbf{u})\}$. Then $\mu_\kappa(\Lambda^{3\zeta}_\kappa(I), \mathbf{u}) \in [r_{\mathcal{X}}, 2r_{\mathcal{X}}]$ for $I \in \mathcal{X}$ (and similarly for $\mathcal{Y}$) and:

$$\begin{aligned}
\sum_{I\in\mathcal{I}}\rho^{\kappa,\kappa'}_F(I, \Lambda^{3\zeta}_\kappa(I)) &\le \sum_{I\in\mathcal{X}}\tfrac{\mu_\kappa(I,\mathbf{u})}{r_{\mathcal{X}}} \cdot \xi^{\kappa'}_F(\Lambda^{3\zeta}_\kappa(I)) + \sum_{I\in\mathcal{Y}}\tfrac{\mu_\kappa(I,\mathbf{u})}{r_{\mathcal{Y}}} \cdot \xi^{\kappa'}_F(\Lambda^{3\zeta}_\kappa(I)) \\
&\overset{Claim\ 35}{=} \tfrac{1}{r_{\mathcal{X}}}\sum_{I\in\mathcal{X}}\mu_\kappa(\Lambda^{3\zeta}_\kappa(I), \mathbf{u}) \cdot \xi^{\kappa'}_F(I) + \tfrac{1}{r_{\mathcal{Y}}}\sum_{I\in\mathcal{Y}}\mu_\kappa(\Lambda^{3\zeta}_\kappa(I), \mathbf{u}) \cdot \xi^{\kappa'}_F(I) \\
&\le \tfrac{1}{r_{\mathcal{X}}}\sum_{I\in\mathcal{X}} 2r_{\mathcal{X}} \cdot \xi^{\kappa'}_F(I) + \tfrac{1}{r_{\mathcal{Y}}}\sum_{I\in\mathcal{Y}} 2r_{\mathcal{Y}} \cdot \xi^{\kappa'}_F(I) \\
&\le 4\xi^{\kappa'}_F.
\end{aligned}$$

Summing over $\zeta \in Z_l$, we get $\sum_{I\in\mathcal{I}}\sum_{\zeta\in Z_l}\frac{\beta^l}{\zeta}\rho_F^{\kappa,\kappa'}(I,\Lambda_\kappa^{3\zeta}(I)) = O(\xi_F^{\kappa'})$. For the second term, defining $\widetilde{\xi_F^\kappa}(I) \triangleq \xi_F^\kappa(\overleftarrow{\pi}^{-1}(I)) = \xi_F^\kappa(\{I' \in \mathcal{I} \mid \overleftarrow{\pi}(I') = I\})$[11]:

$$\begin{aligned}
\sum_{I\in\mathcal{I}}\widetilde{\rho_F^{\kappa,\kappa}}(I,\Lambda_\kappa^{3\zeta}(I)) &\le \sum_{I\in\mathcal{X}}\tfrac{\mu_\kappa(I,\mathbf{u})}{r_\mathcal{X}}\cdot\xi_F^\kappa(\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I))) + \sum_{I\in\mathcal{Y}}\tfrac{\mu_\kappa(I,\mathbf{u})}{r_\mathcal{Y}}\cdot\xi_F^\kappa(\mathcal{B}_\pi(\Lambda_\kappa^{3\zeta}(I))) \\
&\le \sum_{I\in\mathcal{X}}\tfrac{\mu_\kappa(I,\mathbf{u})}{r_\mathcal{X}}\cdot\widetilde{\xi_F^\kappa}(\Lambda_\kappa^{3\zeta}(I)) + \sum_{I\in\mathcal{Y}}\tfrac{\mu_\kappa(I,\mathbf{u})}{r_\mathcal{Y}}\cdot\widetilde{\xi_F^\kappa}(\Lambda_\kappa^{3\zeta}(I)) \\
&\overset{Claim\ 35}{=} \tfrac{1}{r_\mathcal{X}}\sum_{I\in\mathcal{X}}\mu_\kappa(\Lambda_\kappa^{3\zeta}(I),\mathbf{u})\cdot\widetilde{\xi_F^\kappa}(I) + \tfrac{1}{r_\mathcal{Y}}\sum_{I\in\mathcal{Y}}\mu_\kappa(\Lambda_\kappa^{3\zeta}(I),\mathbf{u})\cdot\widetilde{\xi_F^\kappa}(I) \\
&\le \tfrac{1}{r_\mathcal{X}}\sum_{I'\in\mathcal{Y}}\mu_\kappa(\Lambda_\kappa^{3\zeta}(\overleftarrow{\pi}(I')),\mathbf{u})\cdot\xi_F^\kappa(I') + \tfrac{1}{r_\mathcal{Y}}\sum_{I'\in\mathcal{X}}\mu_\kappa(\Lambda_\kappa^{3\zeta}(\overleftarrow{\pi}(I')),\mathbf{u})\cdot\xi_F^\kappa(I') \\
&\le \tfrac{1}{r_\mathcal{X}}\sum_{I\in\mathcal{Y}}3r_\mathcal{X}\cdot\xi_F^\kappa(I) + \tfrac{1}{r_\mathcal{Y}}\sum_{I\in\mathcal{X}}3r_\mathcal{Y}\cdot\xi_F^\kappa(I) \\
&\le 6\xi_F^\kappa.
\end{aligned}$$

Summing over $\zeta \in Z_l$, we get $\sum_{I\in\mathcal{I}}\sum_{\zeta\in Z_l}\frac{1}{\log n}\widetilde{\rho_F^{\kappa,\kappa}}(I,\Lambda_\kappa^{\zeta}(I)) \le \frac{|Z_l|}{\log n}\cdot 6\xi_F^\kappa = O(\xi_F^\kappa)$. By summing up both terms, the claim follows. □

Finally, we can estimate the amount of "new corruption", $\sum_{I\in\mathcal{I}}\mathtt{dd}_{\widehat{F}}(\mu_{\widehat{\kappa}}(I,*),\mu_{\widehat{\kappa}}(\pi[I],*)) - \xi_F^\kappa$, by summing Eqn. (3.15) over all intervals $I \in \mathcal{I}$, and using Claim 36 (noting we only call AMENDCOLORING when $\mu_\kappa(\mathcal{Y},\mathbf{u}), \mu_\kappa(\mathcal{X},\mathbf{u}) \ge \beta^l = \omega(1)$ for level $l > 0$). Below, let $\mathcal{U} \subseteq \mathcal{I}$ denote the intervals I such that both $I, \pi[I]$ are not F-corrupted on $\mathbf{u}$, and $L = \cup_\zeta L_\zeta$:

[11] Recall $\overleftarrow{\pi}(I) = \pi(I+j)$, where $j \ge 0$, is the minimal one such that $\pi(I+j) \ne \perp$.

$$
\begin{aligned}
\mathbb{E}[\xi^{\widehat{\kappa}}_{\widehat{F}}] &= \sum_{I\in\mathcal{I}} \mathbb{E}[\xi^{\widehat{\kappa}}_{\widehat{F}}(I)] \\
&= \sum_{I\in\mathcal{I}\setminus\mathcal{U}} \mathbb{E}[\xi^{\widehat{\kappa}}_{\widehat{F}}(I)] + \sum_{I\in L} \mathbb{E}[\xi^{\widehat{\kappa}}_{\widehat{F}}(I)] + \sum_{I\in\mathcal{U}\setminus L} \mathbb{E}[\xi^{\widehat{\kappa}}_{\widehat{F}}(I)] \\
&\le O(\xi^{\kappa}_{F}) + \sum_{I\in L,\zeta\in Z_l} O(\rho^{\zeta}_{I}) + \sum_{I\in\mathcal{U}\setminus L} \mathbb{E}[\xi^{\widehat{\kappa}}_{\widehat{F}}(I)] \\
&\overset{\substack{Claim\ 36}}{\le} O(\xi^{\kappa}_{F} + \xi^{\kappa'}_{F}) + \sum_{I\in\mathcal{U}\setminus L} \mathbb{E}[\xi^{\widehat{\kappa}}_{\widehat{F}}(I)] \\
&\overset{\substack{Fact\ 29\\ Eqn.\ (3.13)}}{\le} O(\tfrac{\xi^{\kappa}_{F}+\xi^{\kappa'}_{F}}{\epsilon^{l+1}}) \\
&\qquad + 2\sum_{I\in\mathcal{U}\setminus L} \mathbb{E}\left[\mathtt{dd}_{\widehat{F}/2}(\mu_\kappa(I,*_{\neq\mathbf{u}}),\mu_\kappa(\pi[I],*_{\neq\mathbf{u}})) + \mathtt{dd}_{\widehat{F}/2}(m_I\tfrac{u_I(*_{\neq\perp})}{\|u_I\|_1}, m_{\pi[I]}\tfrac{v_I(*_{\neq\perp})}{\|v_I\|_1})\right] \\
&\overset{\substack{Fact\ 33\\ Eqn.\ (3.15)}}{\le} O(\tfrac{\xi^{\kappa}_{F}+\xi^{\kappa'}_{F}}{\epsilon^{l+1}}) + \sum_{I\in\mathcal{U}\setminus L} O\left(\tfrac{1}{\epsilon^{l+1}}\right)\cdot\left(\xi^{\kappa}_{F}(I) + \xi^{\kappa}_{F}(\pi[I]) + \sum_{\zeta\in Z_l}(\rho^{\zeta}_{I} + \rho^{\zeta}_{\pi[I]} + n^{-8})\right) \\
&= O(\tfrac{\xi^{\kappa}_{F}+\xi^{\kappa'}_{F}}{\epsilon^{l+1}}) + O\left(\tfrac{1}{\epsilon^{l+1}}\right)\cdot\sum_{I\in\mathcal{I}}\left(\xi^{\kappa}_{F}(I) + \sum_{\zeta\in Z_l}(\rho^{\zeta}_{I} + n^{-8})\right) \\
&\overset{\substack{Claim\ 36}}{\le} O(\tfrac{\xi^{\kappa}_{F}+\xi^{\kappa'}_{F}}{\epsilon^{l+1}}) + O\left(\tfrac{1}{\epsilon^{l+1}}\right)\cdot\left(\xi^{\kappa}_{F} + \xi^{\kappa'}_{F} + \tilde{O}(n^{-7})\right) \\
&= O\left(\tfrac{1}{\epsilon^{O(1/\epsilon)}}\right)\cdot\left(\xi^{\kappa}_{F} + \xi^{\kappa'}_{F}\right) + \tilde{O}(n^{-7}).
\end{aligned}
$$

The soundness of the Lemma is obtained using the Markov inequality. □

3.5.3 dd under soft operators and proof of Lemma 28

Before proving Lemma 28, we argue how soft operators are resilient under the dd distance. In particular, we show that dd distance behaves nicely under the soft operators of thresholding T and quantile Q (which are used precisely for that reason). Such property will be pivotal in the construction and will be used extensively in the next section, and while the proof of Lemma 28 requires only the use of soft

thresholding T, we introduce them here together for exposition.

Claim 37. *For any $x, y \geq 0$, we have $\mathtt{dd}_{F^q}(T^q_{\delta,\gamma}(x), T^q_{\delta,\gamma}(y)) < \mathtt{dd}_F(x,y) + \gamma(\delta F)^q$.*

Proof. Let $t_x = T^q_{\delta,\gamma}(x)$ and $t_y = T^q_{\delta,\gamma}(y)$ We first observe that $t_x \leq x$, and hence if $x > Fy$, then $\mathtt{dd}_{F^q}(t_x, t_y) \leq t_x \leq x = \mathtt{dd}_F(x,y)$. Now, suppose $x \leq Fy$. If $y < \delta\gamma$, then $t_y = 0$ and $\mathtt{dd}_{F^q}(t_x, t_y) \leq t_x \leq \gamma \cdot (\delta F)^q$ as required. If $y \in [\delta\gamma, \gamma]$, we have $t_y = \gamma \cdot (y/\gamma)^q \geq F^{-q}\gamma \cdot (x/\gamma)^q \geq t_x/F^q$. Finally, for $y > \gamma$, we have $t_y = y$ and so $\mathtt{dd}_{F^q}(t_x, t_y) \leq \mathtt{dd}_{F^q}(x,y) = 0$. □

Claim 38. *Fix $\delta < 1, s \in [0,1], F, F' > 1$ and integers v, d with $v < \delta d$. Also fix $x, y \in [0,1]^d$. For any set of coordinates $U \subseteq [d]$ of size $d - v$ we have: $\mathtt{dd}_{F'\cdot F^v}(Q_{\delta,s,F}(x), Q_{\delta,s,F}(y)) \leq \mathtt{dd}_{F'}(x_U, y_U) + 1/F^{\delta d - v}$.*

Proof. Let $q_x = Q_{\delta,s,F}(x)$, $q_y = Q_{\delta,s,F}(y)$. Consider the set J which maximizes q_x. We have $|J \cap U| \geq |J| - v$. We consider two cases. First, suppose $a_{|J\cap U|} \geq a_{|J|} \cdot F^{-v}$, and let $j^* \in J \cap U$ be minimizing y_{j^*}, then:

$$\begin{aligned}
\mathtt{dd}_{F'\cdot F^v}(q_x, q_y) &\leq \mathtt{dd}_{F'F^v}(a_{|J|} \cdot \min_{j\in J} x_j, a_{|J\cap U|} \cdot \min_{j \in J\cap U} y_j) \\
&\leq \mathtt{dd}_{F'F^v}(a_{|J|} \cdot x_{j^*}, a_{|J\cap U|} \cdot y_{j^*}) \\
&\overset{Fact\ 33}{\leq} a_{|J|} \cdot \mathtt{dd}_{F'}(x_{j^*}, y_{j^*}) \\
&\leq \mathtt{dd}_{F'}(x_U, y_U).
\end{aligned}$$

Second, if $a_{|J|} \leq 1/F^{\delta d - v}$, then $\mathtt{dd}_{F'F^k}(q_x, q_y) \leq q_x \leq F^{v-\delta d} \cdot \|x\|_\infty \leq F^{v-\delta d}$. This concludes the proof.

□

Now consider the specific choice $Q_l(x)$ where $\delta = \epsilon/4, s = 1 - \frac{(l+1)\epsilon}{2}, F = n^{O(\alpha^2)}$ (Table 3).

Corollary 39. *Suppose* $v = \Theta(1/\alpha)$ *with* $\epsilon d = \Omega(1/\alpha^2)$. *Fix* $x, y \in [0,1]^d$ *with* $\|x\|_\infty \le 1$. *For any set of coordinates* $U \subseteq [d]$ *of size* $d-v$ *we have:* $\mathtt{dd}_{F \cdot n^{O(\alpha)}}(Q_l(x), Q_l(y)) \le \mathtt{dd}_F(x_U, y_U) + n^{-10}$.

Proof. Recall $\delta = \epsilon/4$ in Q_l. Apply Claim 38 with $F' = F$, $F = n^{C \cdot \alpha^2}$ for some large enough constant C. □

We now prove the remaining Lemma 28.

Proof of Lemma 28. We first consider the case $\frac{\mu_\kappa(\mathcal{X},\mathbf{u})}{\mu_\kappa(\mathcal{Y},\mathbf{u})} \notin [1/\beta, \beta]$, and, wlog, assume $\frac{\mu_\kappa(\mathcal{X},\mathbf{u})}{\mu_\kappa(\mathcal{Y},\mathbf{u})} > \beta$. That implies that all except $< F/\beta$ fraction of $\mu_\kappa(\mathcal{X}, \mathbf{u}) \ge (1 + 1/\beta)\mu_\kappa(\mathcal{I}, \mathbf{u})$ is already corrupted. Now since $F < \beta/2$, and only $(*, \mathbf{u})$-pairs are amended, that implies $\xi_{\widehat{F}}^{\widehat{\kappa}} \le \xi_F^{\widehat{\kappa}} = O\left(\xi_F^\kappa\right)$ as needed.

Now for the other case, we use Claim 37 to obtain for any $\hat{F} \ge 1$ and each $I \in \mathcal{I}$:

$$\mathtt{dd}_{\hat{F}}\left(T^{\mathbf{u}}(\mu_\kappa(I,\mathbf{u})), T^{\mathbf{u}}(\mu_\kappa(\pi[I],\mathbf{u}))\right) \le \mathtt{dd}_{\hat{F}^{\epsilon/10}}\left(\mu_\kappa(I,\mathbf{u}), \mu_\kappa(\pi[I],\mathbf{u})\right) + n^{-10}.$$

Now, for fixed I, applying Fact 34, and noting that $\|\mu_\kappa(I, *_{\neq \mathbf{u}}), T^{\mathbf{u}}(\mu_\kappa(I,\mathbf{u}))\| \ge 1/2$ (and same for $\pi[I]$), we obtain:

$$\begin{aligned}
&\mathtt{dd}_{\widehat{F}}\left(\mu_{\widehat{\kappa}}(I, *_{\neq \perp}), \mu_{\widehat{\kappa}}(\pi[I], *_{\neq \perp})\right) \\
&\quad \le 16\left(\mathtt{dd}_{\sqrt{\frac{\widehat{F}}{4}}}\left(T^{\mathbf{u}}(\mu_\kappa(I,\mathbf{u})), T^{\mathbf{u}}(\mu_\kappa(\pi[I],\mathbf{u}))\right) + \mathtt{dd}_{\sqrt{\frac{\widehat{F}}{4}}}\left(\mu_\kappa(I, *_{\neq \mathbf{u},\perp}), \mu_\kappa(\pi[I], *_{\neq \mathbf{u},\perp})\right)\right) \\
&\quad = O\left(\mathtt{dd}_{\widehat{F}^{\epsilon/23}}\left(\mu_\kappa(I,*), \mu_\kappa(\pi[I],*)\right) + n^{-10}\right).
\end{aligned}$$

For $\perp$ color, we again use the fact that we rescale by at most a factor 2, and that $\perp$ contributes to $\xi_{\hat{F}}^{\hat{\kappa}}$ iff it contributes to $\xi_{\hat{F}}^\kappa$. The same argument applies to I where $\pi[I] = \perp$ or $\mathtt{ad}(I, \pi[I]) > \mathfrak{c}$. Finally, we sum over all $I \in \mathcal{I}$, proving the claim. □

3.6 Proof of key correctness Lemmas 31, 32

We now prove the key correctness Lemmas 31, 32. Before proceeding, we develop several supporting statements, in particular:

- We provide a proof for density approximation algorithm, Lemma 25;
- We provide a proof for ϕ color assignment properties, Claim 26.

Analysis for Approximating Densities. We now analyze the performance of ApproxDensity and ApproxRelativeDensity (Alg. 3), proving Lemma 25.

Proof of Lemma 25. Let $d = \mathrm{d}_\kappa(I, \chi, \mathcal{S})_c$ and let $\widehat{d} = \mu_\kappa(\widetilde{\mathcal{S}^*}, \chi) + \frac{1}{\gamma} \cdot |\mathcal{S}^* \setminus \widetilde{\mathcal{S}^*}|$ be the estimator. Also, let $d' = \mathrm{d}_\kappa(I, \chi, \mathcal{S} \setminus \widetilde{\mathcal{S}^*})_c$ and $s = |\mathcal{S}^* \setminus \widetilde{\mathcal{S}^*}|$. Note that s is a sum of independent random variables in $[0, 1]$ with expectation $\gamma d'$. If $d' \geq \Omega(\frac{\log n}{\gamma})$, then by Chernoff bound, s would concentrate whp: $s = \Theta(\gamma d')$. Otherwise, if $d' < O(\frac{\log n}{\gamma})$, then whp $s/\gamma \leq O(\frac{\log n}{\gamma}) = \mathrm{d}_m$ for appropriately chosen constant. Hence, for (1), the algorithm outputs whp, up to the $\Theta(\cdot)$ rescaling:

$$\begin{aligned}\max\{\widehat{d}, \mathrm{d}_m\} &= \max\{\mathrm{d}_\kappa(I, \chi, \widetilde{\mathcal{S}^*})_c + \Theta(1) \cdot \max\{\mathrm{d}_\kappa(I, \chi, \mathcal{S} \setminus \widetilde{\mathcal{S}^*})_c, \mathrm{d}_m\}, \mathrm{d}_m\} \\ &= \Theta(1) \cdot \max\{\mathrm{d}_\kappa(I, \chi, \mathcal{S})_c, \mathrm{d}_m\}.\end{aligned}$$

For run-time, we have from Theorem 49, that generating the set $\mathcal{S}'$ costs $\tilde{O}(\gamma\mu_\kappa(\mathcal{S}, \chi) + 1) = \tilde{O}\left(\frac{\mu_\kappa(\mathcal{S},\chi)}{\mathrm{d}_m} + 1\right)$, and since $|\mathcal{S}'|$ is bounded by that amount as well. The total complexity hence is $T_D = T_{\mathsf{ad}} \cdot \tilde{O}\left(\frac{\mu_\kappa(\mathcal{S},\chi)}{\mathrm{d}_m} + 1\right)$.

For (2), we first note that since $I \in \mathcal{S}$, then $d \geq \mu_\kappa(I, \chi)$, and since we set $\mathrm{d}_m = \mu_\kappa(I, \chi)$ in the first call to ApproxDensity, we have that $\widehat{d} = \Theta(d)$ whp. Also, let $D = \mathrm{d}_\kappa(I, \chi)_c$, noting that $D/d = \mathrm{rd}_\kappa(I, \chi, \mathcal{S})_c$. Now, consider the second

call to ApproxDensity, and let $\widehat{D}$ be the output; by part (1), we have that $\widehat{D} = \Theta(1) \cdot \max\{D, \mathrm{rd}_m\, \mu_\kappa(I,\chi)\}$.

Combining the bounds on $\widehat{D}$ and $\widehat{d}$, we have that whp, using that $d \geq \mu_\kappa(I,\chi)$:

$$\max\left\{\tfrac{\hat{D}}{\hat{d}}, \mathrm{rd}_m\right\} = \Theta(1)\max\{\tfrac{D}{d}, \tfrac{\mathrm{rd}_m\,\mu_\kappa(I,\chi)}{d}, \mathrm{rd}_m\} = \Theta(1)\cdot\max\{\tfrac{D}{d}, \mathrm{rd}_m\}.$$

For runtime, we have $T_{RD} = T_D(\hat{d}) + T_D(\hat{D}) + \tilde{O}(1) = T_{\mathsf{ad}} \cdot \tilde{O}\left(\frac{\mu_\kappa(\mathcal{S},\chi)}{\mu_\kappa(I,\chi)} + \frac{1}{\mathrm{rd}_m} \cdot \frac{\mu_\kappa(\mathcal{I},\chi)}{\mu_\kappa(I,\chi)}\right)$ as needed. □

Analysis for ϕ invariants. We now prove the properties of the ϕ potentials, in particular Claim 26.

Proof of Claim 26. Fix pair (I,χ'), and $\hat{c} \in E_{\mathfrak{c}}$. Assuming that (I,χ') was clustered by some anchor (A,χ''), for fixed $j \in \{0,\dots j_{\max}\}$, let $m_{I,\chi',j} = n^{-\alpha j} \cdot \frac{2n}{\lambda^t} \cdot \frac{\mu_{\kappa'}(I,\chi')}{\widehat{d_{I,j}}}$, where $\hat{c}_j = \hat{c} + j\mathfrak{c}$, and $\widehat{d_{I,j}}$ the approximation of $\max\{\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}, n^{-\alpha} d_{\mathcal{A}}\}$ with $d_{\mathcal{A}} = \mathrm{d}_{\kappa'}(A,\chi'')_{3\hat{c}}$. For a color $\chi \in [\lambda^t]$, the contribution of (I,χ') to $\phi_\kappa(I,\chi)$ is $\sum_{j\in\{0,\dots,j_{\max}\}} m_{I,\chi',j} \cdot \mathbb{1}[\mathsf{ad}(A,I) \leq \hat{c}_j]$. Conditioned on all densities being approximated well (whp event from Lemma 25), we have:

$$\begin{aligned}
&\mathbb{E}_{A,\chi''}\left[m_{I,\chi',j} \cdot \mathbb{1}[(I,\chi') \text{ clustered by } (A,\chi'',j)]\right] \\
&\quad= \Pr_{A,\chi''}\left[A \in \mathcal{N}_{\hat{c}_j}(I) \wedge \chi' = \chi''\right] \cdot n^{-\alpha j} \cdot \mathbb{E}_{A\in\mathcal{N}_{\hat{c}_j}(I)}\left[\frac{2n}{\lambda^t} \cdot \frac{\mu_{\kappa'}(I,\chi')}{\widehat{d_{I,j}}}\right] \\
&\quad= n^{-\alpha j} \cdot \sum_{A\in\mathcal{N}_{\hat{c}_j}(I)} \frac{\mu_{\kappa'}(A,\chi')}{2n} \cdot \frac{2n}{\lambda^t} \cdot \mathbb{E}_{A\in\mathcal{N}_{\hat{c}_j}(I)}\left[\frac{\mu_{\kappa'}(I,\chi')}{\widehat{d_{I,j}}}\right] \\
&\quad= n^{-\alpha j} \cdot \mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j} \cdot \frac{1}{\lambda^t} \cdot \mathbb{E}_{A\in\mathcal{N}_{\hat{c}_j}(I)}\left[\frac{\mu_{\kappa'}(I,\chi')}{[\Omega(1),1]\cdot\max\{\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}, n^{-\alpha} d_{\mathcal{A}}\}}\right]
\end{aligned}$$

The last equality is since $\sum_{A\in\mathcal{N}_{\hat{c}_j}(I)} \mu_{\kappa'}(A,\chi') = \mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}$, and that $\widehat{d_{I,j}}$ is a constant-factor approximation to $\max\{\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}, n^{-\alpha} d_{\mathcal{A}}\}$. Summing over all j, this

immediately gives us the required upper bound for item (2), as we need only to consider the first component of the max:

$$\mathbb{E}_{A,\chi'',\hat{c}}\left[\phi_\kappa(I,\chi)\cdot\mathbf{1}[\chi'=\chi'']\right]=O\left(\tfrac{\mu_{\kappa'}(I,\chi')}{\lambda^t}\right).$$

For item (3), recall that $d_{\mathcal{A}}=\mathrm{d}_{\kappa'}(A,\chi')_{3\hat{c}}$ and since $I\in\mathcal{N}_{\hat{c}_j}(A)$, we have that $d_{\mathcal{A}}\le\mathrm{d}_{\kappa'}(I,\chi')_{6\hat{c}}$ (by triangle inequality) . Therefore, specifically for $j=0$, whenever $\max\{\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j},d_{\mathcal{A}}\cdot n^{-\alpha}\}>\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}$ we have that $\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}\le n^{-\alpha}\cdot\mathrm{d}_{\kappa'}(I,\chi')_{6\hat{c}}$, which can happen only for at most $O(1/\alpha)$ costs $\hat{c}\in E_{\mathfrak{c}}$ (using the fact all densities are between n^{-10} and n). Now, consider any other "good cost" and $j=0$; we get the following bound:

$$\mathbb{E}_{A,\chi''}\left[m_{I,\chi',j}\cdot\mathbb{1}[(I,\chi')\text{ clustered by }(A,\chi'',j)]\right]\ge\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}\cdot\tfrac{1}{\lambda^t}\cdot\tfrac{\mu_{\kappa'}(I,\chi')}{\mathrm{d}_{\kappa'}(I,\chi')_{\hat{c}_j}}=\mu_{\kappa'}(I,\chi')/\lambda^t,$$

Since the above quantity is a lower bound on the total potential, and we are sampling a cost from a universe of size $|E_{\mathfrak{c}}|=1/\eta$, this implies that for all but $O(\eta/\alpha)$ fraction of costs $\hat{c}\in\mathbb{E}_{\mathfrak{c}}$:

$$\mathbb{E}_{A,\chi''}\left[\phi_\kappa(I,\chi)\cdot\mathbf{1}[\chi'=\chi'']\right]\ge\tfrac{\mu_{\kappa'}(I,\chi')}{\lambda^t}$$

as needed for (3).

It remains to prove item (1). Assume (I,χ'') is *not F-corrupted*, i.e., $\mu_{\kappa'}(I,\chi'')\le F\cdot\mu_{\kappa'}(\pi[I],\chi'')$. Consider any $j<j_{\max}$, and recall the clustering property — that $\mathcal{N}_{\mathfrak{c}}(\mathcal{A}_j)\cap\mathcal{P}_{\kappa'}^{\chi''}\subseteq\mathcal{A}_{j+1}$ — and hence $\pi[I]\in\mathcal{A}_{j+1}$. Also, we have that $\widehat{d_{I,j}}$ is at least $\Omega(n^{-\alpha}d_{\mathcal{A}})=\Omega(n^{-\alpha}\,\mathrm{d}_{\kappa'}(A,\chi'')_{3\hat{c}})$, and hence

$$\frac{\widehat{d_{I,j}}}{\widehat{d_{\pi[I],j+1}}} = \Omega\left(\frac{\max\{d_{\mathcal{A}}n^{-\alpha}, \mathrm{d}_{\kappa'}(I,\chi'')_{\hat{c}_j}\}}{\max\{d_{\mathcal{A}}n^{-\alpha}, \mathrm{d}_{\kappa'}(\pi[I],\chi'')_{\hat{c}_{j+1}}\}}\right) = \Omega\left(\min\{1, \frac{d_{\mathcal{A}}n^{-\alpha}}{\mathrm{d}_{\kappa'}(A,\chi'')_{3\hat{c}}}\}\right) = \Omega(n^{-\alpha}),$$

where the second derivation is by triangle inequality and that $c_{j+1} + c_j + \mathfrak{c} \leq 3\hat{c}$. Therefore,

$\mathtt{dd}_{O(F\cdot n^{2\alpha})}(m_{I,\chi'',j}, m_{\pi[I],\chi'',j+1}) = 0$ for any $j < j_{\max}$. For $j = j_{\max} \geq 12/\alpha$, we have $m_{I,\chi'',j_{\max}} \leq n^{-\alpha j_{\max}} \cdot 2n \leq n^{-12} \cdot 2n \leq 2n^{-11}$, and by summing over all $m_{I,\chi'',j}$ and $m_{\pi[I],\chi'',j}$ (and using Fact 29), we obtain $\mathtt{dd}_{O(F\cdot n^{2\alpha})}(\phi_\kappa(I,\chi), \phi_\kappa(\pi[I],\chi)) \leq n^{-10}$ whp, as needed for (1).

This concludes the proof of Claim 26.

□

3.6.1 Controlling average φ corruption: proof of Lemma 31

We prove parts 1 and 2 separately. Recall that $\mathcal{B}_I = \Lambda^{\zeta}_{\kappa}(I)$, and let $\mathcal{B}_{\pi[I]} = \Lambda^{\zeta'}_{\kappa}(\pi[I])$. For any $J \in \mathcal{I}$, we define $m_J = \mu_\kappa(J, \mathbf{u})$.

Proof of Lemma 31 (1). Let

$p_\chi = \sum_{J\in\mathcal{B}_I} \phi_\kappa(J,\chi)$, $q_\chi = \sum_{J\in\mathcal{B}_{\pi[I]}} \phi_\kappa(J,\chi)$, $v_\chi = \sum_{J\in\mathcal{B}_I} \varphi^{\zeta}_{\kappa}(J,\chi)$, $u_\chi = \sum_{J\in\mathcal{B}_{\pi[I]}} \varphi^{\zeta'}_{\kappa}(J,\chi)$.

We define vectors $p, q, v, u \in \mathbb{R}^{\nu\setminus\{\mathbf{u},\perp\}}$ correspondingly. Note that since $(I, \mathbf{u})$ is not F-corrupted pair in κ, then $\mathtt{dd}_F(m_I, m_{\pi[I]}) = 0$. We have, using $T^{\mathbf{c}}$ with parameter

$q \geq 11/\epsilon$, for some $\widehat{F} = F^{\Theta(1/\epsilon^2)}$:

$$\begin{aligned}
\mathtt{dd}_{\widehat{F}}(v,u) &= \mathtt{dd}_{\widehat{F}}\left(T^{\mathbf{c}}(m_I \cdot \tfrac{\beta^l}{\zeta^2} \cdot p), T^{\mathbf{c}}(m_{\pi[I]} \cdot \tfrac{\beta^l}{\zeta'^2} \cdot q)\right) \\
&\overset{Claim\ 37}{\leq} \mathtt{dd}_{\widehat{F}^{\Theta(\epsilon)}}\left(m_I \cdot \tfrac{\beta^l}{\zeta^2} \cdot p, m_{\pi[I]} \cdot \tfrac{\beta^l}{\zeta'^2} \cdot q\right) + n^{-10} \\
&\overset{Fact\ 33}{\leq} m_I \cdot \tfrac{\beta^l}{\zeta^2}\, \mathtt{dd}_{\widehat{F}^{\Theta(\epsilon)} \cdot \frac{m_{\pi[I]}}{m_I} \cdot \frac{\zeta^2}{\zeta'^2}}(p,q) + n^{-10} \\
&\leq m_I \cdot \tfrac{\beta^l}{\zeta^2}\, \mathtt{dd}_{F^4}(p,q) + n^{-10},
\end{aligned}$$

where the last inequality holds since $m_I/m_{\pi[I]} \leq F$ and $\frac{\zeta^2}{\zeta'^2} \geq F^{-O(1)}$.

It remains to compute (the expected value of) $\mathtt{dd}_{F^4}(p,q)$, i.e., the difference in ϕ potential. Fix a color (coordinate) χ. If an interval $J \in \mathcal{B}_I$ is matched to some anchor/color (A,χ') pair, and (J,χ') is not *F-corrupted pair*, then, by Claim 26, similar potential will be added to $\pi[J]$ up to distortion factor $O(F \cdot n^{2\alpha}) \leq F^4$: specifically, $\mathtt{dd}_{F^4}(\phi_\kappa(J,\chi), \phi_\kappa(\pi[J],\chi)) \leq \mathtt{dd}_{O(Fn^{2\alpha})}(\phi_\kappa(J,\chi), \phi_\kappa(\pi[J],\chi)) \leq n^{-10}$ whp.

Now focus on corrupted pairs (J,χ') where $J \in \mathcal{B}_I$, $\chi' \in \nu$. Note that $\xi_F^{\kappa'}(\mathcal{B}_I) \leq \frac{2\zeta}{m_I} \cdot \rho_F^{\kappa,\kappa'}(I,\mathcal{B}_I)$; i.e., total $\mu_{\kappa'}$ of (J,χ') with $\mathtt{dd}_F(\mu_{\kappa'}(J,\chi'), \mu_{\kappa'}(\pi[J],\chi')) > 0$ is $O\left(\frac{\zeta}{m_I} \cdot \rho_F^{\kappa,\kappa'}(I,\mathcal{B}_I)\right)$. Also, from Claim 26, the expected potential added for each corrupted pair (J,χ') for an anchor is $O\left(\frac{\mu_{\kappa'}(J,\chi')}{\lambda^t}\right)$. Since we sample λ^t anchor pairs, for each (J,χ') pair, the added potential over all anchors is a r.v. with expectation $O(\mu_{\kappa'}(J,\chi'))$. Overall, we obtain, using Fact 29:

$$\begin{aligned}
\mathbb{E}\left[\mathtt{dd}_{\widehat{F}}(v,u)\right] \leq m_I \cdot \tfrac{\beta^l}{\zeta^2}\, \mathbb{E}\left[\mathtt{dd}_{F^4}(p,q)\right] + n^{-10} &= m_I \cdot \tfrac{\beta^l}{\zeta^2} \cdot O(n^{-8} + \tfrac{\zeta}{m_I}\rho_F^{\kappa,\kappa'}(I,\mathcal{B}_I)) + n^{-10} \\
&= O\left(\tfrac{\beta^l}{\zeta} \cdot \rho_F^{\kappa,\kappa'}(I,\mathcal{B}_I) + n^{-8}\right),
\end{aligned}$$

completing the proof of the first part.

□

In the 2nd part, we need to prove Eqn. (3.8): that $\mathtt{dd}_{\widehat{F}}\left(\varphi_\kappa^\zeta(I,\mathbf{u}),\varphi_\kappa^{\zeta'}(\pi[I],\mathbf{u})\right) = O\left(\frac{\beta^l}{1+\zeta}\cdot\rho+\frac{1}{n^9}\right)$, where $\rho=\rho_F^{\kappa,\kappa'}(I,\mathcal{B}_I^{+\zeta})$. We note that the case $m_I=0$ trivially proves the lemma, and hence below $m_I>0$.

We first introduce a central claim, proven later. Define $a=\theta_\kappa^\zeta(\mathcal{B}_I,\nu\setminus\{\perp\})$, and $b=\theta_\kappa^{\zeta'}(\mathcal{B}_{\pi[I]},\nu\setminus\{\perp\})$.

Claim 40. *Assuming $a=\Omega(\beta^{1.5})$, then, whp, $\mathtt{dd}_{F^{O(1/\epsilon)}}(a,b)=O\left(\frac{k}{m_I}\cdot\frac{1+\zeta}{\beta^l}\cdot\rho+n^{-9}\right)$.*

Proof of Lemma 31 (2), using Claim 40. We have, by definition of $\varphi(I,\mathbf{u})$, a,b, for level $l>0$:

$$\mathtt{dd}_{\widehat{F}}(\varphi_\kappa^\zeta(I,\mathbf{u}),\varphi_\kappa^{\zeta'}(\pi[I],\mathbf{u}))=\mathtt{dd}_{\widehat{F}}\left(T^{\mathbf{u}}(m_I\cdot\tfrac{\beta^{2l}}{\zeta^2}\cdot\min\{\tfrac{a}{k},1\}),T^{\mathbf{u}}(m_{\pi[I]}\cdot\tfrac{\beta^{2l}}{\zeta'^2}\cdot\min\{\tfrac{b}{k},1\})\right). \tag{3.16}$$

Now, suppose $b>k$. Then, as $(I,\mathbf{u})$ is uncorrupted and $\frac{\zeta^2}{\zeta'^2}\geq F^{-O(1)}$:

$$\begin{aligned}\mathtt{dd}_{\widehat{F}}(\varphi_\kappa^\zeta(I,\mathbf{u}),\varphi_\kappa^{\zeta'}(\pi[I],\mathbf{u}))&\leq\mathtt{dd}_{\widehat{F}}\left(T^{\mathbf{u}}(m_I\cdot\tfrac{\beta^{2l}}{\zeta^2}),T^{\mathbf{u}}(m_{\pi[I]}\cdot\tfrac{\beta^{2l}}{\zeta'^2})\right)\\&\overset{\text{Claim 37,Fact 33}}{\leq}\tfrac{\beta^{2l}}{\zeta^2}\,\mathtt{dd}_{\widehat{F}^{\Theta(\epsilon)},\frac{\zeta^2}{\zeta'^2}}(m_I,m_{\pi[I]})+n^{-10}\\&=n^{-10}.\end{aligned}$$

Otherwise, we have:

$$\mathtt{dd}_{\widehat{F}}(\varphi_\kappa^\zeta(I,\mathbf{u}),\varphi_\kappa^{\zeta'}(\pi[I],\mathbf{u}))\overset{\mathtt{dd},T\text{ are monotone}}{\leq}\mathtt{dd}_{\widehat{F}}\left(T^{\mathbf{u}}(m_I\cdot\tfrac{\beta^{2l}}{\zeta^2}\cdot\tfrac{a}{k}),T^{\mathbf{u}}(m_{\pi[I]}\cdot\tfrac{\beta^{2l}}{\zeta'^2}\cdot\tfrac{b}{k})\right).$$

Now, we show a similar derivation for $l=0$, for which we define $\zeta_+=\max\{\zeta,1\},\zeta'_+\triangleq\max\{\zeta',1\}$. In particular, we have using the fact $\beta^l=\zeta_+=\zeta'_+=1$ at level 0, and that $\mathcal{B}_I=\{I\}$ and $\mathcal{B}_{\pi[I]}=\{\pi[I]\}$:

$$\begin{aligned}\mathtt{dd}_{\widehat{F}}(\varphi^0_\kappa(I,\mathbf{u}),\varphi^0_\kappa(\pi[I],\mathbf{u})) &= \mathtt{dd}_{\widehat{F}}\left(T^{\mathbf{u}}(m_I\cdot\tfrac{a}{k}),T^{\mathbf{u}}(m_{\pi[I]}\cdot\tfrac{b}{k})\right)\\ &= \mathtt{dd}_{\widehat{F}}\left(T^{\mathbf{u}}(m_I\cdot\tfrac{\beta^{2l}}{\zeta_+^2}\cdot\tfrac{a}{k}),T^{\mathbf{u}}(m_{\pi[I]}\cdot\tfrac{\beta^{2l}}{\zeta_+'^2}\cdot\tfrac{b}{k})\right).\end{aligned}$$

Note that $T^{\mathbf{u}}=T^{O(1/\epsilon)}_{1/\beta,\Omega_\epsilon(1)}$ transformation zeros-out any $o(1/\beta)$ quantity, and hence if $a=o(k\cdot\beta^{-1})$, the statement is trivial. Therefore, from this point on, we assume $a=\Omega_\epsilon(k\cdot\beta^{-1})=\omega(\beta^{1.5})$. We continue to derive, for some $F'=\widehat{F}^{\Theta(\epsilon)}\cdot\frac{\zeta_+^2}{\zeta_+'^2}\cdot 1/F$:

$$\begin{aligned}\mathtt{dd}_{\widehat{F}}(\varphi^\zeta_\kappa(I,\mathbf{u}),\varphi^{\zeta'}_\kappa(\pi[I],\mathbf{u})) &\le \mathtt{dd}_{\widehat{F}}\left(T^{\mathbf{u}}(m_I\cdot\tfrac{\beta^{2l}}{\zeta_+^2}\cdot\tfrac{a}{k}),T^{\mathbf{u}}(m_{\pi[I]}\cdot\tfrac{\beta^{2l}}{\zeta_+'^2}\cdot\tfrac{b}{k})\right)\\ &\overset{Claim\ 37}{\le} \mathtt{dd}_{\widehat{F}^{\epsilon/11}}\left(m_I\cdot\tfrac{\beta^{2l}}{\zeta_+^2}\tfrac{a}{k},m_{\pi[I]}\cdot\tfrac{\beta^{2l}}{\zeta_+'^2}\tfrac{b}{k}\right)+n^{-10}\\ &\overset{Fact\ 33}{\le} \tfrac{m_I}{k}\cdot\tfrac{\beta^{2l}}{\zeta_+^2}\cdot\mathtt{dd}_{\widehat{F}^{\epsilon/11}\frac{m_{\pi[I]}}{m_I}\frac{\zeta_+^2}{\zeta_+'^2}}(a,b)+n^{-10}\\ &= \tfrac{m_I}{k}\cdot\tfrac{\beta^{2l}}{\zeta_+^2}\,\mathtt{dd}_{F'}(a,b)+n^{-10}\\ &\overset{Claim\ 40}{=} O\left(\tfrac{\beta^l}{\zeta_+}\cdot\rho+n^{-9}\right),\end{aligned}$$

using $F'=F^{O(1/\epsilon)}$ that satisfies Claim 40 and hence $\hat{F}=(F'\frac{\zeta_+'^2}{\zeta_+^2}F)^{\Theta(1/\epsilon)}=F^{\Theta(1/\epsilon^2)}(\zeta_+'/\zeta_+)^{\Theta(1/\epsilon)}=F^{\Theta(1/\epsilon^2)}$.

□

It remains to prove Claim 40. We first establish the following auxiliary claims. First claim argues that for a set $\mathcal{S}$ of uncorrupted intervals I, the density of I in $\mathcal{S}$ (local density) cannot be much larger than the density of $\pi[I]$ in $\pi[\mathcal{S}]$.

Claim 41. *Fix a coloring κ and a color $\chi\in\nu\setminus\{\perp\}$. Fix interval sets $\mathcal{S},\mathcal{S}_\pi\subseteq\mathcal{I}$ such that: (1) $\pi[\mathcal{S}]\subseteq\mathcal{S}_\pi$; and (2) (I,χ) is F-*uncorrupted *for all $I\in\mathcal{S}$. Then, for any $I\in\mathcal{S}$, for all but $20/\alpha$ costs $\hat{c}\in E_{\mathfrak{c}}$ we have $\mathrm{d}_\kappa(I,\chi,\mathcal{S})_{\hat{c}}\le n^\alpha F\cdot\mathrm{d}_\kappa(\pi[I],\chi,\mathcal{S}_\pi)_{\hat{c}}$.*

Proof. For each $I \in \mathcal{S}$, we have $\mathtt{ad}(I, \pi[I]) \leq c$ since (I, χ) is F-*uncorrupted*. Fix some $I \in \mathcal{S}$. By triangle inequality, we have that $\pi[\mathcal{N}_{\hat{c}}(I) \cap \mathcal{S}] \subseteq \mathcal{N}_{\hat{c}+2\mathfrak{c}}(\pi[I]) \cap \mathcal{S}_\pi$, and since all pairs in $\mathcal{S} \times \{\chi\}$ are F-*uncorrupted*, then $\mu_\kappa(\mathcal{N}_{\hat{c}}(I) \cap \mathcal{S}, \chi) \leq F \cdot \mu_\kappa(\pi[\mathcal{N}_{\hat{c}}(I) \cap \mathcal{S}], \chi)$. Overall, we obtain that that, $\mathrm{d}_\kappa(I, \chi, \mathcal{S})_{\hat{c}} \leq F \cdot \mathrm{d}_\kappa(\pi[I], \chi, \mathcal{S}_\pi)_{\hat{c}+2\mathfrak{c}}$. Since there can be at most $20/\alpha$ costs where $\mathrm{d}_\kappa(\pi[I], \chi, \mathcal{S}_\pi)_{\hat{c}+2\mathfrak{c}} > n^\alpha \cdot \mathrm{d}_\kappa(\pi[I], \chi, \mathcal{S}_\pi)_{\hat{c}}$, the claim follows (recall that $\hat{c} + 2\mathfrak{c} < 3\hat{c}$, and the non-zero density range is $[1/n^{10}, n^2]$). □

Second claim shows that for some interval sets $\mathcal{S}$ and any factor $F_d > 1$, the local density of $I \in \mathcal{S}$ in $\mathcal{S}$ is at least a $1/F_d$ fraction of their density with respect to another set of interest $\mathcal{S}'$ (on most costs in $E_\mathfrak{c}$), except for a mass $\approx 1/F_d$-fraction of the mass of $\mathcal{S}'$. This claim will help us both 1) to bound the mass of corrupted pairs where their local density is significantly skewed by corrupted pairs which can potentially generate additional corruption, as well as 2) dealing with the sharp threshold of Line 4 in AssignThetaPotential Algorithm.

Claim 42. *Fix interval sets $\mathcal{S}, \mathcal{S}' \subseteq \mathcal{I}$, color $\chi \in \nu \setminus \{\perp\}$ in a coloring κ and density factor $F_d > 1$. Except for a mass of $O_\epsilon\left(\frac{n^\alpha}{F_d} \cdot \mu_\kappa(\mathcal{S}', \chi)\right)$ of $(I, \chi) \in \mathcal{S} \times \{\chi\}$, for all except $\leq 20/\alpha$ costs $\hat{c} \in E_\mathfrak{c}$, we have $\mathrm{d}_\kappa(I, \chi, \mathcal{S})_{\hat{c}} \geq \frac{1}{F_d} \cdot \mathrm{d}_\kappa(I, \chi, \mathcal{S}')_{\hat{c}}$.*

Proof. Let $\mu_I = \mu_\kappa(I, \chi)$ and $\mu_\mathcal{S} = \mu_\kappa(\mathcal{S}, \chi) = \sum_{I \in \mathcal{S}} \mu_I$. Define a distribution $p_\mathcal{S}$ over $\mathcal{S}$ by setting $p_\mathcal{S}(I) = \frac{\mu_I}{\mu_\mathcal{S}}$. For each $\hat{c} \in E_\mathfrak{c}$, we consider the following expectation $\mathbb{E}_{I \sim p_\mathcal{S}}\left[\frac{\mathrm{d}_\kappa(I, \chi, \mathcal{S}')_{\hat{c}}}{\mathrm{d}_\kappa(I, \chi, \mathcal{S})_{2\hat{c}}}\right]$. By triangle inequality, we have that $\mathrm{d}_\kappa(I, \chi, \mathcal{S})_{2\hat{c}} \geq \max_{J \in \mathcal{N}_{\hat{c}}(I) \cap \mathcal{S}'} \mathrm{d}_\kappa(J, \chi, \mathcal{S})_{\hat{c}}$. Hence, we can estimate the expectation as follows:

$$\sum_{I\in\mathcal{S}} \frac{\mu_I}{\mu_{\mathcal{S}}} \frac{\mathrm{d}_\kappa(I,\chi,\mathcal{S}')_{\hat{c}}}{\mathrm{d}_\kappa(I,\chi,\mathcal{S})_{2\hat{c}}} = \frac{1}{\mu_{\mathcal{S}}} \sum_{I\in\mathcal{S}, J\in\mathcal{S}':\mathtt{ad}(I,J)\le\hat{c}} \frac{\mu_I\cdot\mu_J}{\mathrm{d}_\kappa(I,\chi,\mathcal{S})_{2\hat{c}}}$$
$$\le \frac{1}{\mu_{\mathcal{S}}} \sum_{I\in\mathcal{S}, J\in\mathcal{S}':\mathtt{ad}(I,J)\le\hat{c}} \frac{\mu_I\cdot\mu_J}{\mathrm{d}_\kappa(J,\chi,\mathcal{S})_{\hat{c}}}$$
$$= \frac{1}{\mu_{\mathcal{S}}} \sum_{J\in\mathcal{S}'} \mu_J = \frac{\mu_{\mathcal{S}'}}{\mu_{\mathcal{S}}}.$$

By Markov Inequality, we have that $\Pr_{I\sim p_{\mathcal{S}}}\left[\frac{\mathrm{d}_\kappa(I,\chi,\mathcal{S}')_{\hat{c}}}{\mathrm{d}_\kappa(I,\chi,\mathcal{S})_{2\hat{c}}} > \frac{F_d}{n^\alpha}\right] \le \frac{\mu_{\mathcal{S}'}}{\mu_{\mathcal{S}}}\cdot\frac{n^\alpha}{F_d}$. By union bound over $\hat{c}\in E_{\mathfrak{c}}$, there is a mass of $O_\epsilon(\frac{\mu_{\mathcal{S}'}\cdot n^\alpha}{F_d})$ where the inequality occurs for at least one cost. On the other hand, for intervals where the inequality does not occur for any cost, there can be at most $20/\alpha$ costs where $\frac{\mathrm{d}_\kappa(I,\chi,\mathcal{S})_{2\hat{c}}}{\mathrm{d}_\kappa(I,\chi,\mathcal{S})_{\hat{c}}} > n^\alpha$ (noting that non-zero $\mu(\mathcal{S},\chi)$ values are between $[n^{-10}, n^2]$), and hence for all other costs we have $\mathrm{d}_\kappa(I,\chi,\mathcal{S}')_{\hat{c}} \le F_d\cdot\mathrm{d}_\kappa(I,\chi,\mathcal{S})_{\hat{c}}$ as needed.

□

A corollary is that the sharp threshold of Line 4 in AssignThetaPotential Algorithm cannot affect the θ scores too much. For a pair (J,χ'), we use the notation $\sigma_{J,\chi'} = Q_l\left(T^\theta\left(\Gamma_{l,\zeta}(J,\chi')\right)\right)$ as the quantity in Line 11 of AssignThetaPotential. Note that $\theta^\zeta_\kappa(J,\chi') = m_{\mathcal{V}}((J,\chi'))\cdot\sigma_{J,\chi'}$ unless the threshold of Line 4 passes.

Corollary 43. *Fix a level $l>0$ and $\zeta\in Z_l$. Consider an interval ball $\mathcal{Q}$ with $\mu_\kappa(\mathcal{Q},\mathbf{u}) = O(\zeta)$. Let $\mathcal{Q}^{+\zeta} = \Lambda^\zeta_\kappa(\mathcal{Q})$[12]. Then, $\mathbb{E}_{\mathcal{V}}[\theta^\zeta_\kappa(\mathcal{Q},\nu\setminus\{\perp\})] < (1+o(1))\cdot\frac{k}{\beta^l}\cdot\sum_{(I',\chi')\in\mathcal{Q}^{+\zeta}\times\nu\setminus\{\perp\}}\mu_{\kappa'}(I',\chi')\cdot\sigma_{I',\chi'}$ whp (over relative density algorithm randomness).*

Proof. Let $\theta_{\mathcal{Q}} \triangleq \theta^\zeta_\kappa(\mathcal{Q},\nu\setminus\{\perp\})$. Define the set of colors of large mass in $\mathcal{Q}^{+\zeta}$, where

[12]Recall $\Lambda^\zeta_\kappa(\mathcal{Q}) = \bigcup_{I\in Q}\Lambda^\zeta_\kappa(I)$.

Line 4 of AssignThetaPotential may pass:

$$\mathcal{T} = \{\chi' \in \nu \setminus \{\perp\} \mid \mu_{\kappa'}(\mathcal{Q}^{+\zeta}, \chi') > \lambda \cdot \beta^{l+6}\}.$$

Defining $\bar{\mathcal{T}} = \nu \setminus \mathcal{T} \setminus \{\perp\}$, notice that pairs $(J, \chi') \in \mathcal{Q} \times \bar{\mathcal{T}}$ will not pass the threshold from Line 4 of AssignThetaPotential and their contribution to $\mathbb{E}[\theta_{\mathcal{Q}}]$ is $\mathbb{E}[m_{\mathcal{V}}((J, \chi'))] \cdot \sigma_{J,\chi'} = \frac{k}{\beta^l} \mu_{\kappa'}(J, \chi') \cdot \sigma_{J,\chi'}$.

Now focus on pairs $(J, \chi') \in \mathcal{Q} \times \mathcal{T}$. Here, we split the ball $\mathcal{Q}^{+\zeta}$ into $h = O(1)$ consecutive balls each of $\mathbf{u}$-mass $\leq \zeta$ each: call them $\mathcal{Q}^q$ for $q \in \{1, 2, \ldots, h\}$. By Lemma 30 we have that $\mu_{\kappa'}(\mathcal{I}, \chi') \leq |\mathcal{P}^{\chi'}_{\kappa'}| = \tilde{O}_\epsilon(\frac{n \cdot \lambda}{\lambda^t} \cdot \beta^5)$. For fixed $q \in [h]$, we invoke Claim 42, using $\mathcal{S} = \mathcal{Q}^q$, $\mathcal{S}' = \mathcal{I}$, and $F_d = \frac{n}{\lambda^t \beta^l}$, to obtain that there exists at most $\tilde{O}_\epsilon(n^\alpha \cdot \lambda \cdot \beta^{l+5}) \stackrel{\mathcal{T}}{=}^{\text{def.}} o(\mu_{\kappa'}(\mathcal{Q}^{+\zeta}, \chi'))$ mass of $J \in \mathcal{Q}^q$ where $\mathrm{rd}_{\kappa'}(J, \chi', \mathcal{Q}^q)_{\hat{c}} > \frac{n}{\lambda^t \beta^l}$ on more than $O(1/\alpha)$ costs. Note that when $\mathrm{rd}_{\kappa'}(J, \chi', \mathcal{Q}^q)_{\hat{c}} \leq \frac{n}{\lambda^t \beta^l}$, then also $\mathrm{rd}_{\kappa'}(J, \chi', \Lambda^\zeta_\kappa(J))_{\hat{c}} \leq \frac{n}{\lambda^t \beta^l}$ (by the monotonicity of rd, since for $J \in \mathcal{Q}^q$, $\mathcal{Q}^q \subseteq \Lambda^\zeta_\kappa(J)$).

Hence for all $J \in \mathcal{Q}^{+\zeta}$, except for $O(1) \cdot o(\mu_{\kappa'}(\mathcal{Q}^{+\zeta}, \chi'))$, we have $\mathrm{rd}_{\kappa'}(J, \chi', \Lambda^\zeta_\kappa(J))_{\hat{c}} \leq \frac{n}{\lambda^t \beta^l}$ on all but $O(1/\alpha)$ costs $\hat{c}$. The latter implies that all except for $O(1/\alpha)$ coordinates of the vector Γ will be equal to 1 whp (the $n^{4\alpha}$ overhead dominates any constant factor in the approximation) and the Q_l transformation ignores the $O(1/\alpha)$ lowest entries. Therefore, $\sigma_{J,\chi'} = 1$ whp. This in turn gives us, for any $S \subseteq \mathcal{Q}^{+\zeta}$:

$$\sum_{(J,\chi') \in S \times \mathcal{T}} \mu_{\kappa'}(J, \chi') \cdot \sigma_{J,\chi'} \geq \mu_{\kappa'}(S, \mathcal{T}) - o(\mu_{\kappa'}(\mathcal{Q}^{+\zeta}, \mathcal{T})). \tag{3.17}$$

Now, we combine both cases. Note that by construction, and using the fact $\sigma_{J,\chi'} \leq 1$ for all pairs, we have:

$$\mathbb{E}[\theta_{\mathcal{Q}}] \leq \sum_{(J,\chi') \in \mathcal{Q} \times \bar{\mathcal{T}}} \mathbb{E}[m_{\mathcal{V}}((J, \chi'))] \cdot \sigma_{J,\chi'} + \sum_{(J,\chi') \in \mathcal{Q} \times \mathcal{T}} \mathbb{E}[m_{\mathcal{V}}((J, \chi'))]$$

$$= \frac{k}{\beta^l} \cdot \left(\sum_{(J,\chi') \in \mathcal{Q} \times \bar{\mathcal{T}}} \mu_{\kappa'}(J,\chi') \cdot \sigma_{J,\chi'} + \mu_{\kappa'}(\mathcal{Q},\mathcal{T}) \right).$$

Using Eqn. 3.17, first with $S = \mathcal{Q}$ and then with $S = \mathcal{Q}^{+\zeta}$, we obtain:

$$\mathbb{E}[\theta_{\mathcal{Q}}] \leq \frac{k}{\beta^l} \cdot \left(\sum_{(J,\chi') \in \mathcal{Q} \times \nu \setminus \{\perp\}} \mu_{\kappa'}(J,\chi') \sigma_{J,\chi'} + o\left(\mu_{\kappa'}(\mathcal{Q}^{+\zeta},\mathcal{T})\right) \right)$$

$$\leq \frac{k}{\beta^l} (1 + o(1)) \cdot \left(\sum_{(J,\chi') \in \mathcal{Q}^{+\zeta} \times \nu \setminus \{\perp\}} \mu_{\kappa'}(J,\chi') \sigma_{J,\chi'} \right).$$

This concludes the proof.

□

Finally, we are ready to prove Claim 40, and in particular, show that assuming $a = \Omega(\beta^{1.5})$, then, whp, $\mathtt{dd}_{F^{O(1/\epsilon)}}(a,b) = O\left(\frac{k}{m_I} \cdot \frac{\zeta_+}{\beta^l} \cdot \rho + n^{-9}\right)$ whp.

Proof of Claim 40. Recall that $a = \sum_{(J,\chi') \in \mathcal{V} \cap \mathcal{B}_I \times \nu} \theta^{\zeta}_{\kappa}(J,\chi')$. Since $a = \Omega(\beta^{1.5})$, and is a sum of independent random variables bounded by 1, we have that $a = \Theta(\mathbb{E}[a])$ whp, hence we analyze $\mathbb{E}[a] = \sum_{J \in \mathcal{B}_I, \chi' \neq \perp} \mathbb{E}[m_{\mathcal{V}}((J,\chi'))] \cdot \theta^{\zeta}_{\kappa}(J,\chi')$. The plan is to show that $\mathbb{E}[b] \cdot F^{O(1/\epsilon)} \geq \mathbb{E}[a]$ unless error ρ is sufficiently large. Note that this requires comparing $\theta^{\zeta}_{\kappa}(J,\chi')$ vs $\theta^{\zeta'}_{\kappa}(\pi[J],\chi')$.

As we aim to use Claim 43, we consider $J \in \mathcal{B}_I^{+\zeta}$, where $\mathcal{B}_I^{+\zeta} = \Lambda^{2\zeta}_{\kappa}(I)$.

Consider an *F-uncorrupted* pair $(J,\chi') \in \mathcal{B}_I^{+\zeta} \times \nu \setminus \{\perp\}$; i.e., $\mathtt{dd}_F(\mu_{\kappa'}(J,\chi'), \mu_{\kappa'}(\pi[J],\chi')) = 0$. Let $p = \Gamma_{l,\zeta}(J,\chi')$ and $q = \Gamma_{l,\zeta'}(\pi[J],\chi')$. By the definition of Γ, we have for a cost $\hat{c}$:

$$\frac{p_{\hat{c}}}{q_{\hat{c}}} = O\left(\frac{\mathrm{rd}_{\kappa'}(\pi[J],\chi',\Lambda^{\zeta'}_{\kappa}(\pi[J]))_{\hat{c}}}{\mathrm{rd}_{\kappa'}(J,\chi',\Lambda^{\zeta}_{\kappa}(J))_{\hat{c}}} + 1\right) = O\left(\frac{\mathrm{d}_{\kappa'}(J,\chi',\Lambda^{\zeta}_{\kappa}(J))_{\hat{c}}}{\mathrm{d}_{\kappa'}(\pi[J],\chi',\Lambda^{\zeta'}_{\kappa}(\pi[J]))_{\hat{c}}} \cdot \frac{\mathrm{d}_{\kappa'}(\pi[J],\chi')_{\hat{c}}}{\mathrm{d}_{\kappa'}(J,\chi')_{\hat{c}}} + 1\right),$$

since the two rd quantities are estimated up to a constant-factor, and capped at the same value $\mathrm{rd}_m = n^{4\alpha}\frac{n}{\lambda^t\beta^l}$ whp. For the latter fraction, we note that since $\mathrm{d}_{\kappa'}(J,\chi',\mathcal{S})_{\hat{c}}$ is monotonic in $\hat{c}$ for any fixed $J,\mathcal{S}$, and since (J,χ') is uncorrupted, then by triangle inequality, there are at most $O(1/\alpha)$ costs in $E_{\mathfrak{c}}$ where $\frac{\mathrm{d}_{\kappa'}(\pi[J],\chi')_{\hat{c}}}{\mathrm{d}_{\kappa'}(J,\chi')_{\hat{c}}} \geq n^{\alpha}$.

The ratio $\frac{\mathrm{d}_{\kappa'}(J,\chi',\Lambda^{\zeta}_{\kappa}(J))_{\hat{c}}}{\mathrm{d}_{\kappa'}(\pi[J],\chi',\Lambda^{\zeta'}_{\kappa}(\pi[J]))_{\hat{c}}}$ is more tricky and we analyze it next. Define the set $\mathcal{E} \subseteq \mathcal{B}_I^{+\zeta} \times \nu \setminus \{\perp\}$ to be the set of interval, color pairs (J,χ') which are either corrupted, or where $\frac{\mathrm{d}_{\kappa'}(J,\chi',\Lambda^{\zeta}_{\kappa}(J))_{\hat{c}}}{\mathrm{d}_{\kappa'}(\pi[J],\chi',\Lambda^{\zeta'}_{\kappa}(\pi[J]))_{\hat{c}}} > n^{3\alpha}\cdot F$ for at least $22/\alpha$ distinct costs $\hat{c} \in E_{\mathfrak{c}}$.

For an uncorrupted pair $(J,\chi') \in \{\mathcal{B}_I^{+\zeta} \times \nu \setminus \{\perp\}\} \setminus \mathcal{E}$, we can combine both inequalities above to obtain: there exists at most $O(1/\alpha)$ costs in $E_{\mathfrak{c}}$ where $\frac{p_{\hat{c}}}{q_{\hat{c}}} > n^{4\alpha}\cdot F$. Let $C \subseteq E_{\mathfrak{c}}$ be the set of all other costs.

We now use Corollary 39 with $d = |E_{\mathfrak{c}}| = \frac{1}{\epsilon^3\alpha^2}$ and $v = O(1/\alpha)$, for some $F' = F^{O(1/\epsilon)}$:

$$\begin{aligned}
\mathtt{dd}_{F'}(\sigma_{J,\chi'},\sigma_{\pi[J],\chi'}) &= \mathtt{dd}_{F'}(Q_l(T^{\theta}([p_{\hat{c}}]_{\hat{c}\in E_{\mathfrak{c}}})),Q_l(T^{\theta}([q_{\hat{c}}]_{\hat{c}\in E_{\mathfrak{c}}}))) \\
&\overset{\textit{Corollary 39}}{\leq} \mathtt{dd}_{F'\cdot n^{-O(\alpha)}}([T^{\theta}(p_{\hat{c}})]_{\hat{c}\in C},[T^{\theta}(q_{\hat{c}})]_{\hat{c}\in C}) + n^{-10} \\
&\overset{\textit{Claim 37}}{\leq} \mathtt{dd}_{(F'n^{-O(\alpha)})^{\Omega(\epsilon)}}([p_{\hat{c}}]_{\hat{c}\in C},[q_{\hat{c}}]_{\hat{c}\in C}) + 2\cdot n^{-10} \\
&\leq \mathtt{dd}_{F^2\cdot n^{4\alpha}}([p_{\hat{c}}]_{\hat{c}\in C},[q_{\hat{c}}]_{\hat{c}\in C}) + O(n^{-10}) \\
&= O(n^{-10}),
\end{aligned} \tag{3.18}$$

where last inequality used the lemma assumption that $F \geq n^{\alpha}$, and the last equality used the definition of C above.

Now, for level $l = 0$, we have $\mathbb{E}[a] = k\cdot\left(\sum_{(J,\chi')\in\mathcal{B}_I\times\nu\setminus\{\perp\}}\mu_{\kappa'}(J,\chi')\sigma_{J,\chi'}\right)$. Otherwise, we invoke Corollary 43 for $\mathcal{Q} = \mathcal{B}_I$ to obtain for all levels:

$$\mathbb{E}[a] \le \tfrac{k}{\beta^l}(1+o(1)) \cdot \left(\sum_{(J,\chi') \in \mathcal{B}_I^{+\zeta} \times \nu \setminus \{\perp\}} \mu_{\kappa'}(J,\chi')\sigma_{J,\chi'} \right)$$
$$\le \tfrac{k}{\beta^l}(1+o(1)) \cdot \left(\mu_{\kappa'}(\mathcal{E}) + \sum_{(J,\chi') \in \left(\mathcal{B}_I^{+\zeta} \times \nu \setminus \{\perp\}\right) \setminus \mathcal{E}} \mu_{\kappa'}(J,\chi')\sigma_{J,\chi'} \right).$$

Now, we analyze two cases depending on whether pairs in $\mathcal{E}$ contribute significantly to a or not.

Pairs in $\mathcal{E}$ contribute insignificantly to a. Suppose the second term is the dominant one, i.e.,

$$\mathbb{E}[a] \le 3\tfrac{k}{\beta^l} \cdot \sum_{(J,\chi') \in \left(\mathcal{B}_I^{+\zeta} \times \nu \setminus \{\perp\}\right) \setminus \mathcal{E}} \mu_{\kappa'}(J,\chi')\sigma_{J,\chi'}.$$

Then, for any (J,χ') in the sum, we have $\frac{\mu_{\kappa'}(J,\chi')}{\mu_{\kappa'}(\pi[J],\chi')} \le F$ (as (J,χ') is uncorrupted) and $\frac{\sigma_{J,\chi'}}{\sigma_{\pi[J],\chi'}} \le F^{O(1/\epsilon)}$ unless $\sigma_{J,\chi'} < O(n^{-10})$ (from Eqn. (3.18)). Hence, using the assumption that $F \le \beta^{O(\epsilon)}$:

$$\mathbb{E}[b] = \mathbb{E}\left[\sum_{(J,\chi') \in \mathcal{V} \cap \mathcal{B}_{\pi[I]} \times \nu \setminus \{\perp\}} \theta_\kappa^{\zeta'}(J,\chi') \right] \ge \tfrac{k}{\beta^l} \sum_{(J,\chi') \in \left(\mathcal{B}_{\pi[I]} \times \nu \setminus \{\perp\}\right) \setminus \mathcal{E}} \mu_{\kappa'}(J,\chi')\sigma_{J,\chi'}$$
$$\ge F^{-O(1/\epsilon)}(\mathbb{E}[a] - O(n^{-8})) \ge \beta^{-1} \cdot \Omega(\beta^{1.5}) = \Omega(\beta^{0.5}).$$

Therefore, we have that $b = \Theta(\mathbb{E}[b])$ whp as well, and overall we obtained $\mathtt{dd}_{F^{O(1/\epsilon)}}(a,b) = O(n^{-9})$ whp.

Pairs in $\mathcal{E}$ contribute significantly. Now assume that $\mathbb{E}[a] \le 3\frac{k}{\beta^l} \cdot \mu_{\kappa'}(\mathcal{E})$.

We bound the color mass $\mu_{\kappa'}(\mathcal{E})$. For a color $\chi' \neq \perp$, we define $\mathfrak{C}_{\chi'}$ to be the set of corrupted pairs in $\mathcal{B}_I^{+2\zeta} \times \{\chi'\}$, and $\mathfrak{C} := \cup_{\chi' \in \nu \setminus \{\perp\}} \mathfrak{C}_{\chi'}$. Also let $\mathcal{J}_{\chi'}$ be the interval set

representing the first coordinates of $\mathfrak{C}_{\chi'}$. Like in the proof of part (1) of the Lemma above, we have $\mu_{\kappa'}(\mathfrak{C}) \leq \frac{6\zeta_+}{m_I} \cdot \rho$ where $\rho = \rho_F^{\kappa,\kappa'}(I, \mathcal{B}_I^{+2\zeta})$ and $\zeta_+ = \max\{\zeta, 1\}$. For level $l = 0$, we have $\mathcal{B}_I = \{I\}$ and hence $\mathcal{E} \subseteq \mathfrak{C}$ (when $\mathcal{B}_I = \{I\}$, a pair (I, χ') can be in $\mathcal{E}$ only if it is corrupted, as the other condition in $\mathcal{E}$ implies corruption as well). For $l \geq 1$, let $\mathcal{B}_{\pi[I]}^{-\zeta'/2} = \Lambda_\kappa^{\zeta'/2}(\pi[I])$. We invoke Claim 41 with $\mathcal{S} = \mathcal{B}_I^{+2\zeta} \setminus \mathcal{J}_{\chi'}, \mathcal{S}_\pi = \mathcal{B}_{\pi[I]}^{-\zeta'/2} \supseteq \pi[\mathcal{B}_I^{+2\zeta}]$ (from the precondition of Lemma 31). The claim implies that, for $J \in \mathcal{B}_I^{+\zeta} \setminus \mathcal{J}_{\chi'}$ and hence $\mathcal{B}_{\pi[I]}^{-\zeta'/2} \subseteq \Lambda_\kappa^{\zeta'}(\pi[J])$, we have for all but $20/\alpha$ costs $\hat{c}$:

$$\frac{\mathrm{d}_{\kappa'}(J,\chi',\Lambda_\kappa^\zeta(J)\setminus\mathcal{J}_{\chi'})_{\hat{c}}}{\mathrm{d}_{\kappa'}(\pi[J],\chi',\Lambda_\kappa^{\zeta'}(\pi[J]))_{\hat{c}}} \leq \frac{\mathrm{d}_{\kappa'}(J,\chi',\mathcal{B}_I^{+2\zeta}\setminus\mathcal{J}_{\chi'})_{\hat{c}}}{\mathrm{d}_{\kappa'}(\pi[J],\chi',\mathcal{B}_{\pi[I]}^{-\zeta'/2})_{\hat{c}}} \leq n^\alpha \cdot F.$$

Let $\mathcal{E}_{\chi'}$ be the set of *uncorrupted* pairs $(J, \chi') \in \mathcal{B}_I^{+\zeta} \times \{\chi'\}$ with $\frac{\mathrm{d}_{\kappa'}(J,\chi',\Lambda_\kappa^\zeta(J))_{\hat{c}}}{\mathrm{d}_{\kappa'}(J,\chi',\Lambda_\kappa^\zeta(J)\setminus\mathcal{J}_{\chi'})_{\hat{c}}} > n^{2\alpha}$ on more than $1/\alpha$ costs in $E_{\mathfrak{c}}$. Then, for each uncorrupted pair $(J, \chi') \in \mathcal{B}_I^{+\zeta} \times \{\chi'\} \setminus \mathcal{E}_{\chi'} \setminus \mathfrak{C}_{\chi'}$, except for $\leq 21/\alpha$ costs, we have $\frac{\mathrm{d}_{\kappa'}(J,\chi',\Lambda_\kappa^\zeta(J))_{\hat{c}}}{\mathrm{d}_{\kappa'}(\pi[J],\chi',\Lambda_\kappa^{\zeta'}(\pi[J]))_{\hat{c}}} \leq n^{3\alpha} \cdot F$.

Therefore $\mathcal{E} \subseteq \cup_{\chi'}\mathcal{E}_{\chi'} \cup \mathfrak{C}$. It remains to bound $\mu_{\kappa'}(\cup_{\chi'}\mathcal{E}_{\chi'}) = \sum_{\chi'} \mu_{\kappa'}(\mathcal{E}_{\chi'})$. Partition $\mathcal{B}_I^{+\zeta}$ into 8 balls each of $\mathbf{u}$-mass $\leq \zeta$, called $\mathcal{B}_I^q$ for $q \in \{1, 2, \ldots, 8\}$. Then, for each q and color $\chi' \in \nu$, we invoke Claim 42 with $\mathcal{S} = \mathcal{B}_I^q \setminus \mathcal{J}_{\chi'}$, $\mathcal{S}' = \mathcal{J}_{\chi'}$, and $F_d = n^{2\alpha}/2$, to obtain that there's only a mass of $O_\epsilon(\frac{1}{n^\alpha}\mu_{\kappa'}(\mathfrak{C}_{\chi'}))$ of $(J, \chi') \in \mathcal{B}_I^q \times \{\chi'\} \setminus \mathfrak{C}_{\chi'}$ having more than $1/\alpha$ costs satisfying $\frac{\mathrm{d}_{\kappa'}(J,\chi',\mathcal{J}_{\chi'})_{\hat{c}}}{\mathrm{d}_{\kappa'}(J,\chi',\mathcal{B}_I^q\setminus\mathcal{J}_{\chi'})_{\hat{c}}} > n^{2\alpha}/2$. Now, by definition, we have for each $(J, \chi') \in \mathcal{E}_{\chi'}$ that $\frac{\mathrm{d}_{\kappa'}(J,\chi',\Lambda_\kappa^\zeta(J))_{\hat{c}}}{\mathrm{d}_{\kappa'}(J,\chi',\Lambda_\kappa^\zeta(J)\setminus\mathcal{J}_{\chi'})_{\hat{c}}} > n^{2\alpha}$ on more than $1/\alpha$ distinct costs, implying that $\frac{\mathrm{d}_{\kappa'}(J,\chi',\mathcal{J}_{\chi'})_{\hat{c}}}{\mathrm{d}_{\kappa'}(J,\chi',\Lambda_\kappa^\zeta(J)\setminus\mathcal{J}_{\chi'})_{\hat{c}}} > n^{2\alpha} - 1$, as well as that $\frac{\mathrm{d}_{\kappa'}(J,\chi',\mathcal{J}_{\chi'})_{\hat{c}}}{\mathrm{d}_{\kappa'}(J,\chi',\mathcal{B}_I^q\setminus\mathcal{J}_{\chi'})_{\hat{c}}} > n^{2\alpha}/2$, for q s.t. $J \in \mathcal{B}_I^q$ and hence $\mathcal{B}_I^q \subseteq \Lambda_\kappa^\zeta(J)$. However, as deduced above, such pairs must have mass bounded by $O_\epsilon(\frac{1}{n^\alpha}\mu_{\kappa'}(\mathfrak{C}_{\chi'}))$, implying that $\mu_{\kappa'}(\mathcal{E}_{\chi'}) = o(\mu_{\kappa'}(\mathfrak{C}_{\chi'}))$. Summing over all colors in ν, we obtain that $\mu_{\kappa'}(\mathcal{E}) = (1 + o(1)) \cdot \mu_{\kappa'}(\mathfrak{C}) = O(\frac{\zeta_+}{m_I}\rho)$.

Overall, we conclude:

$$\mathtt{dd}_{F'}(a, b) \leq a \leq O(\mathbb{E}[a]) \leq O\left(\frac{k}{\beta^l} \cdot \frac{\zeta_+}{m_I}\rho\right).$$

This completes the proof of Claim 40. □

3.6.2 Controlling balance of colors: proof of Lemma 32

In this section we analyze the quantity $M_I = \mu_\kappa(I, \nu \setminus \{\mathbf{u}\}) + \sum_{\zeta \in Z_l} \varphi_\kappa^\zeta(I, \nu)$, and in particular, prove that for each level l, $M_I = \Omega_\epsilon(1)$. This will imply that our normalization of $\mu_{\hat{\kappa}}$ introduces only a constant-factor further error.

Most importantly, analyzing M_I requires understanding of the quantity $\varphi_\kappa^\zeta(I, \nu)$. At a high level, we show that one of the following must hold, for $\zeta = \beta^l$ (or $\zeta = 0$ for $l = 0$):

1. $\varphi_\kappa^\zeta(I, \mathbf{u}) = \Omega_\epsilon(1)$, or
2. $\varphi_\kappa^\zeta(I, \nu \setminus \{\mathbf{u}\}) = \Omega_\epsilon(1)$, or,
3. $\mu_\kappa(I, \nu \setminus \{\mathbf{u}\}) = \Omega(1)$.

To do so, we proceed in a few steps. Recall that $\varphi_\kappa^\zeta(I, \nu \setminus \{\mathbf{u}\})$ depends on potential $\phi_\kappa(\Lambda_\kappa^\zeta(I), \nu \setminus \{\mathbf{u}, \perp\})$. First, we show how to bound $\phi_\kappa(\mathcal{S}, \chi)$ for a set $\mathcal{S}$ as a function of $\mathrm{rd}_{\kappa'}(I, *, \mathcal{S})$, for an interval I in the neighborhood of the sampled anchor A (Claim 44). Second, we use this bound to lower bound $\varphi_\kappa^\zeta(I, \nu \setminus \{\mathbf{u}\})$ (which applies $T^{\mathbf{c}}$ to a re-scaling of $\phi_\kappa(\Lambda^\zeta(I), *)$) as a function of $\mathrm{rd}_{\kappa'}(J, \chi', \mathcal{S})$ for $J \in \mathcal{S}$ where $\mathcal{S}$ is $\Lambda^\zeta(I)$ — in particular, if enough such J have rd about $\frac{n}{\beta^l \lambda^t}$ on a non-trivial fraction of costs $c \in E_{\mathbf{c}}$, we get such a lower bound (Claim 45), as long as $\mu_\kappa(I, \mathbf{u})$ is not too small (otherwise, item (3) above holds).

We then focus on $\varphi_\kappa^\zeta(I, \mathbf{u})$ which relies on estimates of $\theta_\kappa^\zeta(\Lambda_\kappa^\zeta(J), \nu \setminus \{\perp\})$. Our third claim bounds the θ quantity for $J \in \mathcal{S}$ as a function of the mass of $I' \in \mathcal{S}, \chi'$

such that $\mathrm{rd}_{\kappa'}(I', \chi', \mathcal{S})_c$ is at most $O^*\left(\frac{n}{\beta^l \lambda^t}\right)$ on some fraction $\approx s_l$ of costs $c \in E_{\mathfrak{c}}$ (Claim 46). Our final fourth claim (Claim 47), uses the previous claim (applied to small ζ balls of the previous level $l-1$) to show that, overall, small $\theta^{\zeta}_{\kappa}(\Lambda^{\zeta}(I))$'s (i.e., insufficient for (1) above to hold) imply that many pairs in I's proximity have the density rd in the right range for enough costs (allowing us to obtain (2) using Claim 45). Finally, we use these four claims to prove Lemma 32.

Throughout this section, each time we refer to sets of costs, we refer to subsets of $E_{\mathfrak{c}}$, where $\mathfrak{c}$ is the base cost of MatchIntervals. Also, when we refer to contribution of a sampled cost $c_i \in E_{\mathfrak{c}}$, we refer to the total contribution of all $c_{i,j} = c_i + \mathfrak{c} \cdot j$ where $j \in \{0, \ldots j_{\max}\}$ where $j_{\max} = O(1/\alpha)$.

In the first claim controlling ϕ, we show that, for a fixed new color χ and chosen anchor (A, χ'), for any ball $\mathcal{S} \subseteq \mathcal{I}$, the total potential added to $\mathcal{S}$ is roughly concentrated.

Claim 44 (Controlling $\phi_\kappa(\mathcal{S}, \chi)$)**.** *Fix an interval set $\mathcal{S} \subseteq \mathcal{I}$ and output color χ. For any interval, color pair (I, χ'), for all but $O(1/\alpha)$ costs $c_i \in E_{\mathfrak{c}}$, and for all $A \in \mathcal{N}_{2c_i}(I)$, we have, with high probability, where $\mathrm{rd}_{c_i} = \mathrm{rd}_{\kappa'}(I, \chi', \mathcal{S})_{c_i}$:*

1. *If (A, χ', c_i) was sampled for color χ, then $\phi_\kappa(\mathcal{S}, \chi) = O(n^{2\alpha}) \cdot \frac{n}{\mathrm{rd}_{c_i} \cdot \lambda^t}$; and*
2. *If (A, χ', c_{i+1}) was sampled for color χ, then $\phi_\kappa(\mathcal{S}, \chi) = \Omega(n^{-2\alpha}) \cdot \frac{n}{\mathrm{rd}_{c_i} \cdot \lambda^t}$.*

Proof. Let $C^{\mathrm{Good}}_{I,\chi'} \subseteq E_{\mathfrak{c}}$ be all c_i costs where:

- $\mathrm{d}_{\kappa'}(I, \chi')_{c_i} \cdot n^{\alpha} \geq \mathrm{d}_{\kappa'}(I, \chi')_{15c_i}$; and
- $\mathrm{d}_{\kappa'}(I, \chi', \mathcal{S})_{c_i} \cdot n^{\alpha} \geq \mathrm{d}_{\kappa'}(I, \chi', \mathcal{S})_{15c_i}$.
- i is not maximal (and hence c_{i+1} may be sampled as well).

We note that by construction, $15c_i < 27c_i = c_{i+3}$ and since $\mu_{\kappa'}(\mathcal{S}, \chi') \in \{0\} \cup [n^{-10}, n]$ for any set $\mathcal{S}$, then there exist at most $66/\alpha + 1$ costs in $E_{\mathfrak{c}} \setminus C^{\text{Good}}_{I,\chi'}$.

We now show the claim holds for all costs in $C^{\text{Good}}_{I,\chi'}$. Fix a sampled anchor, color pair (A, χ') in coloring κ' and a cost $c_i \in C^{\text{Good}}_{I,\chi'}$. For $J \in \mathcal{I}$, denote $d_{J,i,j} = \mathrm{d}_{\kappa'}(J, \chi')_{c_{i,j}}$. By ϕ definition, if (A, χ', c_i) was sampled, we have $\phi_\kappa(\mathcal{S}, \chi) = \sum_j n^{-\alpha j} \cdot m_{i,j}$, where $m_{i,j} = \sum_{J \in \mathcal{N}_{c_{i,j}}(A) \cap \mathcal{S}} \frac{\mu_{\kappa'}(J,\chi')}{\widehat{d_{J,i,j}}} \cdot \frac{2n}{\lambda^t}$ with $c_{i,j} = c_i + j\mathfrak{c}$, and $\widehat{d_{J,i,j}}$ is constant factor approximation to $\max\{d_{J,i,j}, \mathrm{d}_m\}$ for $\mathrm{d}_m = \mathrm{d}_{\kappa'}(A, \chi')_{3c_i} \cdot n^{-\alpha}$ (whp, by Lemma 25). Hence, using the triangle inequality, we have, using $2c_{i,j} \le 3c_i$:

$$\widehat{\mathrm{d}_{J,i,j}} \in [\Theta(n^{-\alpha}), \Theta(1)] \cdot \mathrm{d}_{\kappa'}(A, \chi')_{3c_i} \subseteq [\Theta(n^{-\alpha}) \cdot \mathrm{d}_{\kappa'}(I, \chi')_{c_i}, \Theta(1) \cdot \mathrm{d}_{\kappa'}(I, \chi')_{15c_i}] \subseteq [\Theta(n^{-\alpha}), \Theta(n^{\alpha})] \cdot \mathrm{d}_{\kappa'}($$

and similarly,

$$\widehat{\mathrm{d}_{J,i+1,j}} \in [\Theta(n^{-\alpha}) \cdot \mathrm{d}_{\kappa'}(I, \chi')_{3c_i}, \Theta(1) \cdot \mathrm{d}_{\kappa'}(I, \chi')_{15c_i}] \subseteq [\Theta(n^{-\alpha}), \Theta(n^{\alpha})] \cdot \mathrm{d}_{\kappa'}(I, \chi')_{c_i}.$$

Therefore we have that for each j:

$$\begin{aligned} m_{i,j} &= \sum_{J \in \mathcal{N}_{c_{i,j}}(A) \cap \mathcal{S}} \frac{\mu_{\kappa'}(J,\chi')}{\widehat{d_{J,i,j}}} \cdot \frac{2n}{\lambda^t} \in \frac{[\Theta(n^{-\alpha}), \Theta(n^{\alpha})]}{\mathrm{d}_{\kappa'}(I,\chi')_{c_i}} \cdot \frac{n}{\lambda^t} \cdot \sum_{J \in \mathcal{N}_{c_{i,j}}(A) \cap \mathcal{S}} \mu_{\kappa'}(J, \chi') \\ &= \frac{[\Theta(n^{-\alpha}), \Theta(n^{\alpha})]}{\mathrm{d}_{\kappa'}(I,\chi')_{c_i}} \cdot \frac{n}{\lambda^t} \cdot \mathrm{d}_{\kappa'}(A, \chi', \mathcal{S})_{c_{i,j}} \end{aligned}$$

and by a similar argument, we have that if $c_i \in C^{\text{Good}}_{I,\chi'}$, then $m_{i+1,j} \in \frac{[\Theta(n^{-\alpha}), \Theta(n^{\alpha})]}{\mathrm{d}_{\kappa'}(I,\chi')_{c_i}} \cdot \frac{n}{\lambda^t} \cdot \mathrm{d}_{\kappa'}(A, \chi', \mathcal{S})_{c_{i+1,j}}$ as well.

For (1), we have by triangle inequality and $2c_i + c_{i,j} \le 4c_i$ that $\mathrm{d}_{\kappa'}(A, \chi', \mathcal{S})_{c_{i,j}} \le \mathrm{d}_{\kappa'}(I, \chi', \mathcal{S})_{4c_i} \le n^{\alpha} \cdot \mathrm{d}_{\kappa'}(I, \chi', \mathcal{S})_{c_i}$ and hence $m_{i,j} = \frac{O(n^{2\alpha})}{\mathrm{rd}_{\kappa'}(I,\chi')_{c_i}} \cdot \frac{n}{\lambda^t}$.

For (2), we have by triangle inequality and $2c_i + c_i \leq c_{i+1}$ that $\mathrm{d}_{\kappa'}(A, \chi', \mathcal{S})_{c_{i+1,j}} \geq \mathrm{d}_{\kappa'}(I, \chi', \mathcal{S})_{c_i}$ and hence $m_{i+1,j} = \frac{\Omega(n^{-\alpha})}{\mathrm{rd}_{\kappa'}(I,\chi')_{c_i}} \cdot \frac{n}{\lambda^t}$.

Summing over all j, we obtain the required bounds.

□

The next claim helps control $\varphi^{\zeta}_{\kappa}(I, *_{\neq \mathbf{u}}) = T^{\mathbf{c}}\left(\gamma \cdot \frac{\phi_{\kappa}(\Lambda^{\zeta}_{\kappa}(I),*)}{\beta^l}\right)$ for some parameters γ, ζ. In particular, it shows that if we have a set of pairs $\mathcal{D}$ with each $(I, \chi') \in \mathcal{D}$ of the "right" density within some ball $\mathcal{Q}$, then we add sufficient ϕ mass to $\mathcal{Q}$, which will survive the $T^{\mathbf{c}}$ thresholding whp.

Claim 45 (Controlling $\varphi(I, *_{\neq \mathbf{u}})$)**.** *Fix level $l \geq 0$ and parameters $\delta, \gamma, s \in [\Omega_{\epsilon}(1), 1]$ as well as a set $\mathcal{Q} \subseteq \mathcal{I}$. Consider the (interval, color) set $\mathcal{D} \subseteq \mathcal{Q} \times \nu \setminus \{\perp\}$ of pairs (I, χ') for which there exists a set of $\geq \frac{s}{\eta}$ distinct costs $C^*_{I,\chi'} \subseteq E_{\mathfrak{c}}$, where for each $\hat{c} \in C^*_{I,\chi'}$, we have $\mathrm{rd}_{\kappa'}(I, \chi', \mathcal{Q})_{\hat{c}} \in n^{3\alpha} \cdot \frac{n}{\beta^l \lambda^t} \cdot [1, \beta^2 + \mathbf{1}[l = 0] \cdot n]$. If $\mu_{\kappa'}(\mathcal{D}) \geq \delta \cdot \beta^l$, then, with high probability, we have at step l:*

1. *If $l > 0$, then $\left\| T^{\mathbf{c}}\left(\gamma \cdot \frac{\phi_{\kappa}(\mathcal{Q},*)}{\beta^l}\right)\right\|_1 \geq 0.9\delta\gamma \cdot (s - O(\frac{\eta}{\alpha}))$.*
2. *If $l = 0$, then, $T^{\mathbf{c}}\left(\gamma \cdot \phi_{\kappa}(\mathcal{Q}, \nu)\right) \geq 0.9\delta\gamma \cdot (s - O(\frac{\eta}{\alpha}))$.*

The proof of Claim 45 is deferred to Section 3.6.2.1.

Next, in order to analyze $\varphi(\cdot, \mathbf{u})$, we show an auxiliary claim that bounds $\theta^{\zeta}_{\kappa}(\Lambda^{\zeta}_{\kappa}(J), \nu \setminus \{\perp\})$. This claim will be used for complexity analysis as well. Consider some interval ball $\mathcal{Q}$ at level l containing ζ $\mathbf{u}$-mass in the current coloring κ. Intuitively, this implies there is a mass of sparse pairs proportional to ζ from previous levels. The claim below bounds $\theta^{\zeta}_{\kappa}(\Lambda^{\zeta}_{\kappa}(J), \nu \setminus \{\perp\})$, for all $J \in \mathcal{Q}$ generated at level l, based on the mass of sparse pairs around $\mathcal{Q}$ for the current level.

Claim 46. *Fix a level $l \geq 0$. Fix $\zeta \in Z_l$, and let $s_l = 1 - \frac{(l+1)\cdot\epsilon}{2}$. Consider an interval ball $\mathcal{Q}$ which, for level $l = 0$, contains precisely one interval, and if $l > 0$ satisfies $\mu_\kappa(\mathcal{Q}, \mathbf{u}) \leq 4\zeta$. Let $\mathcal{Q}^{+2\zeta} = \Lambda_\kappa^{2\zeta}(\mathcal{Q})$ and $\mathcal{Q}^{+3\zeta} = \Lambda_\kappa^{3\zeta}(\mathcal{Q})$. Let $\mathcal{D} \subseteq \mathcal{Q} \times \nu \setminus \{\perp\}$ be the set of all $(I, \chi') \in \mathcal{Q} \times \nu \setminus \{\perp\}$ such that $\mathrm{rd}_{\kappa'}(I, \chi', \mathcal{Q})_c \leq n^{3\alpha}\frac{n}{\beta^l \cdot \lambda^t}$ on at least $\frac{s_l}{\eta}$ costs $c \in E_{\mathfrak{c}}$, and let $\mathcal{D}^+ \subseteq \mathcal{Q}^{+2\zeta} \times \nu \setminus \{\perp\}$ be the set of $(I, \chi') \in \mathcal{Q}^{+2\zeta} \times \nu \setminus \{\perp\}$ where $\mathrm{rd}_{\kappa'}(I, \chi', \mathcal{Q}^{+3\zeta})_c \leq n^{3\alpha} \cdot \beta \cdot \frac{n}{\beta^l \cdot \lambda^t}$ on at least $\frac{s_l - \epsilon/4}{\eta}$ distinct costs. Finally, let $\zeta_+ = \max\{\zeta, 1\}$.*

1. *$\theta_\kappa^\zeta(\Lambda_\kappa^\zeta(J), \nu \setminus \{\perp\}) < 1.1k \cdot \max\{\mu_{\kappa'}(\mathcal{D}^+)/\beta^l, \beta^{-2}\}$ for each $J \in \mathcal{Q}$ whp.*
2. *$\theta_\kappa^\zeta(\Lambda_\kappa^\zeta(J), \nu \setminus \{\perp\}) \geq 0.9k \cdot \mu_{\kappa'}(\mathcal{D})/\beta^l$ for each $J \in \mathcal{Q}$ whp, as long as $\mu_{\kappa'}(\mathcal{D}) \geq \beta^{l+1}/k$.*

Proof. Let $\mathcal{Q}^{+\zeta} = \Lambda_\kappa^\zeta(\mathcal{Q})$. Let $m_{\mathcal{Q}} = \theta_\kappa^\zeta(\mathcal{Q}, \nu \setminus \{\perp\})$ and $m_{\mathcal{Q}^{+\zeta}} = \theta_\kappa^\zeta(\mathcal{Q}^{+\zeta}, \nu \setminus \{\perp\})$. Note that $m_{\mathcal{Q}} \leq \theta_\kappa^\zeta(\Lambda_\kappa^\zeta(J), \nu \setminus \{\perp\}) \leq m_{\mathcal{Q}^{+\zeta}}$ for each $J \in \mathcal{Q}$, hence it suffices to bound $m_{\mathcal{Q}}, m_{\mathcal{Q}^{+\zeta}}$. To compute $m_{\mathcal{Q}}, m_{\mathcal{Q}^{+\zeta}}$, we first compute their expectations $\mathbb{E}[m_{\mathcal{Q}}], \mathbb{E}[m_{\mathcal{Q}^{+\zeta}}]$, and then use Chernoff bound to conclude that these variables are roughly bounded by their expectation whp.

We first upper-bound $\mathbb{E}[m_{\mathcal{Q}^{+\zeta}}]$. For level $l = 0$, we have $\mathbb{E}[m_{\mathcal{Q}^{+\zeta}}] = \mathbb{E}[m_{\mathcal{Q}}] = \frac{k}{\beta^l} \cdot \left(\sum_{(J,\chi') \in \mathcal{Q} \times \nu \setminus \{\perp\}} \mu_{\kappa'}(J, \chi')\sigma_{J,\chi'}\right)$. For levels $l > 0$, we use Corollary 43 to obtain

$$\mathbb{E}[m_{\mathcal{Q}^{+\zeta}}] < (1 + o(1)) \cdot \frac{k}{\beta^l} \cdot \sum_{(I,\chi') \in \mathcal{Q}^{+2\zeta} \times \nu \setminus \{\perp\}} \mu_{\kappa'}(I, \chi') \cdot \sigma_{I,\chi'}.$$

Consider a pair $(I, \chi') \in \mathcal{Q}^{+2\zeta} \times \nu \setminus \{\perp\}$. If for a cost c we have $\mathrm{rd}_{\kappa'}(I, \chi', \mathcal{Q}^{+3\zeta})_c > n^{3\alpha} \cdot \beta \cdot \frac{n}{\beta^l \cdot \lambda^t}$, then $\Gamma_{l,\zeta}(I, \chi')_c \leq O(n^\alpha/\beta) < 1/\sqrt{\beta}$, and hence $T^\theta(\Gamma_{l,\zeta}(I, \chi')_c) = 0$ whp. If the above occurs on $\geq \frac{1-(s_l - \epsilon/4)}{\eta}$ distinct costs c — i.e., for $I \notin \mathcal{D}^+$ — then

$\sigma_{I,\chi'} = Q_l\left(T^\theta(\Gamma_{l,\zeta}(I,\chi'))\right) = 0$ (based on Q_l transformation). Hence, $\mathbb{E}[m_{\mathcal{Q}^{+\zeta}}] \leq \mu_{\kappa'}(\mathcal{D}^+) \cdot \frac{k}{\beta^l} \cdot (1+o(1))$ for part 1 (whp over randomness of density estimation).

Similarly, to lower-bound $\mathbb{E}[m_{\mathcal{Q}}]$, we observe that for any $(I,\chi') \in \mathcal{Q}\times\nu\setminus\{\perp\}$ with density $\mathrm{rd}_{\kappa'}(I,\chi',\mathcal{Q})_c \leq n^{3\alpha}\cdot\frac{n}{\beta^l\cdot\lambda^t}$ for a cost c, we have $\Gamma_{l,\zeta}(I,\chi')_c \geq \min\{\Omega(n^\alpha),1\} = 1$ whp. Hence $T^\theta(\Gamma_{l,\zeta}(I,\chi')_c) = 1$, and for (I,χ') where the above occurs on $\geq \frac{s_l}{\eta}$ distinct costs, we have $\sigma_{I,\chi'} = Q_l(T^\theta(\Gamma_{l,\zeta}(I,\chi'))) = 1$. We conclude $\mathbb{E}[m_{\mathcal{Q}}] \geq \frac{k}{\beta^l}\cdot\mu_{\kappa'}(\mathcal{D})$ for part 2.

Finally, note that $m_{\mathcal{Q}}, m_{\mathcal{Q}^{+\zeta}}$ are each sum of independent r.v. in $[0,1]$. Then, by Chernoff bound, $m_{\mathcal{Q}^{+\zeta}} \leq 1.1\max\{\mathbb{E}[m_{\mathcal{Q}^{+\zeta}}], O(\log n)\}$ whp (recall that $k = \tilde{\Theta}(\beta^3)$). Also, $m_{\mathcal{Q}} \geq 0.9\,\mathbb{E}[m_{\mathcal{Q}}]$ whp as long as, say, $\mathbb{E}[m_{\mathcal{Q}}] \geq \beta = \omega(\log n)$. □

Finally, the next claim helps control $\varphi(\cdot,\mathbf{u})$. In particular, it shows that if $\theta^{\beta^l}_\kappa(\Lambda^{\beta^l}_\kappa(I),\nu\setminus\{\perp\})$ is too small to ensure that we **u**-color such interval with sufficient mass, then there must be a sufficiently large mass of pairs in I's proximity of the "right density". The latter property will instead ensure that I is colored with non-**u** color, as we will show later.

Claim 47 (Small θ implies many pairs in I's proximity have right density)**.** *Fix input coloring κ', current coloring κ at a level $l > 0$, and let κ^{-1} be the coloring κ at the previous level $l-1$. Define $b = \min_{I'\in\mathcal{I}}\mu_{\kappa^{-1}}(I',\nu\setminus\{\mathbf{u}\}) + \sum_{\zeta\in Z_{l-1}}\varphi^\zeta_{\kappa^{-1}}(I',\nu)$ (ie, the minimal un-normalized mass of new coloring at level $l-1$); and assume $b = \Omega_\epsilon(1)$. Consider any $I\in\mathcal{I}$ with $\theta^{\beta^l}_\kappa(\Lambda^{\beta^l}_\kappa(I),\nu\setminus\{\perp\}) < \frac{b}{50}\cdot k$. Then, with high probability, there exists a set of interval–color pairs $\mathcal{D}\subseteq\Lambda^{\beta^l}_\kappa(I)\times\nu\setminus\{\perp\}$ with $\mu_{\kappa'}(\mathcal{D}) \geq \frac{b}{25}\cdot\beta^l$ such that every $(J,\chi')\in\mathcal{D}$ pair has relative density $\mathrm{rd}_{\kappa'}(J,\chi',\Lambda^{\beta^l}_\kappa(J))_c \in [1,\beta^2]\cdot n^{3\alpha}\cdot\frac{n}{\beta^l\cdot\lambda^t}$ on $\geq \frac{\epsilon/4}{\eta}$ distinct costs.*

Proof. Let $\mathcal{B}_I = \Lambda_\kappa^{\beta^l}(I)$ and, by the stopping condition, we are guaranteed $\mu_\kappa(\mathcal{B}_I, \mathbf{u}) \geq \beta^l - 1$. For each l, we denote $Z_l^* = Z_l \cap [0, \beta^l \cdot \beta^{0.9}]$.

Since $\mu_\kappa(J, \mathbf{u})$ is monotonically decreasing with level l (AMENDCOLORING can only decrease it), there exists an interval ball $\mathcal{Q} \subseteq \mathcal{B}_I$ with $\mu_\kappa(\mathcal{Q}, \mathbf{u}) \geq \beta^l(1 - o(1))$, such that $\Lambda_{\kappa^{-1}}^{3\zeta}(\mathcal{Q}) \subseteq \mathcal{B}_I$ for any $\zeta \leq \beta^{l-0.1}$ (and hence $\zeta \in Z_{l-1}^*$). Also observe that for $\zeta \in Z_{l-1} \setminus Z_{l-1}^*$, we always have $\varphi_{\kappa^{-1}}^{\zeta}(J, \mathbf{u}) \leq T^{\mathbf{u}}(\frac{\beta^{2(l-1)}}{\zeta^2}) \leq T^{\mathbf{u}}(\beta^{-1.8}) = 0$.

Recall μ recoloring procedure (AMENDCOLORING line 5). Letting $m_J = \mu_{\kappa^{-1}}(J, \mathbf{u})$, $U^{\zeta}(J) = \min\left\{\frac{1}{k} \cdot \theta_{\kappa^{-1}}^{\zeta}(\Lambda_{\kappa^{-1}}^{\zeta}(J), \nu \setminus \{\perp\}), 1\right\}$, and using $\zeta_+ = \max\{\zeta, 1\}$ (as a function of ζ) notation, we have:

$$\beta^l(1-o(1)) \leq \mu_\kappa(\mathcal{Q}, \mathbf{u}) \leq \tfrac{1}{b} \sum_{J \in \mathcal{Q}} m_J \cdot \sum_{\zeta \in Z_{l-1}} \varphi_{\kappa^{-1}}^{\zeta}(J, \mathbf{u}) \leq \tfrac{1}{b} \sum_{J \in \mathcal{Q}} m_J \cdot \sum_{\zeta \in Z_{l-1}^*} T^{\mathbf{u}}(m_J \cdot \tfrac{\beta^{2(l-1)}}{\zeta_+^2} \cdot U^{\zeta}(J)).$$

Now since $m_J, \frac{\beta^{l-1}}{\zeta_+} \leq 1$ and $T^{\mathbf{u}}$ is concave wherever it is > 0, the RHS can be further upper bounded by $\frac{1}{b} \sum_{J \in \mathcal{Q}} m_J \cdot \sum_{\zeta \in Z_{l-1}^*} \frac{\beta^{l-1}}{\zeta_+} \cdot T^{\mathbf{u}}(\frac{\beta^{l-1}}{\zeta_+} \cdot U^{\zeta}(J))$. Now for each radius $\zeta \in Z_{l-1}^*$ from level $l-1$, we decompose $\mathcal{Q}$ into disjoint consecutive interval balls $\mathcal{Q}_i^{\zeta}$ as follows: for level $l = 1$, each $\mathcal{Q}_i^0$ represent the next single interval of $\mathcal{Q}$ starting from the left. For $l > 1$, each ball $\mathcal{Q}_i^{\zeta}$ is a maximal ball containing $\mathbf{u}$-color of mass $\leq \zeta$ at the start of level $l-1$ (i.e. $\mu_{\kappa^{-1}}(\mathcal{Q}_i^{\zeta}, \mathbf{u}) \leq \zeta$ for all i), starting at the first interval on the right of $\mathcal{Q}_{i-1}^{\zeta}$. Also, let W be the set of indices i of the obtained balls, and W' the set of indices i excluding the three smallest and two largest ones.

By switching summation we get:

$$b \cdot \beta^l(1-o(1)) \leq \sum_{\zeta \in Z_{l-1}^*} \tfrac{\beta^{l-1}}{\zeta_+} \sum_{J \in \mathcal{Q}} m_J \cdot T^{\mathbf{u}}(\tfrac{\beta^{l-1}}{\zeta_+} \cdot U^{\zeta}(J)) = \sum_{\zeta \in Z_{l-1}^*} \tfrac{\beta^{l-1}}{\zeta_+} \sum_{i \in W} \sum_{J \in \mathcal{Q}_i^{\zeta}} m_J \cdot T^{\mathbf{u}}(\tfrac{\beta^{l-1}}{\zeta_+} \cdot U^{\zeta}(J)).$$

Notice that since $U^{\zeta}(J) \leq 1$, then for a single i, we have $\sum_{J \in \mathcal{Q}_i^{\zeta}} m_J \cdot T^{\mathbf{u}}(\frac{\beta^{l-1}}{\zeta_+} \cdot U^{\zeta}(J)) \leq \beta^{l-1} = o(b \cdot \beta^l)$, which implies that we can replace W with W' in above: i.e., $b \cdot \beta^l(1 - o(1)) \leq \sum_{\zeta \in Z^*_{l-1}} \frac{\beta^{l-1}}{\zeta_+} \sum_{i \in W'} \sum_{J \in \mathcal{Q}_i^{\zeta}} m_J \cdot T^{\mathbf{u}}(\frac{\beta^{l-1}}{\zeta_+} \cdot U^{\zeta}(J))$ as well. Now, for $i \in W$, let q_i^{ζ} be the $\mu_{\kappa'}$-mass of interval–color pairs $(J, \chi') \in \mathcal{Q}_i^{\zeta} \times \nu \setminus \{\perp\}$ that are relatively sparse: where $\mathrm{rd}_{\kappa'}(J, \chi', \mathcal{B}_I)_c \leq n^{3\alpha}\beta \cdot \frac{n}{\beta^{l-1} \cdot \lambda^t}$ on $\geq \frac{s_{l-1}-\epsilon/4}{\eta}$ distinct costs c. Also let q^* be the total $\mu_{\kappa'}$-mass of relatively sparse pairs in $\mathcal{B}_I$ (using the same definition as above). By the definition of sets $\{\mathcal{Q}_i^{\zeta}\}_{i \in W}$, we have $q^* \geq \sum_{i \in W} q_i^{\zeta}$. We now invoke Claim 46 (1) with $\mathcal{Q} = \mathcal{Q}_i^{\zeta}$, in which case $\mu_{\kappa'}(\mathcal{D}^+) \leq \sum_{j \in [-3,3]} q_{i+j}^{\zeta}$ (noting $\Lambda_{\kappa}^{2\zeta}(\mathcal{Q}_i^{\zeta}) \subseteq \cup_{j \in [-3,3]} \mathcal{Q}_{i+j}^{\zeta}$, and that $\mathrm{rd}_{\kappa'}(J, \chi', \mathcal{B}_I)_c \leq \mathrm{rd}_{\kappa'}(J, \chi', \mathcal{Q}^{+3\zeta})_c)$ — to obtain that, for any $J \in \mathcal{Q}_i^{\zeta}$, we have $U^{\zeta}(J) < 1.1 \cdot \max\left\{\frac{\sum_{j \in [-3,3]} q_{i+j}^{\zeta}}{\beta^{l-1}}, \beta^{-2}\right\}$. Hence $T^{\mathbf{u}}(\frac{\beta^{l-1}}{\zeta_+} \cdot U^{\zeta}(J)) \leq \frac{1.1}{\zeta_+} \sum_{j \in [-3,3]} q_{i+j}^{\zeta}$ (noting that $T^{\mathbf{u}}(x) = 0$ for $x < O(\beta^{-1.5})$). Also, since we defined $\mathcal{Q}_i^{\zeta}$ as containing a-priori $\leq \zeta$ uncolored mass in level $l-1$, then the quantity $\sum_{J \in \mathcal{Q}_i^{\zeta}} m_J \cdot T^{\mathbf{u}}(\frac{\beta^{l-1}}{\zeta_+} \cdot U^{\zeta}(J))$ can be upper bounded by $1.1 \sum_{j \in [-3,3]} q_{i+j}^{\zeta}$. Overall, we obtain:

$$b \cdot \beta^l(1-o(1)) \leq 1.1 \sum_{\zeta \in Z^*_{l-1}} \frac{\beta^{l-1}}{\zeta_+} \sum_{i \in W'} \sum_{j \in [-3,3]} q_{i+j}^{\zeta} \leq 7 \sum_{\zeta \in Z^*_{l-1}} \frac{\beta^{l-1}}{\zeta_+} \sum_{i \in W} q_i^{\zeta} \leq 7q^* \cdot \sum_{\zeta \in Z^*_{l-1}} \frac{\beta^{l-1}}{\zeta_+} \leq 14q^*.$$

Hence we get that $q^* > \frac{b}{16} \cdot \beta^l$, meaning that there exists a set $\mathcal{D}^* \subseteq \mathcal{B}_I \times \nu \setminus \{\perp\}$ with $\mu_{\kappa'}(\mathcal{D}^*) \geq \frac{b}{16} \cdot \beta^l$, such that for all $(J, \chi') \in \mathcal{D}^*$, we have $\mathrm{rd}_{\kappa'}(J, \chi', \mathcal{B}_I)_c \leq n^{3\alpha}\beta^2 \cdot \frac{n}{\beta^l \cdot \lambda^t}$ on $\geq \frac{s_{l-1}-\epsilon/4}{\eta}$ distinct costs c.

On the other hand, let $U_{\kappa}^{\beta^l}(I) := \frac{1}{k} \cdot \theta_{\kappa}^{\beta^l}(\Lambda_{\kappa}^{\beta^l}(I), \nu \setminus \{\perp\})$. From the claim assumption, we have $U_{\kappa}^{\beta^l}(I) \leq \frac{b}{50}$. We apply (the contrapositive of) Claim 46 (2) for level l with $\mathcal{Q} = \mathcal{B}_I$, to obtain that w.h.p, for $\mathcal{D}$ defined as the intervals in $\mathcal{B}_I$ with

relative sparse intervals with respect to level l (i.e. $(J,\chi') \in \mathcal{B}_I \times \nu \setminus \{\perp\}$ pairs where $\mathrm{rd}_{\kappa'}(J,\chi',\mathcal{B}_I)_c \leq n^{3\alpha}\cdot\frac{n}{\beta^l\cdot\lambda^t}$ on $\geq \frac{s_l}{\eta}$ distinct costs), we have $\mu_{\kappa'}(\mathcal{D}) \leq \frac{b}{40}\cdot\beta^l$ (recalling that $b=\Omega_\epsilon(1)$).

The two statements can be combined to conclude that $\mathcal{B}_I \times \nu \setminus \{\perp\}$ contains a set of $\mu_{\kappa'}$-mass of at least $\left(\frac{b}{15}-\frac{b}{40}\right)\beta^l > \frac{b}{25}\cdot\beta^l$ of pairs (J,χ') with relative density $\mathrm{rd}_{\kappa'}(J,\chi',\mathcal{B}_I)_c \in [1,\beta^2]\cdot n^{3\alpha}\cdot\frac{n}{\beta^l\cdot\lambda^t}$ on $\frac{s_{l-1}-\epsilon/4}{\eta}-\frac{s_l}{\eta}=\frac{\epsilon/4}{\eta}$ distinct costs as needed. □

Finally, we prove Lemma 32.

Proof of Lemma 32. We prove by induction on the level l. Fix coloring κ and level l, and let κ^{-1} be coloring κ from the previous level, $l-1$. Consider each $I \in \mathcal{I}$. Let $M_I = \mu_\kappa(I,\nu\setminus\{\mathbf{u}\}) + \sum_{\zeta\in Z_l}\varphi^\zeta_\kappa(I,\nu)$ and let $m_I = \mu_\kappa(I,\mathbf{u})$. Define

$$b_l = \begin{cases} 1 & l=0 \\ \min_{I\in\mathcal{I}}\mu_{\kappa^{-1}}(I,\nu\setminus\{\mathbf{u}\}) + \sum_{\zeta\in Z_{l-1}}\varphi^\zeta_{\kappa^{-1}}(I,\nu) & \text{Otherwise} \end{cases}$$

Our induction hypothesis is $b_l = \Omega(\epsilon)^l$. Since $b_{l+1} = \min_{I\in\mathcal{I}} M_I$, it suffices to prove that $M_I = b_l\cdot\Omega(\epsilon)$ (whp) to complete the induction. First, if $m_I \leq 0.5$, then $M_I \geq 0.5$ (noting $\mu_\kappa(I,\nu)=1$) and the claim is immediate so we assume $m_I > 0.5$.

Base case: $l=0$**.** If there exists a set of colors $\nu^*\subseteq\nu$, s.t. $\mu_{\kappa'}(I,\nu^*)\geq\frac{1}{4}$ where, for each $\chi'\in\nu^*$, we have $\mathrm{rd}_{\kappa'}(I,\chi')_c > n^{3\alpha}\frac{n}{\beta^l\cdot\lambda^t}$ on $\frac{\epsilon/4}{\eta}$ distinct costs, then we can use Claim 45 and obtain that, whp,

$$M_I \geq \varphi^0_\kappa(I,\perp) = T^{\mathbf{c}}(m_I\cdot\phi_\kappa(I,\nu\setminus\{\mathbf{u},\perp\})) \geq T^{\mathbf{c}}(\tfrac{1}{2}\cdot\phi_\kappa(I,\nu\setminus\{\mathbf{u},\perp\})) \geq \tfrac{\epsilon}{32}.$$

Otherwise, we use Claim 46 (2) to obtain $\theta^0_\kappa(I,\nu\setminus\{\perp\})\geq\frac{k}{5}$ and hence, whp,

$$T^{\mathbf{u}}\left(m_I\cdot\tfrac{1}{k}\cdot\theta^0_\kappa(I,\nu\setminus\{\perp\})\right) \geq \tfrac{1}{10},$$

which in turn implies that $M_I \geq \varphi^0_\kappa(I, \mathbf{u}) \geq \frac{1}{10}$.

General case: $l > 0$**.** If $\theta^{\beta^l}_\kappa(\Lambda^{\beta^l}_\kappa(I), \nu \setminus \{\perp\}) \geq \frac{b_l}{50} \cdot k$, then $T^{\mathbf{u}}\left(m_I \cdot \frac{1}{k} \cdot \theta^{\beta^l}_\kappa(\Lambda^{\beta^l}_\kappa(I), \nu \setminus \{\perp\})\right) \geq \frac{1}{100} \cdot b_l$, and hence $M_I \geq \varphi^{\beta^l}_\kappa(I, \mathbf{u}) \geq \min\{\frac{1}{100} b_l, 1\}$. Otherwise, we invoke Claim 47 to obtain there exists a set of pairs $\mathcal{D} \subseteq \Lambda^{\beta^l}_\kappa(I) \times \nu \setminus \{\perp\}$ of mass $\mu_{\kappa'}(\mathcal{D}) \geq \frac{b_l}{25}\beta^l$ where $\mathrm{rd}_{\kappa'}(J, \chi', \Lambda^{\beta^l}_\kappa(I))_c \in [1, \beta^2] \cdot n^{3\alpha}\frac{n}{\beta^l \cdot \lambda^t}$ on $\frac{\epsilon/4}{\eta}$ distinct costs for each $(J, \chi') \in \mathcal{D}$. We now use Claim 45 and obtain that whp,

$$M_I \geq \varphi^{\beta^l}_\kappa(I, \nu \setminus \{\mathbf{u}, \perp\}) \geq \left\| T^{\mathbf{c}}\left(m_I \cdot \frac{\phi_\kappa(\Lambda^{\beta^l}_\kappa(I), *)}{\beta^l}\right)\right\|_1 \geq \left\| T^{\mathbf{c}}\left(\frac{1}{2} \cdot \frac{\phi_\kappa(\Lambda^{\beta^l}_\kappa(I), *)}{\beta^l}\right)\right\|_1 > \frac{\epsilon}{300} \cdot b_l$$

as needed. We conclude that M_I can be lower bounded by the recursive definition,

$$b_{l+1} \geq \min\{b_l \cdot \Omega(\epsilon), 1\} = (\Omega(\epsilon))^{l+1},$$

which completes the proof of the lemma.

□

3.6.2.1 Proof of Claim 45

To prove Claim 45, we wish to apply Claim 44, but we cannot do so directly, as the lower bound we get from there is for a different sample set (since we can only use triangle inequality which gives us a guarantee on a different cost), and hence we show first the following useful statistical fact.

Fact 48. *Let* $\mathbf{X}$ *be a non-negative random variable over some probability space and let* $\mathbf{A}$ *and* $\mathbf{B}$ *be events such that* $\Pr[\mathbf{A}] = \Pr[\mathbf{B}]$*. Fix* $\gamma \in \mathbb{R}$ *such that* $\mathbf{X} \mid \mathbf{B} \geq \gamma$ *(i.e., whenever* $\mathbf{B}$ *happens,* $\mathbf{X} \geq \gamma$*). Then,* $\mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{X} \geq \gamma]] \geq \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A}]]$.

Proof. Since $\Pr[\mathbf{A}] = \Pr[\mathbf{B}]$, then also $\Pr[\mathbf{A} \setminus \mathbf{B}] = \Pr[\mathbf{B} \setminus \mathbf{A}]$. Hence,

$$\begin{aligned}
\mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A}]] &= \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \cap \mathbf{B}]] + \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \setminus \mathbf{B}]] \\
&= \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \cap \mathbf{B}]] + \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \setminus \mathbf{B}] \cdot \mathbf{1}[\mathbf{X} \geq \gamma]] + \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \setminus \mathbf{B}] \cdot \mathbf{1}[\mathbf{X} < \gamma]] \\
&\leq \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \cap \mathbf{B}]] + \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \setminus \mathbf{B}] \cdot \mathbf{1}[\mathbf{X} \geq \gamma]] + \gamma \cdot \Pr[\mathbf{A} \setminus \mathbf{B}] \\
&= \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \cap \mathbf{B}]] + \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \setminus \mathbf{B}] \cdot \mathbf{1}[\mathbf{X} \geq \gamma]] + \gamma \cdot \Pr[\mathbf{B} \setminus \mathbf{A}] \\
&\leq \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \cap \mathbf{B}]] + \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{A} \setminus \mathbf{B}] \cdot \mathbf{1}[\mathbf{X} \geq \gamma]] + \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{B} \setminus \mathbf{A}]] \\
&\leq \mathbb{E}[\mathbf{X} \cdot \mathbf{1}[\mathbf{X} \geq \gamma]]
\end{aligned}$$

□

Proof of Claim 45. Note that every output color χ in output coloring κ is defined via the iid sampled triplet $(A, \chi'', \hat{c}) \sim \mathcal{I} \times \nu \times E_\epsilon$.

Fix $\Delta = \log^2 n$. We define the random variables $\psi_\chi \triangleq \min\{\phi_\kappa(\mathcal{Q}, \chi)/\beta^l, 1/\Delta\}$ and

$$M_\chi \triangleq \begin{cases} T^{\mathbf{c}}(\gamma \cdot \psi_\chi) & l > 0 \\ \gamma \cdot \psi_\chi & l = 0. \end{cases}$$

Note that for $l > 0$, $\left\| T^{\mathbf{c}}\left(\gamma \cdot \frac{\phi(\mathcal{Q},*)}{\beta^l}\right)\right\|_1 \geq \sum_\chi M_\chi$, and for $l = 0$, we have $T^{\mathbf{c}}(x) = x$ for any $x = \Omega_\epsilon(1)$ and hence it suffices to show whp $\sum_\chi M_\chi \geq 0.9\delta\gamma \cdot (s - O(\frac{\eta}{\alpha}))$.

Now since we sample λ^t iid triplets $(A, \chi'', \hat{c})$, the quantity $\sum_\chi M_\chi$ is a sum of λ^t iid r.v. bounded by $\frac{1}{\log^2 n}$. Therefore, by Chernoff bound, we obtain w.h.p.: $\sum_\chi M_\chi \geq 0.9\,\mathbb{E}[\sum_\chi M_\chi] - O_\epsilon\left(\frac{1}{\log n}\right) = 0.9\lambda^t \cdot \mathbb{E}[M_\chi] - O_\epsilon\left(\frac{1}{\log n}\right)$. Therefore, it suffices to show $\mathbb{E}[M_\chi] \geq \delta\gamma \cdot (s - O(\frac{\eta}{\alpha})) \cdot \lambda^{-t}$.

To do that, we introduce another random variable, m_χ, and thereafter show the following inequalities:

- $\mathbb{E}[m_\chi] \geq \delta \cdot \beta^l \cdot (s - O(\frac{\eta}{\alpha})) \cdot \lambda^{-t}$; and,

- $\mathbb{E}[M_\chi] \geq \frac{\gamma}{\beta^l}\,\mathbb{E}[m_\chi]$.

The random variable m_χ. For each $(I,\chi') \in \mathcal{D}$, let $C^{\text{Bad}}_{I,\chi'}$ be union of "bad costs" from Claim 26 (3) and Claim 44 of size $|C^{\text{Bad}}_{I,\chi'}| = O(1/\alpha)$. For a fixed output color χ, define the random variable $z_{I,\chi'}$ as the contribution of (I,χ') to $\phi_\kappa(I,\chi)$ just from costs in $C^*_{I,\chi'} \setminus C^{\text{Bad}}_{I,\chi'}$, and let $m_\chi := \sum_{(I,\chi')\in\mathcal{D}} z_{I,\chi'}$.

The inequality $\mathbb{E}[m_\chi] \geq \delta\cdot\beta^l\cdot(s - O(\frac{\eta}{\alpha}))\cdot\lambda^{-t}$. Here we use Claim 26 (3), which guarantees, for each $(I,\chi')\in\mathcal{D}$: $\mathbb{E}_{(A,\chi'',\hat{c})\sim\mathcal{I}\times\nu\times E_{\mathfrak{c}}}[\phi_\kappa(I,\chi)\cdot\mathbf{1}[\chi'=\chi''] \mid \hat{c}\in E_{\mathfrak{c}}\setminus C^{\text{Bad}}_{I,\chi'}] \geq \mu_{\kappa'}(I,\chi')\cdot\lambda^{-t}$, and hence,

$$\begin{aligned}\mathbb{E}_{(A,\chi'',\hat{c})}[m_\chi] &= \mathbb{E}_{(A,\chi'',\hat{c})}\left[\sum_{(I,\chi')\in\mathcal{D}} z_{I,\chi'}\right]\\ &= \sum_{(I,\chi')\in\mathcal{D}} \mathbb{E}_{(A,\chi'',\hat{c})}[z_{I,\chi'}] \geq \sum_{(I,\chi')\in\mathcal{D}} \Pr_{\hat{c}\in E_{\mathfrak{c}}}[\hat{c}\in C^*_{I,\chi'}\setminus C^{\text{Bad}}_{I,\chi'}]\cdot\mu_{\kappa'}(I,\chi')\cdot\lambda^{-t},\end{aligned}$$

since m_χ only counts contribution for $\hat{c}\in C^*_{I,\chi'}\setminus C^{\text{Bad}}_{I,\chi'}$. Using the fact that $\Pr_{\hat{c}\in E_{\mathfrak{c}}}[\hat{c}\in C^*_{I,\chi'}\setminus C^{\text{Bad}}_{I,\chi'}] \geq (s - O(\eta/\alpha))$ and that $\mu_{\kappa'}(\mathcal{D}) = \delta\cdot\beta^l$, we obtain,

$$\mathbb{E}_{(A,\chi'',\hat{c})\sim\mathcal{I}\times\nu\times E_{\mathfrak{c}}}[m_\chi] \geq \mu_{\kappa'}(\mathcal{D})\cdot(s - O(\tfrac{\eta}{\alpha}))\cdot\lambda^{-t} = \delta\cdot\beta^l\cdot(s - O(\tfrac{\eta}{\alpha}))\cdot\lambda^{-t}.$$

as needed.

The inequality $\mathbb{E}[M_\chi] \geq \frac{\gamma}{\beta^l}\,\mathbb{E}[m_\chi]$. Define the event

$$\mathbf{A}_{\mathcal{D}} \triangleq \left\{\text{for sampled } (A,\chi',c_i) \text{ there exists } (I,\chi')\in\mathcal{D} \text{ s.t. } \mathtt{ad}(I,A)\leq 2c_i \wedge c_i \in C^*_{I,\chi'}\setminus C^{\text{Bad}}_{I,\chi'}\right\}.$$

Note that on one hand, conditioned on *not* $\mathbf{A}_{\mathcal{D}}$, we have $m_\chi = 0$ and hence $m_\chi = m_\chi\cdot\mathbf{1}[\mathbf{A}_{\mathcal{D}}]$. On the other hand, we have $\mathrm{rd}_{\kappa'}(I,\chi',\mathcal{Q})_{\hat{c}} \geq n^{3\alpha}\cdot\frac{n}{\beta^l\lambda^t}$ for any $(I,\chi')\in$

$\mathcal{D}$ and $\hat{c} \in C^*_{I,\chi'}$, and therefore, whenever $\mathbf{A}_{\mathcal{D}}$ happens, by Claim 44, $\phi_\kappa(\mathcal{Q},\chi) \leq O(\beta^l/n^\alpha) < \beta^l/\Delta$. This implies that $\phi_\kappa(\mathcal{Q},\chi)\cdot\mathbf{1}[\mathbf{A}_{\mathcal{D}}] = \psi_\chi \cdot \mathbf{1}[\mathbf{A}_{\mathcal{D}}]\cdot\beta^l$, and overall:

$$\psi_\chi \cdot \mathbf{1}[\mathbf{A}_{\mathcal{D}}]\cdot\beta^l = \phi_\kappa(\mathcal{Q},\chi)\cdot\mathbf{1}[\mathbf{A}_{\mathcal{D}}] \geq m_\chi \cdot \mathbf{1}[\mathbf{A}_{\mathcal{D}}] = m_\chi$$

For $l = 0$, this immediately gives $M_\chi = \gamma\psi_\chi \geq \gamma\psi_\chi \cdot \mathbf{1}[\mathbf{A}_{\mathcal{D}}] \geq \frac{\gamma}{\beta^l} m_\chi$ which implies $\mathbb{E}[M_\chi] \geq \frac{\gamma}{\beta^l}\mathbb{E}[m_\chi]$ as needed.

For $l > 0$, we need to use the upper bound guarantee on the density to "deal with $T^{\mathbf{c}}$ thresholding". For this we first define the event

$$\mathbf{B}_{\mathcal{D}} \triangleq \left\{\text{for sampled } (A,\chi',c_{i+1}) \text{ there exists } (I,\chi')\in\mathcal{D} \text{ s.t. } \mathtt{ad}(I,A) \leq 2c_i \wedge c_i \in C^*_{I,\chi'} \setminus C^{\text{Bad}}_{I,\chi'}\right\}.$$

and notice that $\Pr[\mathbf{B}_{\mathcal{D}}] = \Pr[\mathbf{A}_{\mathcal{D}}]$. Now, since we are guaranteed for any $(I,\chi') \in \mathcal{D}$ and $\hat{c} \in C^*_{I,\chi'}$ that $\mathrm{rd}_{\kappa'}(I,\chi',\mathcal{Q})_{\hat{c}} \leq n^{3\alpha}\cdot\beta^2\cdot\frac{n}{\beta^l\lambda^t}$, then by Claim 44 (2), we have that, whenever $\mathbf{B}_{\mathcal{D}}$ holds, $\psi_\chi \geq \frac{\Omega(1)}{\beta^2 n^{5\alpha}} > \beta^{-3}/\gamma$. We now invoke Fact 48 to obtain $\mathbb{E}[\psi_\chi\cdot\mathbf{1}[\psi_\chi \geq \beta^{-3}/\gamma]] \geq \mathbb{E}[\psi_\chi\cdot\mathbf{1}[\mathbf{A}_{\mathcal{D}}]]$. Since $T^{\mathbf{c}}(x) = x$ whenever $x \geq \beta^{-3}$, we also have that $M_\chi \geq \gamma\psi_\chi\cdot\mathbf{1}[\gamma\psi_\chi \geq \beta^{-3}]$. Overall, we obtain:

$$\mathbb{E}[M_\chi] \geq \mathbb{E}[\gamma\psi_\chi\cdot\mathbf{1}[\gamma\psi_\chi\geq\beta^{-3}]] \geq \gamma\cdot\mathbb{E}[\psi_\chi\cdot\mathbf{1}[\mathbf{A}_{\mathcal{D}}]] \geq \tfrac{\gamma}{\beta^l} m_\chi$$

for $l > 0$ as well.

By combining both inequalities, we conclude:

$$\mathbb{E}[M_\chi] \geq \tfrac{\gamma}{\beta^l}\mathbb{E}[m_\chi] \geq \gamma\delta\cdot(s - O(\tfrac{\eta}{\alpha}))\cdot\lambda^{-t}$$

which implies the claim with high probability as needed.

□

3.7 Runtime complexity of the interval matching algorithm

In this section we analyze the runtime complexity, establishing Theorem 22, item 3.

First we prove the lemma bounding the size of color parts in each step coloring.

3.7.1 Sparsity of colors in colorings: proof of Lemma 30

Recall that Lemma 30 shows that each color $\chi \in \nu \setminus \{\perp\}$ is sparse in each coloring κ'. Specifically, recalling that $\mathcal{P}^{\chi'}_{\kappa'}$ is the set of intervals I with $\mu_{\kappa'}(I,\chi') > 0$, we need to prove that $|\mathcal{P}^{\chi'}_{\kappa'}| \leq O_\epsilon(\beta^5\lambda)\cdot\frac{n}{\lambda^t}$.

Proof of Lemma 30. Consider a color $\chi \in \nu \setminus \{\mathbf{u}, \perp\}$ in coloring κ. For each level l and for each j, we add $O\left(\frac{n}{\lambda^t}\cdot\frac{\mu_{\kappa'}(I,\chi')}{\widehat{d_{I,j}}}\right) = O\left(n^\alpha\cdot\frac{n}{\lambda^t}\cdot\frac{\mu_{\kappa'}(I,\chi')}{d_{A,j}}\right)$ ϕ-potential to each $(I,\chi') \in \mathcal{N}^{\kappa'}_{c_{i,j}}(A)$, hence for level l we have $\phi(\mathcal{I},\chi) = O_\epsilon\left(n^\alpha\cdot\frac{n}{\lambda^t}\right)$. We use the bound on $\phi_\kappa(\mathcal{I},\chi)$ to bound the total number of intervals which "survive the $T^{\mathbf{c}}$ transformation" (i.e., have φ potential still non-zero). First we derive, for $l \geq 1$:

$$\sum_{I\in\mathcal{I}}\sum_{\zeta\in Z_l}\frac{\beta^{2l}}{\zeta^2}\frac{\mu_\kappa(I,\mathbf{u})\cdot\phi_\kappa(\Lambda^\zeta_\kappa(I),\chi)}{\beta^l} \leq \sum_{\zeta\in Z_l}\frac{\beta^l}{\zeta}\sum_{I\in\mathcal{I}}\frac{\mu_\kappa(I,\mathbf{u})}{\zeta}\phi_\kappa(\Lambda^\zeta_\kappa(I),\chi) \overset{Claim\ 35}{\leq} \sum_{\zeta\in Z_l}\frac{\beta^l}{\zeta}2\phi_\kappa(\mathcal{I},\chi) = O_\epsilon(n^\alpha\cdot\tfrac{n}{\lambda^t}). \tag{3.19}$$

Now we have that $\sum_{\zeta\in Z_l}\varphi^\zeta_\kappa(\mathcal{I},\chi) = \sum_{I\in\mathcal{I}}\sum_{\zeta\in Z_l}T^{\mathbf{c}}(\frac{\beta^{2l}}{\zeta^2}\frac{\mu_\kappa(I,\mathbf{u})\cdot\phi_\kappa(\Lambda^\zeta_\kappa(I),\chi)}{\beta^l})$. Since $T^{\mathbf{c}}$ zeros out any input smaller than $1/\beta^4$, we have that at most $O_\epsilon(n\beta^5/\lambda^t)$ intervals I have $\sum_{\zeta\in Z_l}\varphi^\zeta_\kappa(\mathcal{I},\chi) > 0$. Finally, one can observe $I \in \mathcal{P}^{\chi'}_{\kappa'}$ only if $\varphi_{\kappa'}(I,\chi') > 0$ at some level of the previous step $t-1$, hence the bound follows.

It remains to show for the color $\chi = \mathbf{u}$. For this task, let l' be such that $\beta^{l'} \in [1,\beta)\cdot n^{3\alpha}\cdot\beta\cdot\frac{n}{\lambda^t}$. We consider two cases.

First, suppose the algorithm reaches level l' and $\mu_\kappa(\mathcal{X}, \mathbf{u}), \mu_\kappa(\mathcal{Y}, \mathbf{u}) \geq \beta^{l'}$. We now assume (by induction) Lemma 30 holds for κ' and invoke Claim 46. Then we have that each $I \in \mathcal{I}$, $T^{\mathbf{u}}(\frac{1}{k} \cdot \theta_\kappa^\zeta(\Lambda_\kappa^\zeta(I), \nu \setminus \{\perp\}))$ is at most proportional to the color mass where $\mathrm{rd}_\kappa(I, \chi', \mathcal{Q}^{+3\zeta})_c \leq n^{3\alpha} \cdot \frac{n\beta}{\beta^{l'}\lambda^t}$ for some set $\mathcal{Q}$, on some of the costs. However, since, by the definition of relative density it must be at least 1, then $\mathrm{rd}_\kappa(I, \chi', \mathcal{Q}^{+3\zeta})_c \geq 1$ and hence we obtain $T^{\mathbf{u}}(\frac{1}{k} \cdot \frac{\beta^{2l}}{\zeta^2} \cdot \theta_\kappa^\zeta(\Lambda_\kappa^\zeta(I), \nu \setminus \{\perp\})) \leq T^{\mathbf{u}}(\beta^{-1.5} \cdot \frac{1}{k} \cdot \frac{\beta^{2l}}{\zeta^2} \cdot 1.1k\frac{\zeta}{\beta^l}) \leq T^{\mathbf{u}}(1.1\beta^{-1.5}) = 0$ for all $I \in \mathcal{I}$ (note that Claim 46 requires $\delta \geq \beta^{-1.5}$). This means that all $\varphi_\kappa^\zeta(\mathcal{I}, \mathbf{u}) = 0$ and $\mathbf{u}$ will not appear on any interval.

Now, consider the opposite: the algorithm breaks at level $l^* \leq l'$, meaning that $\mu_\kappa(\mathcal{X}, \mathbf{u})$ or $\mu_\kappa(\mathcal{Y}, \mathbf{u})$ is less than $\beta^{l^*} \leq \beta^{l'}$. Assume w.l.o.g $\mu_\kappa(\mathcal{X}, \mathbf{u}) < \beta^{l'}$; this implies $\mu_\kappa(\mathcal{X}, \mathbf{u}) < \frac{n\beta^2 \cdot n^{3\alpha}}{\lambda^t}$. Next, notice that if $\frac{\mu_\kappa(\mathcal{Y}, \mathbf{u})}{\mu_\kappa(\mathcal{X}, \mathbf{u})} > \beta$, then ADJUST-$\mathbf{u}$ zeros out all of $\mathbf{u}$ mass. Otherwise, we have that $\mu_\kappa(\mathcal{I}, \mathbf{u}) = O(\frac{n\beta^4}{\lambda^t})$ before calling to ADJUST-$\mathbf{u}$. Notice that ADJUST-$\mathbf{u}$ omits $\mathbf{u}$ on any palette where $\mathbf{u}$ is of mass at most $\Omega_\epsilon(\beta^{-1})$ (by the $T^{\mathbf{u}}$ transformation), hence we conclude color $\mathbf{u}$ is part of $O_\epsilon(\frac{n\beta^5}{\lambda^t})$ palettes as well at the end of step t for each $\kappa \in K$. Hence in the next step $\hat{t} = t + 1$, we get $|\mathcal{P}_{\kappa'}^{\chi'}| = \tilde{O}_\epsilon(\frac{n\beta^5}{\lambda^{\hat{t}-1}})$ for all $\chi' \in \nu \setminus \{\perp\}$ as needed.

□

3.7.2 Algorithm runtime

We now proceed to the main runtime analysis of the interval matching algorithm.

We define $S_{\kappa'} = \|\mu_{\kappa'}(*, *)\|_0$, i.e., the count of non-zero-mass pairs $(I, \chi') \in \mathcal{I} \times \nu$. We note that from Lemma 30, we have $S_{\kappa'} = n \cdot \tilde{O}_\epsilon(\beta^{O(1)})$. The same Lemma also implies that we converge in $1/\epsilon + O(1)$ steps, since for $t > 1/\epsilon + O(1)$, all $\nu \setminus \{\perp\}$ colors must be non-zero in 0 intervals, in which case we stop iterating.

Before proceeding to the proof, we state a couple theorems for fast evaluation of potentials; they are all proved in Section 3.9 (and can be obtained from known data structures). In particular, for sampling anchors and pivots, we use the following data-structure.

Theorem 49. *Given a set of numbers $a_1, \ldots a_n \geq 0$ in a sparse representation (i.e., as a set S of i's with $a_i > 0$) there exists a data structure D supporting the following queries:*

- *Sample index $i \in [n]$ from the distribution $\{a_i / \sum_j a_j\}_i$ in time $O(\log n)$;*
- *Given $k \geq 1$, produce a set $S \subseteq [n]$ that includes each i with probability $\min\{ka_i, 1\}$ independently. The runtime is $O((1 + k \cdot \sum_{i=1}^{n} a_i) \log n)$ in expectation.*
- *Given an interval $[s, t] \subset [n]$, and $k \geq 1$, produce a set $S \subseteq [n]$ that includes each $i \in [s, t]$ with probability $\min\{ka_i, 1\}$ independently. The runtime is $O((1 + k \cdot \sum_{i=s}^{t} a_i) \log^2 n)$ in expectation.*

Furthermore, the preprocessing time is $O(|S| \cdot \log^{O(1)} n)$.

For fast calculation of equations (3.3) and (3.4), we need data-structures that generate for each $I \in \mathcal{I}$ the sums of only the relevant ϕ, θ colors over the Λ balls (i.e., which pass the threshold). We use the following data-structure and algorithm:

Theorem 50. *Given a set of numbers $a_1, \ldots a_n \geq 0$ in a sparse representation (i.e., as a set S of i's with $a_i > 0$), there exists a data structure D supporting the following queries:*

- *Given some $i < j$ compute $\sum_{k=i}^{j} a_k$. The query complexity is $O(\log n)$.*
- *Given some i and $\zeta > 0$, compute $j \in S, j \leq i$ such that $\sum_{k=j-1}^{i} a_k < \zeta \leq \sum_{k=j}^{i} a_k$ (or output that none exists). The query complexity is $O(\log n)$.*

The preprocessing time is $O(|S| \log n)$.

Theorem 51. *Suppose we are given an $m \times n$ matrix A, where $m \leq n$, given in the sparse form, i.e., as a set S of non-zero entries. Also, suppose we are given n intervals $[s_i, t_i] \subseteq [n]$ such that both $\{s_i\}_i$ and $\{t_i\}_i$ are non-decreasing. Given $\gamma > 0$, we can find all pairs $(i, \chi) \in [n] \times [m]$ such that $\sum_{k=s_i}^{t_i} A_{\chi,k} \geq \gamma$. The runtime is $\tilde{O}\left(|S| + n + \frac{1}{\gamma} \cdot \sum_{\chi,i} \sum_{k=s_i}^{t_i} A_{\chi,k}\right)$.*

Theorem 50 helps us calculating Λ_κ^ζ for each interval, and generating $\varphi_\kappa^\zeta(I, \mathbf{u})$ while Theorem 51 is used for $\varphi_\kappa^\zeta(I, *_{\neq \mathbf{u}})$. We prove these theorems in section 3.9.

We proceed to proving Theorem 22, item 3 next.

Lemma 52. *Clustering and assigning ϕ_κ takes, by expectation, $T_{\mathtt{ad}} \cdot \tilde{O}(n \cdot \lambda \cdot \beta^{O(1)})$ time in each step t and level l.*

Proof. Fix a step t and level l. We claim the following for the clustering and potential ϕ computation (in Alg. 8):

1. We call ClusterAnchor (Alg. 4) at most λ^t times. This is immediate from the fact each anchor can be clustered at most once.
2. ClusterAnchor takes $T_{\mathtt{ad}} \cdot n \cdot \tilde{O}(\lambda^{1-t} \cdot \beta^{O(1)})$ time. Indeed, notice that we call each ClusterAnchor with $|\mathcal{R}| = |\mathcal{P}_{\kappa'}^{\chi'}| = n \cdot \tilde{O}_\epsilon(\lambda^{1-t} \cdot \beta^5)$ (by Lemma 30), and we perform a single $\mathtt{ad}$ calculation per interval $I \in \mathcal{R}$.

3. We call AssignPhiPotential (Alg. 5) at most $\frac{1}{\alpha} \cdot \lambda^t$ times. Indeed, observe each call to ClusterAnchor outputs $O(1/\alpha)$ clusters, and we perform AssignPhiPotential once for each output cluster.

4. AssignPhiPotential takes $n \cdot \tilde{O}(\lambda^{1-t} \cdot \beta^{O(1)})$ time in expectation. We argue this below.

To analyze the time complexity of AssignPhiPotential, we need to bound time spent estimating densities. For this task, notice that for each $I \in \mathcal{A}$, we estimate $\mathrm{d}_{\kappa'}(I, \chi')_{\hat{c}}$ using $\mathrm{d}_m = d_{\mathcal{A}} \cdot n^{-\alpha} \geq n^{-\alpha} \mu_{\kappa'}(\mathcal{A}, \chi')$. Hence the total complexity $T_{\mathrm{APP}}(\mathcal{A}) = T_{\mathtt{ad}} \cdot \tilde{O}(|\mathcal{A}| \cdot \frac{\mu_{\kappa'}(\mathcal{I}, \chi')}{d_{\mathcal{A}}} + |\mathcal{A}|)$. The term $T_{\mathtt{ad}} \cdot \tilde{O}(|\mathcal{A}|)$ term is dominated by the complexity of ClusterAnchor from above, hence we only need to bound:

$$T_{\mathtt{ad}} \cdot \tilde{O}\left(|\mathcal{A}| \cdot \tfrac{\mu_{\kappa'}(\mathcal{I}, \chi')}{d_{\mathcal{A}}}\right) = T_{\mathtt{ad}} \cdot \tfrac{|\mathcal{A}|}{d_{\mathcal{A}}} \cdot n \cdot \tilde{O}\left(\lambda^{1-t} \cdot \beta^{O(1)}\right).$$

It remains to bound $\mathbb{E}\left[\frac{|\mathcal{A}|}{d_{\mathcal{A}}}\right]$. Fix $\chi \in [\lambda^t]$ for which we sampled an anchor triplet (A, χ'', c_i). Now, fix $c_i \in E_c$ which was sampled as part of that triplet. For $(I, \chi') \in \mathcal{I} \times \nu \setminus \{\perp\}$, define $Z_{I,\chi'}$ as the (random) event that (I, χ') is clustered in some $\mathcal{A} \in \mathcal{C}_\chi$:

$$Z_{I,\chi'} = \left[A \in \mathcal{N}_{c_{i,j_{\max}}}(I) \cap \mathcal{P}_{\kappa'}^{\chi'} \wedge \chi'' = \chi'\right]$$

Also define $p_{I,\chi'} = \Pr[Z_{I,\chi'}]$ and $\tau_{I,\chi'} = \max_{\mathcal{A} \in \mathcal{C}_\chi} \frac{1}{d_{\mathcal{A}}} \cdot \mathbb{1}[Z_{I,\chi'}]$. Notice that $p_{I,\chi'} \leq \frac{d_{I,j_{\max}}}{2n}$, where $d_{I,j_{\max}} = \mathrm{d}_{\kappa'}(I, \chi', \mathcal{I})_{c_{i,j_{\max}}}$. Also note that we have $d_{\mathcal{A}} \geq n^{-\alpha} \cdot d_{I,j_{\max}}$ (as $\mathcal{N}_{c_{i,j_{\max}}}(I) \subseteq \mathcal{N}_{3c_i}(A)$), and hence $\tau_{I,\chi'} \leq \frac{n^\alpha}{d_{I,j_{\max}}}$. We can finally bound the expectation:

$$\mathbb{E}\left[\frac{|\mathcal{A}|}{d_{\mathcal{A}}}\right] \leq \sum_{\substack{(I,\chi')\in\mathcal{I}\times\nu\setminus\{\perp\}\\ \mu_{\kappa'}(I,\chi')>0}} \frac{p_{I,\chi'}}{d_{\mathcal{A}}} \leq \sum_{\substack{(I,\chi')\in\mathcal{I}\times\nu\setminus\{\perp\}\\ \mu_{\kappa'}(I,\chi')>0}} p_{I,\chi'}\cdot\tau_{I,\chi'} \leq \frac{n^{\alpha}}{2n} \sum_{\substack{(I,\chi')\in\mathcal{I}\times\nu\setminus\{\perp\}\\ \mu_{\kappa'}(I,\chi')>0}} \frac{d_{I,j\max}}{d_{I,j\max}} = \frac{n^{\alpha}}{2n} S_{\kappa'} = \beta^{O(1)}.$$

We conclude the expected time spent on AssignPhiPotential for each cluster is $T_{\mathtt{ad}}\cdot\beta^{O(1)}\cdot n\cdot\tilde{O}(\lambda^{1-t}\cdot\beta^{O(1)}) = T_{\mathtt{ad}}\cdot n\cdot\tilde{O}(\lambda^{1-t}\cdot\beta^{O(1)})$ as needed.

To summarize the time complexity of clustering, we have for a fixed step t and level l:

- Total time spent on all calls to ClusterAnchor is $\lambda^t\cdot T_{\mathtt{ad}}\cdot n\cdot\tilde{O}(\lambda^{1-t}\cdot\beta^{O(1)}) = T_{\mathtt{ad}}\cdot n\cdot\tilde{O}(\lambda\beta^{O(1)}) = T_{\mathtt{ad}}\cdot n^{1+O(\epsilon)}$.
- Expected total time spent on all calls to AssignPhiPotential is $\frac{1}{\alpha}\cdot\lambda^t\cdot T_{\mathtt{ad}}\cdot n\cdot\tilde{O}(\lambda^{1-t}\cdot\beta^{O(1)}) = T_{\mathtt{ad}}\cdot n\cdot\tilde{O}(\lambda\beta^{O(1)}) = n^{1+O(\epsilon)}$.

□

Second, we show that computing θ's for the pivots is also time efficient.

Lemma 53. *The algorithm* AssignThetaPotential *(Alg. 6) takes* $T_{\mathtt{ad}}\cdot n\cdot\tilde{O}(\lambda\beta^{O(1)})$ *time in each level* l *(whp).*

Proof. Fix step t and level l. We notice that the main runtime term to estimate is the time spend on approximating the relative densities. For this, we note:

1. We only estimate densities for pairs (V,χ') when $\mu_{\kappa'}(\Lambda^{\zeta}_{\kappa}(V),\chi') \leq \lambda\cdot\beta^{l+O(1)}$ (from Line 4 threshold in AssignThetaPotential).
2. We sample each $(I',\chi')\in\mathcal{I}\times\nu$, k times, each with probability $p_{I',\chi'} := \beta^{-l}\cdot\mu_{\kappa'}(I',\chi')$, and spend $\tau_{I',\chi'} := \tilde{O}\left(\frac{1}{\mathrm{rd}_m}\cdot\frac{\mu_{\kappa'}(\mathcal{I},\chi')}{\mu_{\kappa'}(I',\chi')} + \frac{\mu_{\kappa'}(\Lambda^{\zeta}_{\kappa}(V),\chi')}{\mu_{\kappa'}(I',\chi')}\right)$ time approximating such pair (by Lemma 25), where $\mathrm{rd}_m = n^{3\alpha}\cdot\frac{n}{\lambda^t\beta^l}$.

Combining the above, we get $\tau_{I',\chi'} = T_{\mathtt{ad}} \cdot \tilde{O}\left(\frac{\lambda^t\beta^l}{n} \cdot \frac{n\cdot\lambda\cdot\beta^{O(1)}}{\lambda^t} \cdot \frac{1}{\mu_{\kappa'}(I',\chi')} + \frac{\lambda\beta^{l+O(1)}}{\mu_{\kappa'}(I',\chi')}\right) = T_{\mathtt{ad}} \cdot \tilde{O}\left(\frac{\lambda\beta^{l+O(1)}}{\mu_{\kappa'}(I',\chi')}\right)$.

Hence, the expected run-time of all pair approximation is

$$\begin{aligned}
\mathbb{E}_{\mathcal{V}}\left[\text{Density approximation time}\right] &= \mathbb{E}_{\mathcal{V}}\left[\sum_{(I',\chi')\in\mathcal{V}} \tau_{I',\chi'}\right] \\
&\leq \sum_{(I',\chi')\in\mathcal{I}\times\nu:p_{I',\chi'}>0} k \cdot p_{I',\chi'} \cdot \tau_{I',\chi'} \\
&= k \cdot \sum_{(I',\chi')\in\mathcal{I}\times\nu:p_{I',\chi'}>0} \beta^{-l} \cdot \mu_{\kappa'}(I',\chi') \cdot \tilde{O}\left(\frac{\lambda\beta^{l+O(1)}}{\mu_{\kappa'}(I',\chi')}\right) \\
&= k \cdot \sum_{(I',\chi')\in\mathcal{I}\times\nu:p_{I',\chi'}>0} \tilde{O}\left(\lambda\cdot\beta^{O(1)}\right) \\
&= k \cdot S_{\kappa'} \cdot \tilde{O}\left(\beta^{O(1)}\right) = \tilde{O}\left(n\cdot\lambda\cdot\beta^{O(1)}\right).
\end{aligned}$$

Since the expectation is over sum of independent r.v., bounded by $O(n)$, we also have the bound whp.

□

Last, we show that assigning colors using ϕ, θ scores is efficient:

Lemma 54. *Assigning colors in* AmendColoring *(Alg. 7) takes* $\tilde{O}_\epsilon(n\cdot\beta^{O(1)})$ *time in each level* l.

Proof. Let $S_{\mathbf{u}} = \|\mu_\kappa(*,\mathbf{u})\|_0$, $S_\theta = \|\theta_\kappa(*,*)\|_0$, $S_\phi = \|\phi_\kappa(*,*)\|_0$. We note that $S_{\mathbf{u}} \leq 2n$ and $S_\theta, S_\phi \leq S_{\kappa'} = n\cdot\tilde{O}_\epsilon(\beta^{O(1)})$.

First, from Theorem 50, the pre-processing time of the data structures is $\tilde{O}\left(S_{\mathbf{u}} + \tilde{O}(S_\theta)\right) = \tilde{O}_\epsilon(n\cdot\beta^{O(1)})$, and since we query each data structure $O(n)$ times (once for each $I\in\mathcal{I}$), the total query time is $\tilde{O}(1)\cdot O(n) = \tilde{O}(n)$.

Second, focusing on calculating $\varphi_\kappa^\zeta(*, *_{\neq \mathbf{u}})$, define $L_\phi = \sum_{I \in \mathcal{I}} \sum_{\zeta \in Z_l} \frac{\beta^{2l}}{\max\{\zeta^2,1\}} \frac{\mu_\kappa(I,\mathbf{u}) \cdot \phi_\kappa(\Lambda_\kappa^\zeta(I),\nu)}{\beta^l}$, and recall from Eqn. (3.19), we have $L_\phi = \lambda^t \cdot O_\epsilon(n^{2\alpha} \cdot \frac{n}{\lambda^t}) = O_\epsilon(n \cdot \beta)$. We invoke Theorem 51 using $\gamma = \beta^{-4}$, to obtain the time spent on calculating $\varphi_\kappa^\zeta(*, *_{\neq \mathbf{u}})$ is $\tilde{O}\left(S_\phi + n + \frac{L_\phi}{\beta^{-4}}\right) = \tilde{O}_\epsilon(n \cdot \beta^{O(1)})$.

Last, each update rule of $\mu_\kappa(I)$ using φs takes $\tilde{O}(1)$ time, which sums up to $\tilde{O}(n)$ over all intervals. □

Using the above Lemmas, we prove our main complexity guarantee.

Proof of Theorem 22, item 3. We conclude that the runtime of MatchIntervals (Alg. 8), per one step and level, is $\tilde{O}(n^{1+O(\epsilon)} \cdot T_{\mathtt{ad}})$. There's is a constant number of levels, so the same bound holds for each step. Since the algorithm converges in constant steps (as was shown above), and the branching factor is $O(\log n)$, the conclusion follows. □

3.8 Alignment distance algorithm for ad: proof of Theorem 23

In this section we show the algorithm for computing the $\mathtt{ad}_w$ distance, in particular proving Theorem 23. To briefly recall the theorem, we want an algorithm for computing a metric $\mathtt{ad}_w(I, J)$ on w-length strings I, J given an oracle to a metric $\mathfrak{D}_{w/\gamma}$ on w/γ-length strings, running in time poly(γ). Note that we can assume that $w \geq \gamma^4$, as otherwise we can afford to set $\mathtt{ad}(I, J) = \mathtt{ed}(I, J)$ and compute it directly. We manage to ensure it only for "single scale" metrics $\mathtt{ad}_{w,c}$, designed for distances in the range $\approx [c, \gamma c]$.

First, we reduce $\mathtt{ad}_{w,c}$, where $c \in S_w$, to a set of fewer $\mathtt{ad}$ functions, indexed $\mathtt{ad}_{w,t}$, where $t \in \{\gamma, \gamma^2, \ldots, w/\gamma^2\}$. In particular, for given c, we set t by rounding down c/τ to an integer power of γ, for some (large) constant τ (to be fixed later). When $c/\tau < \gamma$, we set $t = \gamma$; and when $c/\tau \geq w/\gamma$, we set $t = w/\gamma^2$.

We note that the algorithm for the "largest distance regime", when $t = w/\gamma^2$, will be different from the rest, when $t \leq w/\gamma^3$. We describe each of the two algorithms separately starting with the "large" distance regime, which is easier.

Henceforth, for simplicity, we set $w' = w/\gamma$, and often refer to $\mathfrak{D}_{w'}$ as $\mathfrak{D}$. Also, for strings I, J of length w, we let $I_{w'}, J_{w'}$ be the strings starting at the same position but of length w' only (e.g., for $I = X_{i,w}$, $I_{w'} = X_{i,w'}$). Finally, we cap all the output distances at w; note that cannot break any of the guarantees (e.g., triangle inequality, or the fact that each $\mathtt{ad}_{w,c} \geq \mathtt{ed}_w$ as $\mathtt{ed}_w \leq w$).

3.8.1 Large distance algorithm: $\mathtt{ad}_{w,w'/\gamma}$ metric

For simplicity of notation, we set $\Gamma = \mathtt{ad}_{w,t}$ when $t = w'/\gamma = w/\gamma^2 \geq \gamma^2$.

The algorithm constructs the following graph H. For $m = \gamma^4$, the vertices are $V_{p,q}$ where $p \in [-m, m]$ and $q \in [-m, m]$. For each node $V_{p,q}$, we add the following edges to H:

- $V_{p,q} \to V_{p+1,q+1}$ of cost $\frac{w/m}{w'}\mathfrak{D}_{w'}(I_{w'} + \frac{w}{m}p, J_{w'} + \frac{w}{m}q)$ (diagonal edges).
- $V_{p,q} \to V_{p,q+1}$ and $V_{p,q} \to V_{p+1,q}$ of cost w/m (gap edges).

Now we run the shortest path from vertex $V_{-m,-m}$ to $V_{m,m}$. Define Γ to be 4 times that shortest path value. It is immediate to note that this can be computed in $m^{O(1)}$ time.

3.8.1.1 Lower and upper bounds on Γ

First, we prove the lower bound.

Lemma 55. *$\Gamma(X_{i,w}, Y_{j,w}) \geq \mathtt{ed}_w(X_{i,w}, Y_{j,w})$ for all i, j.*

Proof. Note that any path in H corresponds to an alignment $A : [-m, m) \to [-m, m) \cup \{\perp\}$ as follows. For each $p \in [-m, m)$, consider the edge which increases p: if it's the diagonal edge $V_{p,q} \to V_{p+1,q+1}$ then $A(p) = q$; otherwise set $A(p) = \perp$. The number of all gap edges is then $2 \cdot |\{p : A(p) = \perp\}|$. Hence, recalling the convention that $\mathtt{ed}_{w'}(X_i, Y_{f(\perp)}) = w'$ for any function $f : \mathbb{N} \to \mathbb{N}$:

$$\Gamma(X_{i,w}, Y_{j,w}) \geq 4\tfrac{w/m}{w'} \sum_{k \in [-m,m)} \mathtt{ed}_{w'}(X_{i+kw/m,w'}, Y_{j+w/m \cdot A(k),w'}) + 8\tfrac{w}{m} \cdot |\{p : A(p) = \perp\}|.$$

By averaging, there must exist some $\delta \in \left[\frac{w'}{w/m}\right]$ such that:

$$\Gamma(X_{i,w}, Y_{j,w}) \geq 4 \sum_{k \in [-\gamma,\gamma)} \mathtt{ed}_{w'}(X_{i+w/m \cdot \delta + kw',w'}, Y_{j+w/m \cdot A(\delta+km/\gamma),w'}) + 8\tfrac{w}{m} \cdot |\{p : A(p) = \perp\}|.$$

We now lower bound this by $\mathtt{ed}(X_{i,w}, Y_{j,w})$. For this, we consider the LCS between these w-length strings from the LCS of the w'-length strings $X_{i+w/m \cdot \delta + kw',w'}$ and $Y_{j+w/m \cdot A(\delta+km/\gamma),w'}$. If $A(\delta+km/\gamma) = \perp$ for all k or $\min_k A(\delta+km/\gamma) \geq 0$, then this is immediate. Otherwise, let $i' \triangleq i+w/m \cdot \delta - w$ and $j' \triangleq j+w/m \cdot \min_{k:A(\delta+km/\gamma) \neq \perp} A(\delta+km/\gamma)$. Define $\overleftarrow{A}(l)$ to be as follows: for $A(\delta + m) \triangleq j' + 2w$ let $\overleftarrow{A}(\delta + km/\gamma) \triangleq \min_{k' \in [k,\gamma], A(\delta+k'm/\gamma) \neq \perp} A(\delta + k'm/\gamma)$. Noting that for our definition of $\mathtt{ed}$, we have $\mathtt{ed}_w(I, J) = w - LCS(I, J)$, we obtain the following bound on LCS, taking into

consideration the overlap:

$$\begin{aligned}LCS(X_{i',2w}, Y_{j',2w}) &\geq \sum_{k\in[-\gamma,\gamma)} LCS(X_{i+w/m\cdot\delta+kw',w'}, Y_{j+w/m\cdot\overleftarrow{A}(\delta+km/\gamma),w'}) \\ &\quad - \sum_{k\in[-\gamma,\gamma)} \tfrac{w}{m}\cdot\left|\overleftarrow{A}(\delta+km/\gamma) - \overleftarrow{A}(\delta+(k+1)m/\gamma) - \tfrac{w'}{w/m}\right| \\ &\geq \sum_{k\in[-\gamma,\gamma)} w' - \mathtt{ed}(X_{i+w/m\cdot\delta+kw',w'}, Y_{j+w/m\cdot A(\delta+km/\gamma),w'}) \\ &\quad - 2\tfrac{w}{m}\cdot|\{p : A(p)=\perp\}| - 2\sum_{k:A(k)=\perp} w',\end{aligned}$$

where we bounded the subtracted overlap by sum of: 1) (scaled) number of $p \in [-m, m)$ s.t. $A(p) = \perp$ (for k's where $A(\delta + km/\gamma), A(\delta + (k+1)m/\gamma)$ are both not-$\perp$), and 2) w' times twice the number of k's with $A(\delta + km/\gamma) = \perp$. Furthermore, $\mathtt{ed}_w(X_{i,w}, Y_{j,w}) \leq \mathtt{ed}_{2w}(X_{i',2w}, Y_{j',2w}) + \frac{w}{m}\cdot|\{p : A(p) = \perp\}|$ and hence:

$$\begin{aligned}\Gamma(X_{i,w}, Y_{j,w}) &\geq \mathtt{ed}_w(X_{i',w}, Y_{j',w}) + \tfrac{w}{m}\cdot|\{p : A(p)=\perp\}|, \\ &\geq \mathtt{ed}_w(X_{i,w}, Y_{j,w}).\end{aligned}$$

□

Now we prove the upper bound. Below, constant $C' > 1$ is from Theorem 23 hypothesis:

Lemma 56. *For any fixed i and any alignment π with $\pi(i) \neq \perp$, we have, for $\Gamma_{i,w} = \Gamma(X_{i,w}, Y_{\pi(i),w})$:*

$$\Gamma_{i,w} \leq O\left(\tfrac{w}{\gamma^2} + \sum_{l\in[-w,w)} \tfrac{1}{w'}\mathfrak{D}(X_{i+l,w'}, Y_{\pi(i+l),w'}) + C'\cdot\sum_{l\in[-w,w)} |\overleftarrow{\pi}(i+l+1) - \overleftarrow{\pi}(i+l) - 1|\right).$$

In the above, in the last term, recall from the preliminaries that $\overleftarrow{\pi}(i')$ is the minimum $\pi(j)$, $j \geq i'$, which is defined ($\neq \perp$). This last term should be thought of

the error stemming from π skipping large chunks of y (which, overall, can be charged to the edit distance between x and y).

Proof. Defining

$$u_\pi \triangleq \sum_{l\in[-w,w)} \tfrac{1}{w'}\mathfrak{D}(X_{i+l,w'}, Y_{\pi(i+l),w'}) + C' \sum_{k\in[-m,m)} |\overleftarrow{\pi}(i+(k+1)w/m) - \overleftarrow{\pi}(i+k\cdot w/m) - w/m|,$$

we note that it is enough to prove that $\Gamma(X_{i,w}, Y_{\pi(i),w}) \le O(w/\gamma^2 + u_\pi)$ as, by simple triangle inequality, $|\overleftarrow{\pi}(i+(k+1)w/m) - \overleftarrow{\pi}(i+k\cdot w/m) - w/m| \le \sum_{l\in[w/m]} |\overleftarrow{\pi}(i+k\cdot w/m+l+1) - \overleftarrow{\pi}(i+k\cdot w/m+l) - 1|$.

Now, let $j = \pi(i)$. For $k \in [-m,m)$, define $A'(k) = \left\lfloor \frac{\overleftarrow{\pi}(i+kw/m)-j}{w/m} + \xi \right\rfloor$ for a tbd $\xi \in [0,1)$. If $A'(i) = A'(i-1)$ or $A'(i) \notin [-m,m)$ then set $A(i) = \perp$ and $A(i) = A'(i)$ otherwise (i.e., only the first copy of a sequence of equal numbers remains).

We claim that, there exists some ξ such that:

$$|\{k : A(k) = \perp\}| \le O(1 + \sum_{k\in[-m,m)} \tfrac{|\overleftarrow{\pi}(i+(k+1)w/m) - \overleftarrow{\pi}(i+kw/m) - w/m|}{w/m}) \le O(1 + \tfrac{u_\pi}{w/m}).$$

Indeed, first we note that any set of real numbers $\{r_1+\xi, \dots r_m+\xi\}$ (for us $r_k = \frac{\overleftarrow{\pi}(i+kw/m)-j}{w/m}$), when rounded down for a random $\xi \in [0,1]$, will have the expected number of duplicates bounded by $\sum_{k=1}^{m} |r_{k+1} - r_k - 1|$ (and hence a fixed ξ exists so that it happens deterministically). Second, the number of distinct $A'(i) \notin [-m,m)$ is upper bounded by (using triangle inequality):

$$1 + \tfrac{m}{w}|\overleftarrow{\pi}(i+w) - \overleftarrow{\pi}(i-w) - w| \le 1 + \sum_{k\in[-m,m)} \tfrac{|\overleftarrow{\pi}(i+(k+1)w/m) - \overleftarrow{\pi}(i+kw/m) - w/m|}{w/m}.$$

Now we have that

$$\begin{aligned}\Gamma_{i,w} &\leq 4 \sum_{k\in[-m,m),A(k)\neq\perp} \tfrac{w/m}{w'}\mathfrak{D}(X_{i+kw/m,w'}, Y_{j+w/m\cdot A(k),w'}) + \tfrac{w}{m}\cdot 8|\{k : A(k)=\perp\}| \\ &\leq 4 \sum_{k\in[-m,m)} \tfrac{w/m}{w'}(\mathfrak{D}(X_{i+kw/m,w'}, Y_{\overleftarrow{\pi}(i+kw/m),w'}) + C'\cdot w/m) + \tfrac{w}{m}\cdot O(1+u_\pi) \\ &= O(u_\pi + C'w\gamma/m) + 4\sum_{k\in[-m,m)} \tfrac{w/m}{w'}\mathfrak{D}(X_{i+kw/m,w'}, Y_{\overleftarrow{\pi}(i+kw/m),w'}).\end{aligned}$$

Now note that, for any $l \in [w/m]$, we have that:

$$\begin{aligned}&\mathfrak{D}(X_{i+kw/m,w'}, Y_{\overleftarrow{\pi}(i+kw/m),w'}) \\ &\quad\leq \mathfrak{D}(X_{i+l+kw/m,w'}, Y_{\overleftarrow{\pi}(i+l+kw/m),w'}) + 8C'l + C'|\overleftarrow{\pi}(i+(k+1)w/m) - \overleftarrow{\pi}(i+kw/m)|.\end{aligned}$$

Hence:

$$\begin{aligned}\Gamma_{i,w} &\leq O(u_\pi + C'w\gamma/m) + 4\sum_{k\in[-m,m)} \tfrac{1}{w/m}\sum_{l\in[w/m]} \tfrac{w/m}{w'}\left(\mathfrak{D}(X_{i+kw/m,w'}, Y_{\overleftarrow{\pi}(i+kw/m),w'})\right) \\ &\leq O(u_\pi + C'w\gamma/m) + 4\sum_{k\in[-m,m)} \tfrac{1}{w'}\sum_{l\in[w/m]} \\ &\qquad\left(\mathfrak{D}(X_{i+l+kw/m,w'}, Y_{\overleftarrow{\pi}(i+l+kw/m),w'}) + 8C'l + C'|\overleftarrow{\pi}(i+(k+1)w/m) - \overleftarrow{\pi}(i+kw/m)|\right) \\ &\leq O(u_\pi + C'w\gamma/m) + 4\sum_{l\in[-w,w)} \tfrac{1}{w'}\mathfrak{D}(X_{i+l,w'}, Y_{\overleftarrow{\pi}(i+l),w'}) + \tfrac{8C}{w'}m\tfrac{w^2}{m^2} \\ &\qquad + \tfrac{4C'}{w'}\sum_{k\in[-m,m)} \tfrac{w}{m}\left(\tfrac{w}{m} + |\overleftarrow{\pi}(i+(k+1)w/m) - \overleftarrow{\pi}(i+kw/m) - \tfrac{w}{m}|\right) \\ &\leq O(u_\pi + C'w\gamma/m) + 4\sum_{l\in[-w,w]} \tfrac{1}{w'}\mathfrak{D}(X_{i+l,w'}, Y_{\overleftarrow{\pi}(i+l),w'}) + 8C'\tfrac{w\gamma}{m} + 4C'\tfrac{w\gamma}{m} + 4\tfrac{w/m}{w'}\cdot u_\pi \\ &\leq O(u_\pi + w/\gamma^2),\end{aligned}$$

where we use $O(C'w\gamma/m) \leq w/\gamma^2$ by our choice of $m = \gamma^4$. □

3.8.2 Not large distance regime: $\mathtt{ad}_{w,t}$ for $t \leq w/\gamma^3$

To compute the $\mathtt{ad}_{w,t}$ distance for smaller t, we use a slightly different alignment representation called *block alignment*, which maps "grid blocks" into x-axis and y-axis "shifts" (loosely speaking). We think of alignment of square blocks of size w' on w by w grid, defined by coordinates p_i, q_i for every square block $i \in [\gamma]$ in the alignment. The first coordinate represents accumulated horizontal shifts, corresponding to inter-block insertions, and the second one represents accumulated vertical shifts, corresponding to inter-block deletions. Such representation is somewhat easier to handle in our case, and will be formally defined below.

Define $\theta = t/w$. Let $T = \gamma^3$. To ease exposition, we use $|\cdot|_1$ notation for the ℓ_1 norm.

Definition 57 ($\mathtt{ad}$)**.** *Let $\mathcal{A}$ be the set of functions $A = (A_x, A_y)$ where $A_x, A_y : [-\gamma, \gamma] \to [T]$ are non-decreasing functions with $A_x[-\gamma] = A_y[-\gamma] = 0$ and $A_x[\gamma] = A_y[\gamma]$. For $I, J \in \mathcal{I}_w$, define the distance $\mathtt{ad}_{w,t}(I, J) = \min\{\mathtt{ad}^*_{w,t}(I, J), w'\theta T\}$, where $\mathtt{ad}^*_{w,t}(I, J)$ is equal to:*

$$\min_{A \in \mathcal{A}} \left(A_x[\gamma] + A_y[\gamma]\right) \theta w' + \sum_{k \in [-\gamma, \gamma)} \frac{1}{T} \sum_{\Delta \in [3T - |A[k]|_1]} \mathfrak{D}\left(I_{w'} + w'(k + \theta(\Delta + A_x[k])), J_{w'} + w'(k + \theta(\Delta + A_y[k]))\right).$$

The former term can be thought of as "shift costs" and the latter as "diagonal costs".

3.8.2.1 Fast computation

Algorithm. The vertices are $V_{k,p,q}$ where $k \in [-\gamma, \gamma]$ and $p, q \in [T]$, as well as final node V^*. For each node $V_{k,p,q}$ we add the following edges to H:

- $V_{k,p,q} \to V_{k+1,p,q}$ of cost $\frac{1}{T}\sum_{\Delta\in[3T-p-q]} \mathfrak{D}(I_{w'} + w'(k+\theta(\Delta+p)), J_{w'} + w'(k+\theta(\Delta+q)))$ (diagonal edges).
- $V_{k,p,q} \to V_{k,p+1,q}$ and $V_{k,p,q} \to V_{k,p,q+1}$ of cost $\theta w'$ (gap edges).
- when $k = \gamma$, $V_{\gamma,p,q} \to V^*$ of cost $\theta w' \cdot |p-q|$ (target edges).

We run shortest path from $V_{-\gamma,0,0}$ to V^*, and output its value, capped at $w'\theta T$ (from above).

Complexity. It's immediate to see that the runtime complexity is $\gamma^{O(1)}$ since H has $O(\gamma T^2)$ edges. Note that this is also the upper bound on the number of calls to $\mathfrak{D}$ distance queries.

3.8.2.2 Lower bound on $\mathtt{ad}_{w,t}$

We establish the lower bounds for $\mathtt{ad}_{w,t}$ in the following lemma.

Lemma 58. *For all $t \in \{\gamma, \dots w/\gamma^3\}$, intervals $I, J \in \mathcal{I}_w$, we have that $\mathtt{ad}_{w,t}(I,J) \geq \min\{\mathtt{ed}_w(I,J), w'\theta T\}$.*

For both the above, as well as for the upper bound shown later, we define distances:

$$d_t(I,J) \triangleq \min_A \sum_{k\in[-\gamma,\gamma)} \frac{1}{T} \sum_{\Delta\in[T]} \mathfrak{D}(I_{w'} + w'(k+\theta(\Delta+A_x[k])), J_{w'} + w'(k+\theta(\Delta+A_y[k]))) + |A[\gamma]|_1\theta w'. \tag{3.20}$$

$$D_t(I,J) \triangleq \min_A \sum_{k\in[-\gamma,\gamma)} \frac{1}{T} \sum_{\Delta\in[3T]} \mathfrak{D}(I_{w'} + w'(k+\theta(\Delta+A_x[k])), J_{w'} + w'(k+\theta(\Delta+A_y[k]))) + |A[\gamma]|_1\theta w'. \tag{3.21}$$

Note that $d_t(I,J) \leq \mathtt{ad}^*_{w,t}(I,J) \leq D_t(I,J)$, hence it is enough to prove Lemma 58 for d_t. We will use D_t later for the upper bound, in Lemma 59. Note that while each

of d_t, D_t lower/upper bounds $\mathtt{ad}_{w,t}$, those may not satisfy triangle inequality over $\mathcal{I}$, hence we use Def. 57 for $\mathtt{ad}$.

Proof of Lemma 58. Without loss of generality, assume that $I = X_{i,w}$ (i.e., it is an interval from x, starting at position i) and $J = Y_{j,w}$ (the cases when I, J are both from x or both from y are treated in exactly the same manner). By the above, it is enough to prove that $d_t(X_{i,w}, Y_{j,w}) \geq \mathtt{ed}(X_{i,w}, Y_{j,w})$. Note that we can rewrite:

$$d_t(X_{i,w}, Y_{j,w}) = \min_A \tfrac{1}{T} \sum_{\Delta\in[T]} \sum_{k\in[-\gamma,\gamma)} \mathfrak{D}(X_i{+}w'(k{+}\theta(\Delta{+}A_x[k])), Y_j{+}w'(k{+}\theta(\Delta{+}A_y[k]))){+}|A[\gamma]|_1\theta w',$$

and hence, for the minimizing A, there's some $\Delta \in [T]$ such that

$$\begin{aligned} d_t(X_{i,w}, Y_{j,w}) &\geq \sum_{k\in[-\gamma,\gamma)} \mathfrak{D}(X_i + w'(k + \theta(\Delta + A_x[k])), Y_j + w'(k + \theta(\Delta + A_y[k]))) + |A[\gamma]|_1\theta w'. \\ &= \sum_{k\in[-\gamma,\gamma)} (\mathfrak{D}(X_i + w'\theta\Delta + w'(k + \theta \cdot A_x[k]), Y_j + w'\theta\Delta + w'(k + \theta \cdot A_y[k])) \\ &\qquad + (A_x[k+1] - A_x[k] + A_y[k+1] - A_y[k]) \cdot \theta w'). \end{aligned}$$

Using that $\mathfrak{D} \geq \mathtt{ed}$, we get, using the notation $i' \triangleq i + w'\theta\Delta \leq i + w$ and $j' \triangleq j + w'\theta\Delta \leq j + w$:

$$\begin{aligned} d_t(I, J) &\geq \sum_{k\in[-\gamma,\gamma)} \mathtt{ed}(X_{i'+w'(k+\theta\cdot A_x[k])}, Y_{j'+w'(k+\theta\cdot A_y[k])}) \\ &\qquad + w'\theta \cdot (A_x[k+1] - A_x[k] + A_y[k+1] - A_y[k]) \\ &\geq \mathtt{ed}(X[i' - w : i' + w + w'\theta A_x[\gamma]], Y[j' - w : j' + w + w'\theta A_y[\gamma]]) \\ &\geq \mathtt{ed}(X[i : i + w], Y[j : j + w]), \end{aligned}$$

where we also used the fact that $A_x[\gamma] = A_y[\gamma]$.

In conclusion: $\mathtt{ad}_{w,t}(I, J) = \min\{\mathtt{ad}^*_{w,t}(I, J), w'\theta T\} \geq \min\{d_t(I, J), w'\theta T\} \geq \min\{\mathtt{ed}(I, J), w'\theta T\}$. □

3.8.3 $\mathtt{ad}_w$ align-approximates $\mathtt{ed}$

Recall that $\mathtt{ad}_w(X_i, Y_j) = \sum_{c \in S_w} c \cdot \mathbb{1}[\mathtt{ad}_{w,c}(X_i, Y_j) \geq c]$. Hence the lower bound, that $\mathtt{ad}_w \geq \mathtt{ed}_w$ point-wise follows immediately from Lemmas 55 and 58 (for the latter, note that $w'\theta T \geq \frac{c}{\tau\gamma^2} T \geq c\sqrt{\gamma}$). In particular for $c^* \in S_w$ such that $1 \leq c^* \leq \mathtt{ed}_w(X_i, Y_j) < 2c^*$, we have $\mathtt{ad}_w(X_i, Y_j) = \sum_{c \in S_w : c \leq c^*} c \geq 2c^* - 1 \geq \mathtt{ed}_w(X_i, Y_j)$ (the case of $\mathtt{ed}_w(X_i, Y_j) = 0$ is immediate).

We now prove the upper bound on $\mathtt{ad}_w$, where C, C' are the constants from the Theorem 23 hypothesis. Fix the alignment $\pi \in \Pi$ that minimizes the cost $k_\pi = \sum_{i \in [n]} \frac{1}{w'} \mathfrak{D}_{w'}(X_{i,w'}, Y_{\pi(i),w'})$. Note that the hypothesis of the theorem implies that $k_\pi \leq C \cdot \mathtt{ed}(x, y)$. We prove the following lemma.

Lemma 59. $\sum_{i \in [n]} \frac{1}{w} \mathtt{ad}_w(X_{i,w}, Y_{\pi(i),w}) \leq O(C') \cdot \sum_{i \in [n]} \frac{1}{w'} \mathfrak{D}_{w'}(X_{i,w'}, Y_{\pi(i),w'})$.

Note that this implies the desired upper bound of $O(CC') \cdot \mathtt{ed}(x, y)$ as required for $\mathtt{ad}_w$ to $O(CC')$-align-approximate $\mathtt{ed}$.

Proof. We first introduce a bit of useful notation, $\mathtt{ed}_\pi(i, w)$, which, intuitively, is the edit distance of the alignment π from $[i, i+w)$ to $[\overleftarrow{\pi}(i), \overleftarrow{\pi}(i+w))$ (though by a factor 2 larger than our definition for regular $\mathtt{ed}$):

$$\mathtt{ed}^i_\pi(i, w) \triangleq |\{i' \in [i, i+w) \mid \pi(i') = \perp\}|.$$

$$\mathtt{ed}^j_\pi(i, w) \triangleq \left|\{j \in [\overleftarrow{\pi}(i), \overleftarrow{\pi}(i+w)) \mid \pi^{-1}(j) = \perp\}\right|.$$

$$\mathtt{ed}_\pi(i, w) \triangleq \mathtt{ed}^i_\pi(i, w) + \mathtt{ed}^j_\pi(i, w).$$

We rewrite $\mathtt{ad}_w$ in terms of $\mathtt{ad}^*_{w,t}$ and Γ as follows, where τ is a constant from the start of the section, still to be determined. Recall that $\mathbb{1}[\mathtt{ad}_{w,c}(I, J) \geq c] =$

$\mathbb{1}[\mathtt{ad}_{w,t}(I,J) \geq c]$, where t is the rounding down of c/τ to a power of γ . Considering the "large" and "not large" cases for t separately, we can further rewrite the "large" case as:

$$\sum_{\substack{c \in S_w \\ c/\tau \geq w/\gamma^2}} c \cdot \mathbb{1}[\mathtt{ad}_{w,c}(I,J) \geq c] \leq 2 \cdot \Gamma(I,J) \cdot \mathbb{1}[\Gamma(I,J) \geq w/\gamma^2 \cdot \tau].$$

Hence, using Lemma 56, for τ at least twice the implicit constant from the lemma (in front of w/γ^2), we obtain that, letting $\mathbf{1}_{i,w,c} \triangleq \mathbb{1}[\mathtt{ad}_{w,c}(X_{i,w}, Y_{\pi(i),w}) \geq c]$:

$$\sum_{\substack{c \in S_w \\ c/\tau \geq w/\gamma}} c \cdot \mathbf{1}_{i,w,c} \leq O\left(\sum_{k \in [-w,w)} \tfrac{1}{w'} \mathfrak{D}(X_{i+k,w'}, Y_{\pi(i+k),w'}) + C' | \overleftarrow{\pi}(i+k+1) - \overleftarrow{\pi}(i+k) - 1| \right).$$

In particular, by summing over all i, and noting that each $\mathfrak{D}(X_{i,w'}, Y_{\pi(i),w'})$ appears at most $w = w'\gamma$ times (and same with the absolute difference terms):

$$\begin{aligned} \sum_i \sum_{\substack{c \in S_w \\ c/\tau \geq w/\gamma}} c \cdot \mathbf{1}_{i,w,c} &\leq w \cdot O\left(\sum_{i \in [n]} \tfrac{1}{w'} \mathfrak{D}(X_{i,w'}, Y_{\pi(i),w'}) + C' | \overleftarrow{\pi}(i+1) - \overleftarrow{\pi}(i) - 1| \right) \\ &\leq w \cdot O\left((1+C') \sum_{i \in [n]} \tfrac{1}{w'} \mathfrak{D}(X_{i,w'}, Y_{\pi(i),w'}) \right), \end{aligned} \tag{3.22}$$

as $\mathfrak{D}(X_{i,w'}, Y_{\pi(i),w'}) = w'$ whenever $\pi(i) = \perp$.

We now upper bound the "not large" regime, which is significantly more involved. For fixed c with $c/\tau < w/\gamma^2$, recall that t is the round-down of c/τ; except if $c/\tau < \gamma$, when we set $t = \gamma$. Hence we have $\mathbb{1}[\mathtt{ad}_{w,c}(I,J) \geq c] = \mathbb{1}[\mathtt{ad}_{w,t}(I,J) \geq c] \leq \mathbb{1}[\mathtt{ad}^*_{w,t}(I,J) \geq c] \leq \mathbb{1}[D_t(I,J) \geq t\tau] + \mathbb{1}[D_\gamma(I,J) \geq c]$, where D_t is as defined in

Eqn. (3.21). We can bound:

$$\sum_{\substack{c\in S_w \\ c/\tau<w/\gamma^2}} c\cdot\mathbb{1}[\mathtt{ad}_{w,c}(I,J)\geq c] \leq O\left(\max_{t:D_t(I,J)\geq t\tau} D_t(I,J)+D_\gamma(I,J)\right)$$
$$\leq O\left(\sum_t \left[D_t(I,J)-\tau t/2\cdot\mathbb{1}[t>\gamma]\right]^+\right), \tag{3.23}$$

where $[x]^+ = \max\{0,x\}$, and recalling that there are $O(\log_\gamma n) = O(1)$ different possible t's considered.

We upper bound $D_t(X_{i,w},Y_{j,w})$ for any fixed t and indeces $i\in[n]$ and $j=\overleftarrow{\pi}(i)$, by exhibiting a convenient choice for function A. We define $A_x[k]$'s, for fixed i and t. For $k=-\gamma$, $A_x[k]=0$ by definition. For $k\in(-\gamma,\gamma)$, define real $\delta_k\geq 0$ to be the smallest such that $\mathtt{ed}^i_\pi(i+w'(k-1+\theta A_x[k-1]),w'+w'\theta\delta_k) = w'\theta\delta_k$ (infinity if it doesn't exist). Set $A_x[k]=\min\{A_x[k-1]+\lceil\delta_k\rceil,T\}$. Then, if $A_x[k]=T$, then set $A_y[k]=T$, as well as the subsequent A_x,A_y. Otherwise, let $A^*_y[k]$ to be the unique integer such that

$$\overleftarrow{\pi}(i+w'(k+\theta A_x[k])) \in \overleftarrow{\pi}(i)+w'(k+\theta A^*_y[k])+[0,w'\theta). \tag{3.24}$$

and set $A_y[k]=\min\{A^*_y[k],T\}$. Note that for $t=\gamma$, we have that $w'\theta=1$ and hence $\overleftarrow{\pi}(i+w'(k+\theta A_x[k])) = \overleftarrow{\pi}(i)+w'(k+\theta A^*_y[k])$.

Finally, set $A_x[\gamma]=A_y[\gamma]=\max\{A_x[\gamma-1],A_y[\gamma-1]\}$.

Claim 60. *Functions A_x,A_y are non-decreasing and $|A[\gamma]|_1\leq O(\frac{\mathtt{ed}_\pi(i-w,3w)}{w'\theta}+\gamma)$. Futhermore, when $t=\gamma$, we have that $|A[\gamma]|_1\leq O(\frac{\mathtt{ed}_\pi(i-w,3w)}{w'\theta})$.*

Proof. While A_x is non-decreasing by construction, we need to prove that A_y is non-decreasing. We have that, as long as $A_x[k]<T$, for $i_k = i+w'(k+\theta A_x[k])$ and

$i_{k-1} = i + w'(k - 1 + \theta A_x[k-1])$, and noting that $i_k - i_{k-1} = w' + w'\theta\lceil\delta_k\rceil$:

$$\begin{aligned}\overleftarrow{\pi}(i_k) - \overleftarrow{\pi}(i_{k-1}) &\geq i_k - i_{k-1} - \mathtt{ed}^i_\pi(i_{k-1}, i_k - i_{k-1}) \\ &\geq w' + w'\theta\lceil\delta_k\rceil - w'\theta\lceil\delta_k\rceil \\ &= w'.\end{aligned}$$

Hence $A_y[k]$ must also be non-decreasing.

To prove the bound on $|A|_1$, we first note that the bound is immediate if $\mathtt{ed}_\pi(i - w, 3w) \geq \gamma^2 t/10$: then $|A|_1 \leq 2T = 2\gamma^3 = 2\frac{\gamma^2 t}{w'\theta}$. Suppose for the rest that $\mathtt{ed}_\pi(i - w, 3w) \leq \frac{\gamma^2}{10}t$ (and hence $\mathtt{ed}^i_\pi(i - w, 3w)$ as well). We now show by induction on k that $A_x[k] \leq T/5$ for $k \geq -\gamma$ and $\delta_k \leq T$ for $k > -\gamma$ — satisfied for $k = -\gamma$ by definition of $A_x[k]$. As for the inductive step:

$$\begin{aligned}\mathtt{ed}^i_\pi(i - w, w + w'(k + \theta A_x[k])) &= \sum_{l=-\gamma+1}^{k} \mathtt{ed}^i_\pi(i + w'(l - 1 + \theta A_x[l-1]), w' + w'\theta\lceil\delta_l\rceil) \\ &\geq \sum_{l=-\gamma+1}^{k} w'\theta\delta_l \\ &\geq \sum_{l=-\gamma+1}^{k} w'\theta\lceil\delta_l\rceil - w'\theta \\ &= w'\theta(A_x[k] - \gamma - k), \qquad (3.25)\end{aligned}$$

and thus $A_x[k] \leq \frac{\mathtt{ed}^i_\pi(i-w, w+w'(k+\theta A_x[k]))}{w'\theta} + \gamma + k \leq \gamma^3/10 + 2\gamma \leq T/5$. This also implies $w'\theta A_x[k] \leq t/\gamma \cdot T/5 \leq t\gamma^2/5 < w'$. Now we also show that δ's are $< T$. Indeed, define $f(\delta) \triangleq \mathtt{ed}^i_\pi(i + w'(k - 1 + \theta A_x[k-1]), w' + w'\theta\delta)$, and note that $f(0), f(T/10) \leq \mathtt{ed}^i_\pi(i - w, 3w) \leq t\gamma^2/10$. At the same time the function $g(\delta) = w'\theta\delta$ grows from 0 to $g(T/10) = w'\theta T/10 \geq t\gamma^2/10$. Hence the non-decreasing functions f, g must intersect somewhere at $\delta \in [0, T/10]$ (with $w'\theta\delta$ an integer).

From the above we have that $A_x[\gamma-1] \le \frac{\mathtt{ed}^i_\pi(i-w,3w)}{w'\theta} + \gamma$. As for $A_y[\gamma-1]$, we have that, using that $\overleftarrow{\pi}(i+z) - \overleftarrow{\pi}(i) \le z + \mathtt{ed}^j_\pi(i,z)$:

$$A_y[\gamma-1] \le \frac{\overleftarrow{\pi}(i+w-w'+w'\theta A_x[\gamma-1]) - (\overleftarrow{\pi}(i)+w-w')}{w'\theta} \le \frac{w'\theta A_x[\gamma-1] + \mathtt{ed}^j_\pi(i,2w)}{w'\theta} \le O(\frac{\mathtt{ed}_\pi(i-w,3w)}{w'\theta} + \gamma).$$

The conclusion follows since $|A[\gamma]|_1 = 2\max\{A_x[\gamma-1], A_y[\gamma-1]\}$.

In the case when $t = \gamma$, we note that $w'\theta = 1$, and hence it is enough to consider δ_k to be integers only. Thus in Eqn. (3.25) we don't need the negative term (that comes only from rounding δ_k), and hence $A_x[\gamma-1], A_y[\gamma-1] \le O(\frac{\mathtt{ed}_\pi(i-w,3w)}{w'\theta})$, from which we obtain the conclusion. □

Since A depends on i and t, we denote it as $A^{i,t}$ below. For each $i \in [n]$, we also define $t_i \in \{\gamma, \dots w/\gamma^2\}$ as the round-down of $\mathtt{ed}_\pi(i-w,3w)$ (i.e., $t_i \le \mathtt{ed}_\pi(i-w,3w) < \gamma t_i$), unless $\mathtt{ed}_\pi(i-w,3w) < \gamma$, in which case $t_i = \gamma$. Also let θ_i be the θ from D_{t_i}, i.e., $\theta_i = t_i/w$.

By the claim above, we have that for any i, t:

$$w'\tfrac{t}{w} \cdot |A^{i,t}[\gamma]|_1 \le O(\mathtt{ed}_\pi(i-w,3w) + t \cdot \mathbb{1}[t > \gamma]). \tag{3.26}$$

We also observe that

$$\tfrac{1}{w}\sum_i \mathtt{ed}_\pi(i-w,3w) = \tfrac{1}{w}\sum_{j\in[3w]}\sum_{i=0}^{n/w-3} \mathtt{ed}_\pi(iw-w+j,3w) \le \tfrac{3w}{w}\cdot O\left(|\{i\in[n] \mid \pi(i) = \perp\}|\right) \le O(k_\pi). \tag{3.27}$$

Hence, for τ sufficiently large constant, and t ranging over $\{\gamma, \dots, w/\gamma^2\}$, we can bound the total "not-large" contribution, $\sum_{i\in[n]} \frac{1}{w}\sum_{\substack{c\in S_w \\ c/\tau < w/\gamma}} c \cdot \mathbb{1}[\mathtt{ad}_{w,c}(X_i, Y_{\pi(i)}) > c]$

as follows, up to the constant from Eqn. (3.23):

$$\overset{(3.23)}{\leq} 2 \cdot |\{i : \pi(i) = \perp\}| + \tfrac{1}{w} \sum_{i \in [n]} \sum_{t} \left[D_t(X_i, Y_{\overleftarrow{\pi}(i)}) - \tau t/2 \cdot \mathbb{1}[t > \gamma] \right]^+$$

$$\overset{(3.21)}{\leq} 2k_\pi + \sum_{\substack{i \in [n] \\ t}} \tfrac{1}{w} \left[\tfrac{1}{T} \sum_{\substack{k \in [-\gamma, \gamma) \\ \Delta \in [3T]}} \mathfrak{D}(X_{i + w'(k + \theta_t(\Delta + A_x^{i,t}[k]))}, Y_{\overleftarrow{\pi}(i) + w'(k + \theta_t(\Delta + A_y^{i,t}[k]))}) + w' \theta_t |A^{i,t}[\gamma]|_1 - \tfrac{\tau t \cdot \mathbb{1}[t > \gamma]}{2} \right]^+$$

$$\overset{(3.26)}{\leq} O(k_\pi) + \tfrac{1}{w} \sum_{\substack{i \in [n] \\ t}} \left[\tfrac{1}{T} \sum_{\substack{k \in [-\gamma, \gamma) \\ \Delta \in [3T]}} \mathfrak{D}(X_{i + w'(k + \theta_t(\Delta + A_x^{i,t}[k]))}, Y_{\overleftarrow{\pi}(i) + w'(k + \theta_t(\Delta + A_y^{i,t}[k]))}) + O(\mathtt{ed}_\pi(i - w, 3w)) - \tfrac{\tau t \cdot \mathbb{1}[t > \gamma]}{3} \right]^+$$

$$\overset{(3.27)}{\leq} O(k_\pi) + O(k_\pi) + \tfrac{1}{w} \sum_{\substack{i \in [n] \\ t}} \left[\tfrac{1}{T} \sum_{\substack{k \in [-\gamma, \gamma) \\ \Delta \in [3T]}} \mathfrak{D}(X_{i + w' \theta_t \Delta + w'(k + \theta_t A_x^{i,t}[k])}, Y_{\overleftarrow{\pi}(i) + w' \theta_t \Delta + w'(k + \theta_t A_y^{i,t}[k])}) - \tfrac{\tau t \cdot \mathbb{1}[t > \gamma]}{3} \right]^+$$

$$\overset{(3.24)}{\leq} O(k_\pi) + \tfrac{1}{w} \sum_{\substack{i \in [n] \\ t}} \left[\tfrac{1}{T} \sum_{\substack{k \in [-\gamma, \gamma) \\ \Delta \in [3T]}} \mathfrak{D}(X_{w' \theta_t \Delta + i + w'(k + \theta_t A_x^{i,t}[k])}, Y_{w' \theta_t \Delta + \overleftarrow{\pi}(i + w'(k + \theta_t A_x^{i,t}[k]))}) + \left(6\gamma \cdot C' w' \theta_t - \tfrac{\tau t}{3}\right) \cdot \mathbb{1}[t > \gamma] \right]^+$$

$$\leq O(k_\pi) + \tfrac{1}{wT} \sum_{\substack{i \in [n] \\ t}} \sum_{\substack{k \in [-\gamma, \gamma) \\ \Delta \in [3T]}} \mathfrak{D}\left(X_{w' \theta_t \Delta + i + w'(k + \theta_t A_x^{i,t}[k])}, Y_{\overleftarrow{\pi}(w' \theta_t \Delta + i + w'(k + \theta_t A_x^{i,t}[k]))}\right)$$

$$+ \tfrac{O(C')}{wT} \sum_{\substack{i \in [n] \\ t}} \sum_{\substack{k \in [-\gamma, \gamma) \\ \Delta \in [3T]}} \left| \overleftarrow{\pi}(w' \theta_t \Delta + i + w'(k + \theta_t A_x^{i,t}[k])) - (w' \theta_t \Delta + \overleftarrow{\pi}(i + w'(k + \theta_t A_x^{i,t}[k]))) \right|,$$

where the last two inequalities are due to triangle inequality for $\mathfrak{D}$ and $\mathfrak{D}(Y_j, Y_{j+1}) \leq C'$, as well as for τ satisfying $6\gamma \cdot C' w' \theta_t \leq \frac{\tau t}{3}$, i.e., $\tau \geq 18C$. To estimate the last term, we note

that, again since $w'\theta T \le w'$:

$$\begin{aligned}
S &\triangleq \tfrac{1}{wT} \sum_{\substack{i\in[n]\\ t}} \sum_{\substack{k\in[-\gamma,\gamma)\\ \Delta\in[3T]}} \left| \overleftarrow{\pi}(w'\theta_t\Delta + i + w'(k+\theta_t A_x^{i,t}[k])) - (w'\theta_t\Delta + \overleftarrow{\pi}(i + w'(k+\theta_t A_x^{i,t}[k]))) \right| \\
&\le \tfrac{1}{wT} \sum_{\substack{i\in[n]\\ t}} \sum_{\substack{k\in[-\gamma,\gamma)\\ \Delta\in[3T]}} \mathtt{ed}_\pi^j(i + w'(k+\theta_t A_x^{i,t}[k]), w'\theta_t\Delta). \\
&\le \tfrac{1}{wT} \sum_{\Delta\in[3T]} \sum_{i\in[n],t} 3\cdot \mathtt{ed}_\pi(i-w, 3w). \\
&\le O(\log_\gamma n \cdot \mathtt{ed}_\pi(1,n)) \\
&\le O(k_\pi).
\end{aligned}$$

Finally, overall we have, using that $A_x^{i,t}[k] \in [0,T]$ and hence we can absorb it into Δ-summation, now over $\Delta \in [4T]$ (the main reason we are using Δ to start with):

$$\begin{aligned}
\sum_{i\in[n]} \tfrac{1}{w} \sum_{\substack{c\in S_w\\ c/\tau<w/\gamma}} c\cdot \mathbb{1}[\mathtt{ad}_{w,c}(X_i, Y_{\pi(i)}) > c] &\le O(C'k_\pi) + \tfrac{1}{w'} \sum_t \sum_{\substack{k\in[-\gamma,\gamma)\\ \Delta\in[4T]}} \tfrac{1}{\gamma T} \sum_{i\in[n]} \mathfrak{D}(X_{i+w'k+\frac{t}{\gamma}\Delta}, Y_{\overleftarrow{\pi}(i+w'k+\frac{t}{\gamma}\Delta)}) \\
&\le O(C'k_\pi) + \tfrac{1}{w'} \sum_t 8 \sum_i \mathfrak{D}(X_i, Y_{\pi(i)}) \\
&\le O(\log_\gamma n \cdot C'k_\pi) \\
&= O(C'k_\pi).
\end{aligned}$$

We complete the proof of Lemma 59, combining the bound for "large" distance regime (Eqn. (3.22), scaled by $1/w$), and the above "not large" distance regime, we obtain that:

$$\sum_{i\in[n]} \tfrac{1}{w} \mathtt{ad}_w(X_{i,w}, Y_{\pi(i),w}) \le O(C'k_\pi) \le O(C') \cdot \sum_{i\in[n]} \tfrac{1}{w'} \mathfrak{D}_{w'}(X_{i,w'}, Y_{\pi(i),w'}),$$

which completes the proof of Lemma 59. □

3.8.4 $\mathtt{ad}_{w,c}$ are metrics

Finally, we prove that $\mathtt{ad}_{w,c}$ is a metric for each c. Given how we reduce them to $\Gamma, \mathtt{ad}_{w,t}$, it is enough to prove metricity for the latter two.

Γ **metric.** Identity follows by definition. Symmetry follows from the fact that the graph H is symmetric.

It remains to prove the triangle inequality. Consider three intervals I, J, K. We want to prove that:

$$\Gamma(I, K) \leq \Gamma(I, J) + \Gamma(J, K).$$

Note that, for the pair (I, J), the optimal path in graph H corresponds to an alignment $A_{IJ} : [-m, m) \to [-m, m) \cup \{\perp\}$. In particular, for each $p \in [-m, m)$, consider the edge which increases p: if it's the diagonal edge $V_{p,q} \to V_{p+1,q+1}$ then $A_{IJ}(p) = q$; otherwise $A_{IJ}(p) = \perp$. The number of used gap edges is then $2 \cdot |\{p : A_{IJ}(p) = \perp\}|$. Hence:

$$\Gamma(I, J) = 4\tfrac{w/m}{w'} \sum_{\substack{k \in [-m,m) \\ A_{IJ}(i+\frac{w}{m}k) \neq \perp}} \mathfrak{D}(I_{w'} + \tfrac{w}{m}k, J_{w'} + \tfrac{w}{m} \cdot A_{IJ}(k)) + 8 \cdot |\{p : A_{IJ}(p) = \perp\}|.$$

Similarly we can extract A_{JK}. We now define an alignment A_{IK} from I to K as

$A_{IK}(p) = A_{JK}(A_{IJ}(p))$ (and $\perp$ if either functions has value $\perp$). Then we have that:

$$\begin{aligned}
\Gamma(I,K) &\le 4\tfrac{w/m}{w'} \sum_{\substack{k\in[-m,m)\\ A_{IK}(i+\frac{w}{m}k)\neq\perp}} \mathfrak{D}(I_{w'}+\tfrac{w}{m}k, K_{w'}+\tfrac{w}{m}\cdot A_{IK}(k)) + 8\cdot|\{p : A_{IK}(p)=\perp\}| + 2w/\gamma^2 \\
&\le 4\tfrac{w/m}{w'} \sum_{\substack{k\in[-m,m)\\ A_{IK}(i+\frac{w}{m}k)\neq\perp}} \mathfrak{D}(I_{w'}+\tfrac{w}{m}k, J_{w'}+\tfrac{w}{m}\cdot A_{IJ}(k)) + 8\cdot|\{p : A_{IK}(p)=\perp\}| \\
&+ 4\tfrac{w/m}{w'} \sum_{\substack{k\in[-m,m)\\ A_{IK}(i+\frac{w}{m}k)\neq\perp}} \mathfrak{D}(J_{w'}+\tfrac{w}{m}\cdot A_{IJ}(k), K_{w'}+\tfrac{w}{m}\cdot A_{IK}(k)) + 2w/\gamma^2 \\
&\le 4\tfrac{w/m}{w'} \sum_{\substack{k\in[-m,m)\\ A_{IJ}(k)\neq\perp}} \mathfrak{D}(I_{w'}+\tfrac{w}{m}k, J_{w'}+\tfrac{w}{m}\cdot A_{IJ}(k)) + 8\cdot|\{p : A_{IJ}(p)=\perp\}| + 2w/\gamma^2 \\
&+ 4\tfrac{w/m}{w'} \sum_{\substack{k\in[-m,m)\\ A_{IK}(i+\frac{w}{m}k)\neq\perp}} \mathfrak{D}(J_{w'}+\tfrac{w}{m}\cdot A_{IJ}(k), K_{w'}+\tfrac{w}{m}\cdot A_{JK}(A_{IJ}(k))) + 8\cdot|\{p : A_{JK}(p)=\perp\}| \\
&\le \Gamma(I,J)+\Gamma(J,K).
\end{aligned}$$

$\mathtt{ad}_{w,t}$ **metric.** Identity and Symmetry are trivial (for identity, we use $A = (\vec{0},\vec{0})$ and for symmetry we switch the coordinates of A).

To show triangle inequality, fix $I, J, K \in \mathcal{I}$. We prove that $\mathtt{ad}_{w,t}(I,K) \le \mathtt{ad}_{w,t}(I,J) + \mathtt{ad}_{w,t}(J,K)$. Fix A, A' which minimize $\mathtt{ad}^*_{w,t}(I,J), \mathtt{ad}^*_{w,t}(J,K)$. Define $A'' = A + A'$ (i.e. $A''_x[l] = A_x[l] + A'_x[l]$ and $A''_y[l] = A_y[l] + A'_t[l]$). Note that if we obtain $A''_x[l] \ge T$ or $A''_y[l] \ge T$, then we already have that $\mathtt{ad}_{w,t}(I,J) + \mathtt{ad}_{w,t}(J,K) \ge w'\theta T \ge \mathtt{ad}_{w,t}(I,K)$, and hence we can assume that $A''_X[l], A''_y[l] < T$ below. Last, we use the shorthand $|A|_1 = |A[\gamma]|_1$, meaning the total number of shifts of A.

In the below, $\mathtt{ad}^* = \mathtt{ad}^*_{w,t}(I,K)$:

$$
\begin{aligned}
\mathtt{ad}^* \le & \sum_{l\in[-\gamma,\gamma)} \frac{1}{T} \sum_{\Delta\in[3T-|A''[l]|_1]} \mathfrak{D}(I_{w'} + w'(l+\theta(\Delta+A''_x[l])), K_{w'} + w'(l+\theta(\Delta+A''_y[l]))) + |A''|_1\theta w' \\
\le & \sum_l \frac{1}{T} \sum_{\Delta\in[3T-|A''[l]|_1]} \Big(\mathfrak{D}(I_{w'} + w'(l+\theta(\Delta+A''_x[l])), J_{w'} + w'(l+\theta(\Delta+A'_x[l]+A_y[l]))) \\
& + \mathfrak{D}(J_{w'} + w'(l+\theta(\Delta+A'_x[l]+A_y[l])), K_{w'} + w'(l+\theta(\Delta+A''_y[l])))\Big) + |A''|_1\theta w' \\
= & \sum_l \frac{1}{T} \sum_{\substack{\Delta\in[3T-|A''[l]|_1] \\ +A'_x[l]}} \mathfrak{D}(I_{w'} + w'(l+\theta(\Delta+A''_x[l]-A'_x[l])), J_{w'} + w'(l+\theta(\Delta+A_y[l]))) \\
& + \frac{1}{T} \sum_{\substack{\Delta\in[3T-|A''[l]|_1] \\ +A_y[l]}} \mathfrak{D}(J_{w'} + w'(l+\theta(\Delta+A'_x[l])), K_{w'} + w'(l+\theta(\Delta+A''_y[l]-A_y[l]))) + |A''|_1\theta w' \\
\le & \sum_l \frac{1}{T} \sum_{\Delta\in[3T-|A[l]|_1]} \mathfrak{D}(I_{w'} + w'(l+\theta(\Delta+A_x[l])), J_{w'} + w'(l+\theta(\Delta+A_y[l]))) \\
& + \frac{1}{T} \sum_{\Delta\in[3T-|A'[l]|_1]} \mathfrak{D}(J_{w'} + w'(l+\theta(\Delta+A'_x[l])), K_{w'} + w'(l+\theta(\Delta+A'_y[l]))) + |A|_1\theta w' + |A'|_1\theta w' \\
\le & \mathtt{ad}^*_{w,t}(I,J) + \mathtt{ad}^*_{w,t}(J,K).
\end{aligned}
$$

where the 2nd step is triangle inequality of $(\mathcal{I}, \mathfrak{D})$, 3rd step is change of variables, 4th step is triangle inequality of $(\mathbb{N}^2, \ell_1)$, and last step is by definition. The conclusion follows from the fact that all $\mathtt{ad}^*$'s are thresholded at the same threshold.

3.9 Fast data structures

In this section, all reals have $O(\log^2 n)$ of precision (in fixed point representation).

Theorem 61 (Theorem 50, restated)**.** *Given a set of numbers $a_1, \dots a_n \ge 0$ in a sparse representation (i.e., as a set S of i's with $a_i > 0$), there exists a data structure D supporting the following queries:*

- *Given some $i < j$ compute $\sum_{k=i}^{j} a_k$. The query complexity is $O(\log n)$.*

- *Given some i and $\zeta > 0$, compute $j \in S, j \leq i$ such that $\sum_{k=j-1}^{i} a_k < \zeta \leq \sum_{k=j}^{i} a_k$ (or output that none exists). The query complexity is $O(\log n)$.*

The preprocessing time is $O(|S| \log n)$.

Proof. First of all, at preprocessing time, we can compute all the prefix sums $\sigma_i = \sum_{k=1}^{i} a_k$ for $i \in S$. Then we build a predecessor data structure [84] on S as well as on the reals σ_i for $i \in S$. Then the first type of queries is answered by two predecessor queries (and taking their difference). The second type of queries is answered by a single query to the (second) predecessor data structure. The preprocessing and query runtime follow immediately from standard predecessor data structures.

□

Theorem 62 (Theorem 49, restated)**.** *Given a set of numbers $a_1, \ldots a_n \geq 0$ in a sparse representation (i.e., as a set S of i's with $a_i > 0$) there exists a data structure D supporting the following queries:*

- *Sample index $i \in [n]$ from the distribution $\{a_i / \sum_j a_j\}_i$ in time $O(\log n)$;*
- *Given $k \geq 1$, produce a set $S \subseteq [n]$ that includes each i with probability $\min\{ka_i, 1\}$ independently. The runtime is $O((1 + k \cdot \sum_{i=1}^{n} a_i) \log n)$ in expectation.*
- *Given an interval $[s, t] \subset [n]$, and $k \geq 1$, produce a set $S \subseteq [n]$ that includes each $i \in [s, t]$ with probability $\min\{ka_i, 1\}$ independently. The runtime is $O((1 + k \cdot \sum_{i=s}^{t} a_i) \log^2 n)$ in expectation.*

Furthermore, the preprocessing time is $O(|S| \cdot \log^{O(1)} n)$.

We note that similar data structures were also developed in [58]; we include our proof below for completeness of the algorithm.

Proof. Let $m = |S|$. First, note that we can assume wlog that $m = n$. When $m < n$, we pre-process vector $b \in \mathbb{R}^m$ consisting of the non-zero entries in a in an increasing order. Then, for the third query type, define the semi-monotone function $\Phi : [n+1] \to [m+1] - 1$, which maps each $i \in [n+1]$ to the number of non-zero entries whose index is strictly smaller than i in a. For that task, we build during preprocessing a BST to calculate $\Phi(i)$ (in additive time $\tilde{O}(m)$), and when queried on interval $[s,t]$, we use interval $[\Phi(s)+1, \Phi(t+1)]$ instead (with $O(\log n)$ additive query overhead).

At preprocessing, precompute all partial sums $p_i = \sum_{j=1}^{i} a_j$ for all $i \in [n]$. Let $p_0 = 0$. Also, assuming n is a power of 2 (otherwise we can pad to nearest power), we build a balanced binary search tree with each leaf corresponding to an index $i \in [n]$. Each node v of the tree stores the sum of leafs in the subtree, σ_v. Also, sort all a_i in increasing order (and keep as pairs (i, a_i)).

Query of the first type. Pick a random uniform $r \in [0,1]$ (say, to $O(\log^2)$ bits of precision) and perform a binary search on $r \cdot \sum_j a_j$ in the set $p_0, p_1, \ldots p_n$. If $p_i \leq r \sum_j a_j \leq p_{i+1}$, then output $i+1$. Note that the probability that output is i is precisely $a_i / \sum_j a_j$.

Query of the second type. The algorithm works as follows. First, using the sorted list, find all i's such that $a_i \geq 0.5/k$ (i.e., the ones that are included with set S with probability $\geq 1/2$). Call this set L. Sample each $i \in L$ into set S accordingly (taking $O(|L|)$ time).

Start at the root r, generate a integer q_r from the Poisson distribution with expectation $2k\sigma_r$. Then, we proceed recursively as follows: for a node v with integer q_v, for children with sums σ, σ', pick random integers q, q' such that: 1) $q + q' = q_v$, and 2) assuming that q_v is from distribution $\text{Poi}(2k\sigma_v)$, then q, q' are independently from $\text{Poi}(2k\sigma), \text{Poi}(2k\sigma')$ respectively. Recurse into each child with > 0 integer q. At a leaf $i \in [n] \setminus L$, if $q_i > 0$ (and hence $q_i \geq 1$), then include i into set S with probability $\frac{a_i k}{1-e^{-2a_i k}}$. (Indeces in L are ignored here.)

We now briefly argue that the above produces the desired set. First of all note that the procedure on the tree generates q_i from $\text{Poi}(a_i k)$ distribution independently (using Poission's distribution property). We note that $\frac{1-e^{-2x}}{x} = 2-2(2x)/2!+2(2x)^2/3!-... \geq 1$ for $x \leq 1/2$, hence the probability $\frac{a_i k}{1-e^{-2a_i k}}$ is indeed less than 1 for $i \in [n] \setminus L$. For such i, we indeed include it into S with probability

$$\Pr[i \in S] = \Pr[q_i \geq 1] \cdot \frac{a_i k}{1-e^{-2a_i k}} = a_i k < 1.$$

We now bound the query time. First, we note that L has size at most $2k \cdot \sum_{i=1}^{n} a_i$ (since each included i satisfies $a_i \geq 1/2k$). Second, we note that $\mathbb{E}[q_r] = O(k\sigma_r)$, and the tree procedure has runtime at most $O(\log n)$ factor of the number of leafs with $q_i > 0$, of which there are exactly q_r. Hence runtime is $O((1 + k\sum_{i=s}^{t} a_i) \log n)$ in expectation.

Query of the third type. We augment the data structure D from above as follows. Recall the dyadic intervals on $[n]$ are intervals of the form $[j2^i + 1, (j+1)2^i]$, where $i \in \{0, \ldots \log n\}$ and $j \in \{0, \ldots n/2^i - 1\}$. For each dyadic interval $I = [u, v]$, we build a data structure D_I on the numbers $a_u, \ldots a_v$. For a query interval $[s, t]$, we decompose $[s, t]$ into $O(\log n)$ dyadic intervals, and perform query of the second type

in each of these. □

Theorem 63 (Theorem 51, restated)**.** *Suppose we are given an $m \times n$ matrix A, where $m \leq n$, given in the sparse form, i.e., as a set S of non-zero entries. Also, suppose we are given n intervals $[s_i, t_i] \subseteq [n]$ such that both $\{s_i\}_i$ and $\{t_i\}_i$ are non-decreasing. Given $\gamma > 0$, we can find all pairs $(i, \chi) \in [n] \times [m]$ such that $\sum_{k=s_i}^{t_i} A_{\chi,k} \geq \gamma$. The runtime is $\tilde{O}\left(|S| + n + \frac{\sum_{\chi,i} \sum_{k=s_i}^{t_i} A_{\chi,k}}{\gamma}\right)$.*

Proof. Suppose n is a power of 2; otherwise, just pad with zero's until the closest power of 2. Recall the dyadic intervals on $[n]$ are intervals of the form $[j2^i+1, (j+1)2^i]$, where $i \in \{0, \ldots \log n\}$ and $j \in \{0, \ldots n/2^i - 1\}$. We call i to be the level of the dyadic interval.

First, for each dyadic interval I, we store two lists S_I, T_I, containing a subset of (i, s_i) and (i, t_i) respectively, sorted by s_i's/t_i's respectively. Specifically, for each given interval $[s_i, t_i]$, we decompose it into at most $2 \log n$ dyadic intervals $I_1, I_2, \ldots$. For each such dyadic interval I_k, we add s_i to S_{I_k} and t_i to T_{I_k}. Note that this takes $O(n \log n)$ time.

We also store a dictionary data structure (e.g., a BST) storing, for each pair (χ, I), where $\chi \in [m]$ and I a dyadic interval, the quantity $\tau_{\chi,I} = \sum_{k \in I} A_{\chi,k}$. In particular, we store only the *non-zero* $\tau_{\chi,I}$. We can compute this efficiently as follows: 1) initialize an empty dictionary; 2) enumerate through all $(\chi, k) \in S$ and add $A_{\chi,k}$ to all $\tau_{\chi,I}$ where $k \in I$; 3) if some $\tau_{\chi,I}$ is not yet in the dictionary, create an entry for it first. Note that the runtime is $O(|S| \log n)$.

The rest of the algorithm proceeds as follows. Using the dictionary structure, find all pairs (χ, I) of $\chi \in [m]$ and dyadic interval $I = [u, v]$ such that $\tau_{\chi,I} \geq \frac{\gamma}{2 \log n}$. For

each one of them, using S_I, let $s_i \in S_I$ be the max $s_i \in S_I$. Similarly, let $t_j \in T_I$ be the min. For each interval index l such that $j \leq l \leq i$ and $s_l \in S_I$ (equivalently $t_l \in T_I$), we compute $\sum_{k=s_l}^{t_l} A_{\chi,k}$ (using the dyadic interval sums) in $O(\log n)$ time. We output (l, χ) if the resulting sum is $\geq \gamma$. At the end we remove the duplicate (l, χ) if multiple copies have been output.

We now argue correctness and runtime. Consider an interval $[s_l, t_l]$ with mass at least γ for some $\chi \in [m]$. Then it can be decomposed into $\leq 2 \log n$ dyadic intervals, at least one of which, say $I = [u, v]$, has to have mass $\gamma_{\chi,I} \geq \frac{\gamma}{2 \log n}$. Hence, when we consider the pair (χ, I) where the dyadic interval $I = [u, v]$, we will have that $s_l \leq s_i \leq u$ and $v \leq t_j \leq t_i$ (by the definition of i, j). Hence we will output the pair (l, χ).

Let's argue runtime. First of all, the number of pairs (χ, I) with non-zero τ is at most $|S| \cdot \log n$ as each $A_{\chi,i}$ can contribute to $O(\log n)$ pairs (χ, I). Hence enumerating the dictionary takes $O(|S| \cdot \log n)$ time. For each pair (χ, I) with $\tau_{\chi,I} > 0$, and each interval index $l \in [n]$ that we end up checking, we have that $\sum_{k=s_l}^{t_l} A_{\chi,k} \geq \gamma / 2 \log n$. Furthermore any such (l, χ) can be considered only $O(\log n)$ times (as this is how many times l appears in the data structures S_I, T_I over all dyadic intervals I). Hence the total number of pairs (l, χ) for which we estimate the precise mass is upper bounded by:

$$O(\log n) \cdot \sum_{\chi} \sum_{l} \frac{\sum_{k=s_l}^{t_l} A_{\chi,k}}{\gamma / 2 \log n}.$$

Hence total time spend in this phase is $O(\log^3 n) \frac{\sum_{\chi,i} \sum_{k=s_i}^{t_i} A_{\chi,k}}{\gamma}$. This completes the analysis.

□

Chapter 4

Longest Increasing Subsequence

Longest Increasing Subsequence (LIS) is a fundamental statistic of a sequence. For an ordered sequence $x_1, \ldots, x_n$, the problem is finding the length of the largest subsequence of $x_1, \ldots, x_n$ which is strictly increasing.

In this chapter, we focused on the popular parametrized version of the problem. In particular, we show that for any $n \in \mathbb{N}$ and $\lambda = o(1)$, there exists a (randomized) *non-adaptive* algorithm that, given a sequence of length n with $\mathsf{LIS} \geq \lambda n$, approximates the LIS up to a factor of $1/\lambda^{o(1)}$ in $n^{o(1)}/\lambda$ time, proving Theorem 4, which we restate below:

Theorem 64 (Theorem 4, restated)**.** *For $n \in \mathbb{N}$, and any $\lambda = o(1)$, there exists a (randomized) non-adaptive algorithm that, given a sequence of length n with $\mathsf{LIS} \geq \lambda n$, approximates the length of LIS up to a $1/\lambda^{o(1)}$ factor in $O\left(\frac{1}{\lambda} \cdot n^{o(1)}\right)$ time with high probability.*

Technical Contributions. Our main technical contributions are three-fold (see technical overview in Section 4.2):

- We define (the LIS problem in) a new sub-linear computational model, which we call Genuine-LIS and which may be of independent interest. In the Genuine-LIS problem, we are given a sequence $y \in \mathbb{N}^n$, with a caveat that only some of the elements of y are "genuine" and the others are "corrupted", a property we can test an element for. The goal is to estimate the length of LIS *among the genuine* elements of y, using as few tests as possible, while the values of y are known to the algorithm "for free".

- We show how one can efficiently reduce the LIS problem to Genuine-LIS and vice versa. These reductions together constitute the backbone of our recursive algorithm.

- To obtain the promised query complexity, we develop a new data structure, that we call a Precision-Tree. This data structure samples non-adaptively (but non-uniformly) elements of the input y and preprocesses them for efficient operations on the sampled elements. This part is the only part of the algorithm that samples the input string — the aforementioned recursive calls of LIS and Genuine-LIS problems access this data structure only. Furthermore, the data structure allows one to compute accurate estimations for composition of general functions from "coarse" (sub-)estimates — a requirement for our recursive approach.

While we make progress in understanding the complexity of estimating the LIS, the following important question remains open:

Open Question ($(1+\epsilon)$-approximation)**.** *Does there exist an algorithm that, given a sequence of length n with $\mathit{LIS} \geq \lambda n$, estimates the LIS up to a $1+\epsilon$ factor in*

$O\left(\frac{1}{\lambda} \cdot n^{o(1)}\right)$ *time with probability 2/3?*

We believe that finding an improved approximation algorithm for the Genuine-LIS problem above may lead to a $(1+\epsilon)$ approximation algorithm for LIS.

We give a technical overview of our algorithm in Section 4.2, after setting up preliminaries in Section 4.1. Section 4.3 contains the proof of the main theorem, assuming results proved in subsequent sections. In Section 4.4, we develop the Precision-Trees data structure that is used to improve the sample and time complexity of the main algorithm. The guarantees of the two primary subroutines are proved in Section 4.5 and Section 4.6. In Section 4.7, we extend these algorithms in certain ways critical to our final application.

4.1 Preliminaries

Sequences and intervals. Given a set Ω and $n \in \mathbb{N}$, a sequence $y = (y_1, y_2, \cdots, y_n) \in \Omega^n$ is an ordered collection of elements in Ω. A block sequence $y \in \Omega^{n \times k}$ is a partially ordered collection of elements in Ω. Abusing notation, we will also allow for some elements in a block to be "null" — for example, we write $y \in \mathbb{N}^{n \times k}$ to also mean $y \in \{\mathbb{N}, \perp\}^{n \times k}$, where $\perp$ is "null".

For $\ell \in [n]$, we say that $z = (z_1, \cdots, z_\ell)$ is a subsequence of y of length ℓ, and denote it by $\{y_{i_j}\}_j$, if there exist integers $1 \le i_1 < i_2 < \cdots < i_\ell \le n$ such that $z_j = y_{i_j}$ for all $j \in [\ell]$. We refer to the indices i_j as coordinates and the values y_{i_j} as element values. For a block sequence y, we say that $z = (z_1, \cdots, z_\ell)$ is a subsequence of y of length ℓ, and denote it by $\{y_{w_j}\}_j$, if there exist integers $1 \le i_1 < i_2 < \cdots < i_\ell \le n$ such that $z_j \in y_{i_j,*}$ for all $j \in [\ell]$.

Define the interval space $\mathcal{I} \triangleq \{[a,b] \mid a,b \in \mathbb{N}, a \leq b\} \cup \{[a,b) \mid a \in \mathbb{N}, b \in [1,\infty], a < b\}$. For $I \in \mathcal{I}$, we use $|I|$ to denote $|I \cap \mathbb{N}|$, i.e., the number of natural numbers contained in interval I. We often refer to an x-interval $X \in \mathcal{I} \cap 2^{[n]}$ as an interval of coordinates, and a y-interval $Y \in \mathcal{I}$ as an interval of element values.

For a block sequence $y \in \mathbb{N}^{n \times k}$ and a y-interval $Y \in \mathcal{I}$, we write $y \cap Y$ to denote the *multi-set* of elements in y that are also in Y. Also, for $X \subseteq [n]$, $y(X)$ is the sequence with first coordinates restricted to X. We also define $y(X,Y) := y(X) \cap Y$.

Monotonicity. We define monotone sets as follows:

Definition 65 (Monotone sets)**.** *Fix a (potentially partial) ordered set $(\Omega, <)$. We say that a set $P \subseteq \mathbb{N} \times \Omega$ is* monotone *if for all $(i,u),(j,v) \in P \times P$, we have (i) $i = j \Leftrightarrow u = v$; and (ii) $i < j \Leftrightarrow u < v$.*

Note that this definition captures the notion of an increasing subsequence. In particular, for the standard notion of an increasing subsequence over natural numbers, we take $\Omega = \mathbb{N}$ and $<$ as the usual "less than" relation over $\mathbb{N}$ (a total order). However, we will need this more general definition to consider increasing subsequences over other partially ordered sets, like the space of intervals $\mathcal{I}$.

For a finite set $P \subset \mathbb{N} \times \Omega$, a longest increasing subsequence (LIS) of P is a monotone set $Q \subset P$ of maximum cardinality. We often use OPT to refer to a particular LIS, and use $|\mathsf{OPT}|$ to denote its length.

Distributions. For $p \in [0,1]$, we use $\texttt{Ber}(p)$ to denote the Bernoulli distribution with success probability p, and $\texttt{Bin}(n,p)$ to denote the binomial distribution with parameters n and p. By convention, we project p to the range $[0,1]$ whenever $p > 1$ or $p < 0$.

We use "i.i.d. random variables" to mean that a collection of random variables is independent and identically distributed, and use "sub-sampling i.i.d. with probability p" to mean that each element is sampled independently with equal probability p.

Operations on Vectors, Sets, Functions. For a set $A \subset \mathbb{R}$ and a number α, we define $A + \alpha := \{a + \alpha : a \in A\}$ and $\alpha A := \{\alpha\, a : a \in A\}$. We write log to denote binary logarithm.

The notation $\circ$ is used for function composition (i.e., $g \circ f(x) = g(f(x))$, and $\oplus$ is used for direct sum. We use the notation $*$ as argument of a function, by which we mean a vector of all possible entries. For example, $f(*)$ is a vector of $f(i)$ for i ranging over the domain of f (usually clear from the context).

We use $E_b(k)$ for $b, k \in \mathbb{N} \cup \{\infty\}$ to denote the set of powers of b bounded by k, i.e., $E_b(k) \triangleq \{b^i \mid i \in \mathbb{N}\} \cap [1, k]$. We also define $E_b \triangleq E_b(\infty)$. We use $\mathbb{R}_+$ to denote the set of non-negative real numbers.

Approximations. Our constructions generate approximations which contain both multiplicative and additive terms. We use the following definition:

Definition 66 ((α, β)-Approximation)**.** *For $\alpha \geq 1$ and $\beta > 0$, an (α, β)-approximation $\hat{q}$ of a quantity q is an α-multiplicative and β additive estimation of q, i.e., $\hat{q} \in [q/\alpha - \beta, q]$.*

The following fact shows that to obtain a multiplicative approximation which is a function of the LIS, it suffices to show (α, β)-approximation.

Fact 67. *Suppose we have an algorithm $\mathcal{A}$ that for any $n \in \mathbb{N}, \lambda \in (0, 1)$, for some $q \in (0, 1)$, outputs a $(1/\lambda^q, \lambda n)$-approximation for some unknown quantity $\ell \in [0, n]$*

in time t. Then there exists an algorithm that outputs $\widehat{\ell} \in [\Omega(\delta^p), 1] \cdot \ell$ in time $t + O(1)$, where $p \triangleq \frac{q}{1-q}$, as long as $\ell \geq 2\delta n$.

Proof. The algorithm calls $\mathcal{A}$ with parameter $\lambda = \delta^{1+p}$, and returns the same output. Then, the upper bound is immediate by definition of (α, β)-approximation. For the lower bound, as long as $\ell \geq 2\delta n$, we have that:

$$\widehat{\ell} \geq \delta^{q(1+p)} \cdot \ell - \delta^{1+p} \cdot n \geq \delta^p \cdot \ell - \tfrac{1}{2}\delta^p \cdot \ell \geq \tfrac{1}{2}\delta^p \ell \qquad (q = p/(1+p))$$

□

Other notation. Notation $\tilde{O}(\cdot)$ hides polylog(n) factors, while $O^*(\cdot)$ hides a factor of $n^{o(1)}$.

4.2 Technical overview

Our starting point is the algorithm of [185] which achieves $O(1/\lambda^3)$ approximation in time $\tilde{O}(\sqrt{n}/\lambda^7)$. Let OPT be (the coordinates of) an optimum solution (LIS) with length $|\mathsf{OPT}| \geq \lambda n$. Their algorithm consists of 2 main steps:

1. Partition $[n]$ into $\sqrt{n}$ disjoint contiguous x-intervals $X_1, X_2, \ldots, X_{\sqrt{n}}$ of length $\sqrt{n}$ each. For $i \in [\sqrt{n}]$, let $y(X_i)$ be the restriction of the input sequence y to x-interval X_i. The goal is to approximate the y-interval $Y_i \triangleq [s_i, \ell_i]$, where s_i and ℓ_i are, respectively, the minimum and maximum values in $y(X_i \cap \mathsf{OPT})$. To do this, one samples $O(1/\lambda)$ elements in X_i, and generates $O(1/\lambda^2)$ *candidate y-intervals* (using pairs of sampled element values).

2. Generate a set of mutually disjoint *pseudo-solutions*, each of which is a sequence of $\Omega(\lambda\sqrt{n})$ candidate y-intervals that are monotone, i.e., all values in a candidate

interval in x-interval X_i are less than all values in a candidate interval in x-interval X_j for all $i < j$. Estimate the quality of each pseudo-solution (the sum of LIS of the candidate intervals in it) by simply sampling $O(\log^{O(1)} n/\lambda^4)$ x-intervals and computing the LIS within relevant candidate intervals. Output the largest quality.

Their analysis proceeds as follows. For each x-interval X_i, some candidate y-interval is a good approximation of the interval Y_i w.h.p., so the union of the LIS in all pseudo-solutions covers a large fraction of OPT. Since there are $O(1/\lambda^3)$ pseudo-solutions, the output is a $O(1/\lambda^3)$-approximation of OPT. The runtime of $\tilde{O}(\sqrt{n}/\lambda^7)$ is dominated by the time required to evaluate the quality of pseudo-solutions to sufficient accuracy. One can also apply this technique recursively to improve the runtime, at the cost of worse approximation.

In [161], this result is improved by giving an algorithm for LIS with approximation factor $O(1/\lambda^\epsilon)$ and runtime $O(n^{1-\Omega(\epsilon)} \cdot (\log n/\lambda)^{1/\epsilon})$ for any constant $\epsilon > 0$. The polynomial dependence on $1/\lambda$ seems essential since the algorithm in [185] is used as a sub-routine. In [172], a *non-adaptive* algorithm is presented with sample complexity $\tilde{O}(\sqrt{r}/\lambda^2)$ and approximation $O(1/\lambda)$, where r is the number of distinct values. Again, the polynomial dependence of the sample complexity on r and $1/\lambda$ seems intrinsic.

There are several fundamental obstacles in improving these bounds, and in particular, getting the runtime down all the way to $1/\lambda$ while improving the approximation. One such obstacle is that the straight-forward approach (as in prior work) of independent recursions cannot obtain better than $1/\lambda^2$ run-time in the worst-case. The

issue is that even if one is able to isolate all possible y-interval ranges efficiently, there must still be at least $1/\lambda$ such ranges (optimally), and recusing of each such y-interval without knowing a priori which coordinates contain the relevant y-values would require $1/\lambda$ samples just to find a single sample of relevance in each (sub-)instance (i.e. for each y-interval). We will return to this obstacle later and show how we improve the *amortized complexity* for it.

4.2.1 Our approach

Similarly to [185], we generate candidate y-intervals based on random samples, and generally look for increasing sequences of intervals, each with a large local LIS. Beyond this general similarity, our algorithm develops a few new ideas, in particular in how we work with these candidate intervals, and especially how we find LIS's among these candidate intervals. In particular, our contribution can be seen through three main components.

First, we introduce a new problem, termed Genuine-LIS, which captures the problem of estimating the longest (sub-)sequence of increasing intervals, each with a large "local LIS". This problem is a new model for sublinear-time algorithms, which has not appeared in prior literature to the best of our knowledge.

The Genuine-LIS problem is defined as follows. The input is a sequence g where each element is associated with an additional flag signifying whether it is *genuine* or not, i.e., $g \in (\mathbb{N} \times \{0,1\})^n$. One receives unrestricted access to the elements (i.e., the first coordinates) which is denoted by $g(*)_1 \in \mathbb{N}^n$. However, one must "pay" to test whether an element is genuine; this is determined by the second coordinates $g(*)_2 \in \{0,1\}^n$, referred to as genuineness flags. The goal is to compute the length

of the longest increasing subsequence of $g(*)_1$ restricted to genuine elements, i.e., of $g((g(*)_2)^{-1}(1))_1$, using as few *tests for genuineness* as possible.

Second, we show how to efficiently reduce the LIS problem to a (smaller) instance of the Genuine-LIS problem, where the "genuineness" flag of an element corresponds to LIS of an x-interval being large enough. In particular, this is where we generate the aforementioned candidate y-invervals. The algorithm reduces the problem to finding a (global) LIS amongst a set of candidate intervals, each with a large (local) LIS. We solve this via a composition of 2 sub-problems: the Genuine-LIS problem takes care of the "global LIS" of candidate-intervals, while the standard LIS problem recursively estimates the "local LIS" (these ideas and terms will be made precise later). The main challenge here is to generate candidate intervals in a *succinct* manner: we manage to reduce the number of sampled anchor needed to $\approx 1/\lambda$, as well as limiting the number of candidate intervals to be *near-linear* in the number of sampled points.

Third, to optimize the sampling of the string, we use a novel sampling scheme endowed with a data structure, termed *Precision-Tree*. At preprocessing, the structure samples a number of positions in the input string (non-adaptively), which are not uniform but rather correspond to a tree with non-uniform leaf levels (hence the name). We then build a global data structure on these samples for efficient access. In particular, we develop a *tree decomposition procedure* (to be discussed later), and repeatedly use *sub-trees* across different sub-instances we generate, improving the overall sample complexity. We highlight the fact that we *reuse* both the samples (and hence randomness) across different (sub-)instances for both of the LIS and Genuine-LIS problems. In addition, the Precision-Tree data structure provides a quick way to narrow down on all samples in some x-interval whose value is in given y-interval Y.

Before continuing this overview, we formally define a slightly generalized version of the LIS problem, termed Block-LIS, as well as the aforementioned Genuine-LIS problem. Following that, we provide an overview of the algorithms to solve these two problems. The high-level flow of our algorithm is described in Figure 4.

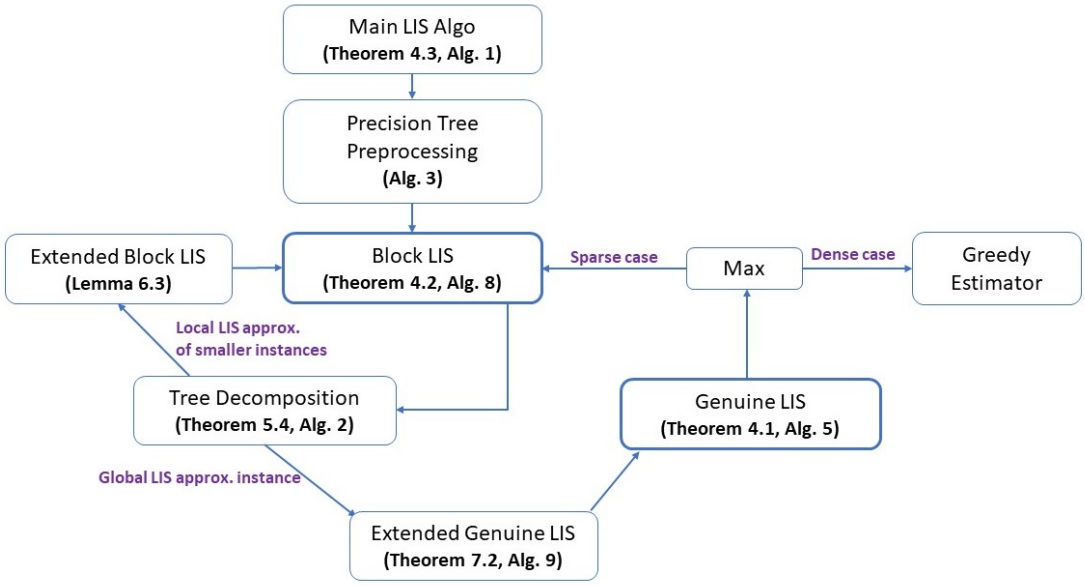


Figure 4: High level flow of the main algorithm (Theorem 64). Each arrow represents one or more calls to the corresponding algorithm.

The Block-LIS problem. This problem extends standard LIS problem in two ways. First, the main input is a sequence of n blocks, each containing (at most) k elements, and at most one element in each block may participate in a sequence. Second, we are also given a range of values $Y \in \mathcal{I}$, such that each element of a subsequence must be in Y.

Formally, we define the Block-LIS as follows: given a sequence $y \in \mathbb{N}^{n \times k}$ and a range of values $Y \in \mathcal{I}$, compute the length of a maximal sub-sequence $\mathsf{OPT} \subseteq [n] \times [k]$, such that $\{(w_1, y_w)\}_{w \in \mathsf{OPT}}$ is monotone and $\{y_w\}_{w \in \mathsf{OPT}} \subseteq Y$. Sometimes,

we also restrict the set of blocks to an interval $X \subseteq [n]$, and define the quantity Block-LIS(y, X, Y) as the longest increasing subsequence of $y \in \mathbb{N}^{X \times [k]}$ using elements in Y.

The standard LIS problem can be seen as the Block-LIS problem with $k = 1$ and $Y = \mathbb{N}$ — which is how we instantiate the original input sequence $y \in \mathbb{N}^n$ in our Block-LIS algorithm. The main advantage of this generalization appears when the sequence is *sparse in* Y. Then, we show the multiplicative approximation factor for Block-LIS is not only a function of the additive error λn, but also of the total number of elements of y in range Y (i.e., $|y(X, Y)|$). In particular, we show that this approximation is a function of $\frac{\lambda n}{|y(X,Y)|}$.

While the generalization to blocks is not a core necessity of the algorithm (in fact, one can consider the values in each block in descending manner, yielding an equivalent problem with no blocks required), its main use comes from our need to instantiate several overlapping instances using the same Precision-Tree data structure (to be discussed later).

Finally, we note that we similarly generalize the Genuine-LIS to work over blocks: input $g \in (\mathbb{N} \times \{0, 1\})^{n \times k}$ and the LIS is allowed to use only one (genuine) element per block.

4.2.2 Main algorithm

The main algorithm for estimating the LIS is merely the following (see Algorithm 11):

1. Preprocess Precision-Tree $T = T_\delta(y)$ of the input sequence with parameter $\delta \triangleq \lambda / n^{o(1)}$ (the exact $o(1)$ parameter will be made precise later).

2. Run the Block-LIS algorithm that has access only to the data structure T. (we note that this algorithm itself recursively runs the algorithms for Genuine-LIS and Block-LIS.)

4.2.3 Genuine-LIS problem: overview of the algorithm EstimateGenuineLIS

Recall that in the Genuine-LIS problem with input $g \in (\mathbb{N} \times \{0,1\})^n$, one receives unrestricted access to the elements (i.e., the first coordinates), but must "pay" to test whether an element is genuine (determined by the second coordinates). The goal is to compute the length of the longest increasing subsequence of $g(*)_1$ restricted to genuine elements.

We derive two complementary solutions to the Genuine-LIS problem: one "direct" (without further recursion), and the other by reducing the problem to standard sublinear-time LIS estimation. To highlight our novel contribution, we contrast it with the framework of [185], that implicitly gives a solution to the Genuine-LIS problem, albeit with sub-optimal approximation. In particular, the following algorithm for Genuine-LIS is analogous to the algorithm in [185] based on generating pseudo-solutions (maximal sequences of candidate intervals).

1. Using the first coordinates only, look for a maximal *increasing subsequence* P_1 containing $\gtrapprox \lambda n$ elements and remove P_1 from the sequence.

2. Repeat step (1) until there are no more sequences of length $\gtrapprox \lambda n$. This generates (disjoint) solutions $P_1, \ldots, P_t$ for some $t \lessapprox 1/\lambda$.

3. Sub-sample $\approx 1/\lambda$ elements from the union $P = \cup_{i \in t} P_i$ and check if each one is

genuine. Let κ be the number of Genuine elements in this sampled set.

4. Output $\lambda^2 n \cdot \kappa$.

It is easy to see that this algorithm yields, with constant probability, a $(2/\lambda, \lambda^2 n)$-approximation with approximately $1/\lambda$ tests for genuineness. First, if the Genuine-LIS is at most $2\lambda n$, then the output (being greater than 0) is already a $(2/\lambda, \lambda^2 n)$-approximation, as

$$\frac{\text{Genuine-LIS}}{2/\lambda} - \lambda^2 n \leq \frac{2\lambda n}{2/\lambda} - \lambda^2 n = 0 \leq \text{Genuine-LIS}.$$

Otherwise, P contains most of the Genuine-LIS elements, and since $t \lessapprox 1/\lambda$, the Genuine-LIS is at least $\lambda/2$ times the number of genuine elements in P.

Thus, by the sampling mechanism above, we can estimate the proportion of genuine elements to sufficient accuracy with constant probability. Observe that the approximation is in fact proportional to $1/\lambda\kappa$, and that the worst approximation happens when P is "sparse", i.e., when only approximately λn elements in P are genuine. For such a *sparse case*, however, one can instantiate the Genuine-LIS algorithm as a Block-LIS one, restricted to the genuine elements. In fact, this "sparse" case is precisely where the approximation factor is minimal, i.e., $\frac{\lambda n}{|y|} = 1$ where y is the genuine-elements-only subsequence. Our improved approximation stems, in part, from balancing between the dense and sparse cases as above.

Speeding up LIS extraction using dynamic LIS. The Genuine-LIS instances we generate are of size $\lessapprox 1/\lambda$, and our goal is to obtain overall run-time that is near-linear in $1/\lambda$ as well. The standard, dynamic-programming solution for finding and extracting the optimal LIS each time for step (1) above can potentially incur

an overhead that is quadratic in the instance size (each LIS extraction would take linear time, and we need to greedily extract up to a near-linear number of solutions), leading to a polynomially-worse time. To ensure near-linear time, we adopt a fast dynamic LIS data structure from [101], and use it to iteratively extract near-maximal pseudo-solutions.

Extensions to the Genuine-LIS algorithm. Our complete algorithm needs two further extensions to the algorithm for the Genuine-LIS problem, as follows.

- *Sparse (unbalanced) instances:* some Genuine-LIS instances we generate are sparse, with many "null" elements, i.e., some blocks have less than k elements (note that a "null" element is different from a "non-genuine" one, in that it does not need to be tested). When the instance g consists of $m \ll nk$ non-null elements, we would like the approximation and runtime bounds to be a function of m/n (average block size) instead of k (maximum block size). To obtain the improved bound, we partition the blocks based on an exponential discretization of the number of non-null elements, and output the maximum over all instances, where each instance consists only of blocks containing a similar number of non-null elements.

- *Genuine-LIS over intervals:* the first coordinates of the Genuine-LIS instances we generate apriori consist of intervals rather than integers. The challenge is that we then need to find LIS over the *partial order* of intervals $\mathcal{I}$. To exemplify the challenge, the dynamic LIS data-structures from [160; 101; 138] which are useful in speeding up our algorithm, cannot immediately handle partial order sequences. Our ideal solution involves a mapping $\varphi : \mathcal{I} \to \mathbb{N}$ that approximately

preserves the overall LIS over all subsequences, and therefore also preserves the overall LIS over the genuine elements. While we are unable to show a single mapping that works for all intervals, we partition the space of intervals into $\log(k/\lambda)$ sets $\mathcal{I}_\ell$ — intervals in $\mathcal{I}$ whose length is in $[\ell, 2\ell)$ — based on an exponential discretization of the interval length ℓ, and provide a mapping $\varphi_\ell : \mathcal{I}_\ell \to \mathbb{N}$ for each set of intervals $\mathcal{I}_\ell$. We eventually output the maximal Genuine-LIS result over all such maps. This costs us merely another $\log(k/\lambda)$-factor approximation, and a small additive error.

The formal statement for the algorithm to solve Genuine-LIS, named EstimateGenuineLIS, is presented in Section 4.3, and its description and analysis are in Section 4.5. The pseudo-code is presented in Algorithm 15. The Genuine-LIS extensions are presented and analyzed in Section 4.7.

4.2.4 Block-LIS problem: overview of the algorithm EstimateBlockLIS

For the problem Block-LIS(y, X, Y), we develop the algorithm EstimateBlockLIS. Fundamentally, we reduce the instance to a Genuine-LIS instance, where each genuineness flag is set to 1 if and only if the LIS of a certain (smaller) sequence (which is itself a Block-LIS problem) is large enough.

Our reduction starts by partitioning X into consecutive intervals $X_1, \ldots, X_\tau$ of equal length, where τ is a dynamic, carefully chosen branching factor, and is usually sub-polynomial in the instance size (here think $\tau = n^\epsilon$). Next, we sample approximately τ/λ blocks (w_j, y_{w_j}) (termed *anchors*), and use all the elements in Y across

all the anchors to generate sets of values $\mathcal{S}_i \subseteq Y$ for each corresponding X_i.

Using each set of y-values $\mathcal{S}_i$, we construct *candidate intervals* $\mathcal{Y}_i$. While this part is similar to the construction from [185], we note that we need a more efficient construction to obtain the near-optimal sample complexity. In particular, we require the number of candidate intervals to be near-linear in $|\mathcal{S}_i|$, and hence we construct a *small* candidate interval set $|\mathcal{Y}_i| \approx |\mathcal{S}_i|$, while still ensuring that this set covers all "relevant options" with only an extra logarithmic factor loss in approximation. In particular, instead of looking for all possible pairs of endpoints, we choose the endpoints in dyadic fashion (in particular, their distance in the set $\mathcal{S}$ must be a power of 2.

Finally, we reduce the Block-LIS problem to a Genuine-LIS instance over τ blocks, where each block i contains all the candidates intevals $\mathcal{Y}_i$ — this Genuine-LIS instance thus captures "global" LIS (over the candidate intervals). The first coordinates of the Genuine-LIS instance are the candidate intervals $\mathcal{Y}_i$ themselves, while the second coordinates (i.e., the genuineness flags) indicate whether the corresponding "local" Block-LIS(y, X_i, Y') estimated values are above a certain threshold κ, for each $Y' \in \mathcal{Y}_i$.

This threshold κ itself depends on a parameter $\rho > 1$, which characterizes the relation between the "global" Genuine-LIS instance and the "local" Block-LIS instances. Intuitively, ρ is such that (roughly) $1/\rho$ fraction of the X_i intervals each have a $\lambda\rho$ fraction of them participating in the LIS. Consider two extreme cases, one where the LIS is uniformly distributed among all X_i ($\rho = 1$), and one where the LIS is maximally concentrated among a small subset of the x-intervals ($\rho = 1/\lambda$). Then, it is more difficult to certify an increasing subsequence in the latter case, when the LIS is sparse. But in this case testing the genuineness of a local LIS (inside a block X_i) is much

easier, since we only need to establish that it is $\Omega(|X_i|)$. In general, there is a precise trade-off between the complexity of certifying the global LIS versus the complexity of certifying the local Block-LIS. A priori, we do not know ρ, so we simply iterate over all possible magnitudes (again, by exponential discretization).

Once we formulate the overall Block-LIS problem as a composition of a "global" Genuine-LIS over multiple "local" Block-LIS instances, we decompose the problem through a procedure called *Precision-Tree decomposition* which will be described next.

The formal statement for the EstimateBlockLIS algorithm is presented in Section 4.3, and the algorithm description and analysis are in Section 4.6. The pseudo-code (including sub-routines) are presented in Algorithms 16 to 18.

4.2.5 Precision-Tree data structure

We note that a direct instantiation of the above algorithm yields the right approximation, but will require at least $\approx 1/\lambda^2$ queries into the input string y (and hence run-time), and, moreover, require the algorithm to be *adaptive*. The source of this inefficiency is precisely the "small slopes" obstacle discussed earlier, which would lead to "wasting samples" in any naïve rejection sampling mechanism. To improve the complexity, and to allow our sampling algorithm to be non-adaptive, we introduce the *Precision-Trees* data structure (Section 4.4). While based on the *precision sampling* tool introduced in [21; 22], the main development here is that we augment the resulting tree for special operations described henceforth.

Intuitively, the Precision-Tree data structure can be thought of as a global data structure holding enough information for simulating random samples for multiple non-uniform (sub-)instances, specifically ones arising from recursion. In particular,

the data structure is an (incomplete) tree, where (lowest-level) leafs correspond to samples of some input $e \in \mathcal{E}^n$. The initial tree is defined over the input string (i.e., $\mathcal{E} = \mathbb{N}$), but we also construct Precision-Trees over other domains $\mathcal{E}^n$ as described later. The initial tree for the string $y \in \mathbb{N}^n$ is constructed at random (using precision sampling), and this is the only time when we access the string y (in the entire LIS algorithm). Overall, the Precision-Tree data-structure (the initial one as well as the other trees we generate) supports several important properties we need:

- Since our algorithm is recursive (composition of Genuine-LIS and Block-LIS algorithms), we need to provide samples for potentially very small parts of y (that depend on what was sampled earlier in the recursion). In other words, we need to be able to "zoom into" different location of the input string with the right precision.

  Our precision tree is build such that any sub-tree under any "sampled" node, is another Precision-Tree instantiation with a different (random) parameter.

- In contrast to the original precision sampling tool that was designed for the simple addition function, our application at hand requires more complicated functions (indeed, the sub-linear Block-LIS and Genuine-LIS algorithms).

  We show how we can *decompose* the tree into a number of different sub-trees (as mentioned, each sub-tree by itself is a precision tree), run algorithms on each sub-tree as well as a final algorithm, recomposing the results using another algorithm (and another sub-tree, this time over a new domain resulting from the sub-trees computation)

- Another crucial challenge to deal with is the following: as the recursion proceeds,

we may be "zooming in" on the same part of y multiple times, but focusing on different y-range intervals Y. Naïvely, one would use rejection sampling here (rejecting samples with value not in Y), which however would yield a polynomially worse sample complexity. In addition, our recursion happens not only down the tree (i.e., on sub-trees) but also on the top portion of the tree (reminiscent of the van Emde Boas layout).

The precision tree allows us to *reuse* the randomness over independent recursive calls into the same x-intervals, each focusing on a different y-value ranges. In particular, for such an x-interval we will have processed the samples to directly report samples with values in a desired y-interval Y (this is essentially a 2D range reporting). In addition, the randomness is reused when computing the global function on the top portion of the tree as described above.

4.2.5.1 Construction of the Precision-Tree data structure

We now describe the construction in a bit more detail. Original precision sampling from [21; 22] is designed to estimate a *summation* function $a = \sum_{i=1}^{n} a_i$, for unknown $a_i \in [0, 1]$, from "coarse" estimates for a_i. In our application, we need to *generalize* precision sampling from a simple addition to *general functions*, allowing one to approximate $g \circ f$ where g is a general function over m coordinates and $f = (f_1, \ldots, f_n)$ consists of n *independent* functions on different parts of the input, sharing the same co-domain.

We define Precision-Tree with precision parameter $\lambda < 1$, for a given a vector of elements $e = (e_1, \ldots, e_n) \in \mathcal{E}^n$, denoted $T_\lambda(e)$, as follows. The tree T is a trimmed version of the full β-ary tree of $\ell = \log_\beta n + 1$ levels, with $n = \beta^\ell$ leafs, where

parameter $\beta > 1$ is fixed; in particular, all nodes of T have fan-out of either β or 0. Each leaf at level ℓ [1] is associated with an integer representing its location in the tree. Each internal node v is associated with an interval representing the unions of its leaf descendants (in the full tree); for example, the root of T is associated with the entire interval $[1, n]$, and its children with $[1, n/\beta]$, $[n/\beta + 1, 2n/\beta]$, etc). For a node v in the tree, we use $\mathcal{E}_v$ or $\mathcal{E}(v)$ to denote the set of elements under v.

Given $e \in \mathcal{E}^n$, λ, and β, we construct the trimmed Precision-Tree $T_\lambda(e)$ as follows. We assign a (random) precision score $\mathcal{P}_v$ to each node v by the following recursive procedure. We set $\mathcal{P}_{\text{root}(T)} = \lambda$. Recursively we define the precision score of a node v by

$$\mathcal{P}_v = \mathbf{Z}_v \cdot \mathcal{P}_{\text{parent}(v)} \text{ where } \mathbf{Z}_v \sim_{i.i.d.} \text{Uniform}(\{1, 2, 3, \ldots, \beta/4\}) \ . \tag{4.1}$$

Most importantly, for a node v, we recurse into its children only if $\mathcal{P}_v \leq 1$; otherwise we stop the recursion for v with $\mathcal{P}_v > 1$ (v has 0 children). If v is a leaf (at level ℓ), v also stores the input element corresponding the location of v (i.e., an element of e) — note that this event corresponds to sampling an element of the input string. In other words, we store only a "trimmed" version of a full β-ary tree.

Initially, we generate one such tree $T_{1/c}(y)$ for the input string $y \in \mathbb{N}^n$, for some $c = c(n, \lambda) \geq 1$, which we refer to as "precision tree complexity". One can prove that the sample complexity (into y) is bounded by $c \cdot \log^{O(\log_\beta n)}(\beta)$ (see Lemma 71). Overall, our entire sampling mechanism is merely a simple, one-round non-adaptive precision-sampling tree over the coordinates of y, which we store in a data structure

[1] By convention, the level of the root is 0.

T[2] of size $O^*(1/\lambda)$ with convenient fast access.[3] After such preprocessing, the rest of the algorithm has no access to y, but only to T.

Most importantly, using this single tree $T_{1/c}(y)$, we will simulate Precision-Tree access for different trees over different inputs and parameters in our algorithms using a certain procedure which we call the Trim-Tree algorithm (see Algorithm 12).

We also preprocess the samples stored in T so that we can quickly retrieve all samples in some x-interval $X \subseteq [n]$ and some y-interval Y through an ancillary data-structure which we call SODS. The SODS is used for a "rejection sampling"-type operations where we need a number of samples from a substring of input y (or later on for some intervals of blocks), but care only about values in a certain range Y.

4.3 Main algorithm for estimating LIS

Our main algorithm is composed of two algorithms, for solving Block-LIS and Genuine-LIS. These algorithms recursively call each other, with access to a Precision-Tree data structure. This data structure queries the input sequence at the beginning (non-adaptively and non-uniformly), and our algorithms access the sequence only via this data structure. We describe the details of Precision-Tree in Section 4.4, and for now refer to it as a tree $T_\lambda(y)$ for some parameter $\lambda < 1$ and an input string y.

We now state the main guarantees of the two algorithms; their proof will appear in later sections. We then show how these two algorithms yield our main algorithm for estimating LIS. Below, for an instance $g \in (\mathbb{N} \times \{0,1\})^{n \times k}$, let g restricted to

[2]We abuse notation and use T to denote both the conceptual Precision-Tree and the data structure implementing it.

[3]The notation $O^*(\cdot)$ hides a $n^{o(1)}$ factor.

first/second coordinate be $g(*)_1 \in \mathbb{N}^{n\times k}$ and $g(*)_2 \in \{0,1\}^{n\times k}$ respectively.

Theorem 68 (EstimateGenuineLIS algorithm; Section 4.5)**.** *Fix integers n, k and $\lambda \in (1/n, 1)$. Fix an instance $g \in (\mathbb{N}\times\{0,1\})^{n\times k}$. For some monotone functions $a_s : \mathbb{R}_+^3 \to [1,\infty)$ and $c_s : \mathbb{R}_+^4 \to [1,\infty)$. Suppose there exists a randomized algorithm $\mathcal{A}_{BL}$ that, given $X \subseteq [n], Y \subseteq \mathbb{N}$, parameter τ_s, and a Precision-Tree $T_{1/c_s\left(|X|,\tau_s,\lambda,\frac{\lambda\cdot|X|}{|y(X,Y)|}\right)}(y)$ for some $y \in \mathbb{N}^{n\times k}$, can produce an $(\alpha_s, \lambda|X|)$-approximation for Block-LIS(y, X, Y) where $\alpha_s = a_s\left(|X|, \tau_s, \frac{\lambda\cdot|X|}{|y(X,Y)|}\right)$ w.h.p. in time t_s.*

Fix any parameter $\gamma, \tau \geq 1$, and let $c \geq \tilde{O}(1/\lambda) + c_s(n, \tau, \lambda, \gamma\lambda/k)$. Then, there exists an algorithm $\mathcal{A}$, that, given free access to $g(*)_1$ and Precision-Tree access to $g(*)_2$, $T_{1/c}(g(*)_2)$, produces a $(\alpha_g, \lambda n)$-approximation for Genuine-LIS(g) w.h.p., where $\alpha_g = \log(k/\lambda)\cdot\max\{\gamma, a_s(n, \tau, \gamma\lambda/k)\}$. The algorithm $\mathcal{A}$ runs in time $\tilde{O}(nk) + O(t_s)$.

Theorem 69 (EstimateBlockLIS algorithm; Section 4.6)**.** *Fix monotone functions $t_s, a_g : \mathbb{R}_+^2 \to [1,\infty)$, $a_s, c_g : \mathbb{R}_+^3 \to [1,\infty)$ and $c_s : \mathbb{R}_+^4 \to [1,\infty)$, satisfying, for all $r, \tau \in E_\beta$, $m \in \mathbb{N}$, $\lambda < 1$, $\lambda_1, \lambda_2 \in [\lambda, 1]$, and $k' \in [1, 1/\lambda]$ with $\lambda_1\lambda_2 = \Omega(\lambda k')$:*

- $a_s\left(r, \tau, \frac{\lambda}{m}\right) \geq \text{polylog}\left(\frac{k}{\lambda}\right)\cdot a_g\left(\tau, \frac{\lambda_2}{k'}\right)\cdot a_s\left(\frac{r}{\tau}, \tau, \frac{\lambda_1}{m}\right)$;
- $c_s\left(r, \tau, \lambda, \frac{\lambda}{m}\right) \geq \beta^{O(1)}\cdot\frac{\tau}{\lambda} + c_g(\tau, \lambda_2, k')\cdot c_s\left(\frac{r}{\tau}, \tau, \Theta(\lambda_1/k'), \frac{\lambda_1}{m}\right)$; *and,*
- $t_s(r, \tau) \geq \log^{O(\log_\beta(\tau))}(\beta)\cdot t_s\left(\frac{r}{\tau}, \tau\right)$.

Suppose there exists and algorithm $\mathcal{A}_{GL}$ with the following guarantee: given Genuine-LIS instance $g \in (\mathbb{N}\times\{0,1\})^{n_g\times k_g}$ with β-ary Precision-Tree $T_{1/c_g(n_g,\lambda_g,k_g)}(g(*)_2)$ access, $\mathcal{A}_{GL}$ outputs $\left(a_g\left(n_g, \frac{\lambda_g}{k_g}\right), \lambda_g n_g\right)$-approximation to Genuine-LIS(g) in time $\tilde{O}(n_g\cdot k_g)$ w.h.p.

Now fix input $y \in \mathbb{N}^{n\times k}$, a block interval $X \subseteq [n]$, value range interval $Y \subseteq \mathbb{N}$, parameters $\lambda \in (0,1)$, $\beta \in \mathbb{N}$, $\tau \in E_\beta$. Then, given a β-ary Precision-Tree $T_{1/c}(y)$, we can produce a $(\alpha, \lambda|X|)$-approximation for Block-LIS(y, X, Y) *w.h.p., as long as* $\alpha \geq a_s\left(|X|, \tau, \frac{\lambda|X|}{|y(X,Y)|}\right)$ *and* $c \geq c_s\left(|X|, \tau, \lambda, \frac{\lambda|X|}{|y(X,Y)|}\right)$.

The algorithm's expected run-time is at most $c \cdot t_s(|X|, \tau) \cdot \frac{|y(X,Y)|}{|X|}$.

The proofs are deferred to Section 4.5 and Section 4.6. Combining the two algorithms from above, we obtain the following theorem.

Theorem 70. *Fix any $\lambda = o(1)$ and $\epsilon < 1$. There exists a randomized non-adaptive algorithm that solves the* LIS *problem with $(\alpha, \lambda n)$-approximation, where $\alpha = (1/\lambda)^{\sqrt{\epsilon}} \cdot (\log 1/\lambda)^{2^{O(\log^2(1/\epsilon)/\sqrt{\epsilon})}}$ using $\frac{1}{\lambda} \cdot n^{O(\sqrt{\epsilon}\log 1/\epsilon)}$ time (and hence samples from the input).*

The algorithm for Theorem 95 (see Algorithm 11) follows the outline from Figure 4. In particular, we first build a β-ary precision tree $T_{1/c}$ for $\beta = \Theta(\log n)$, and $c = c_s(n, n^\epsilon, \lambda, \lambda) = \frac{1}{\lambda} \cdot n^{O(\sqrt{\epsilon}\log 1/\epsilon)}$. We then apply the algorithm of Theorem 69 by interpreting the string as a Block-LIS instance with $k = 1$. The proof follows from recursive implementation of Theorems 68 and 69 with carefully chosen parameters. Most importantly τ and γ are carefully chosen, as a function of the other parameters, to balance approximation and complexity. Informally, we pick $\tau \approx |X|^\epsilon$ in each Block-LIS instance, and $\gamma \approx (k/\lambda)^{\sqrt{\epsilon}}$ for a recursion of depth $\approx 1/\sqrt{\epsilon}$, then we stop and use the dense estimator only by setting γ to be maximal. The formal proof involves tedious calculations, and is deferred to Section 4.8. We now complete Theorem 64, by instantiating the LIS problem with the parameters above set suitably.

Algorithm 11 EstimateLISMain

▷ Input: A sequence $y \in \mathbb{N}^n$, error parameter $\lambda \in (1/n, 1)$, and $\epsilon < 1$.

▷ Output: An integer $\widehat{L} \in [0, \mathsf{LIS}(y)]$.

$T \leftarrow$ PreprocessPrecisionTree$(y, \lambda \cdot n^{-\epsilon}, \beta)$, where $\beta \leftarrow \Theta(\log n)$.

return $\widehat{L} \leftarrow$ EstimateBlockLIS$(T, n^{\epsilon}, \lambda)$.

Figure 5: Description of the algorithm EstimateLISMain.

Proof of Theorem 64. Let $y \in \mathbb{N}^n$ be an input of the LIS problem. We solve the problem using the algorithm of Theorem 95 with inputs y, λ, and $\epsilon = 1/\log\log 1/\lambda$, noting that $\alpha = \lambda^{o(1)}$ and runtime complexity is $\frac{1}{\lambda} n^{O(1/\sqrt[3]{\log\log 1/\lambda})} = \frac{1}{\lambda} n^{o(1)}$. Finally, we invoke Fact 67 with $q = o(1)$ to obtain the claimed approximation (as $q = o(1)$, $p = q/(1-q) = o(1)$). □

4.4 Precision-Trees data structure

In this section we discuss the Precision-Trees data structure used to improve our bounds, and the properties of this data structure.

The basic construction of the β-ary tree $T_\lambda(e)$ was introduced in Section 4.2.5.1. Before proceeding, we establish the sample complexity of the Precision-Tree.

Lemma 71 (Sample complexity)**.** *The expected number of coordinates in* $[n]$ *that are sampled is* $\log^{O(\log_\beta n)}(\beta)/\lambda$.

Proof. Let $s_v = 1/\mathcal{P}_v$. Since we only sample leaves with $s_v \geq 1$, then the total sampled leaves is at most $\sum_{v \in V_{\log_\beta(n)}} s_v$.

Now, since $\mathbf{Z}_v$ are chosen i.i.d. uniformly in $\{1, \ldots, \Omega(\beta)\}$, then we deduce the recursive relation:

$$\mathbb{E}[s_v] \leq \mathbb{E}\left[\frac{1}{\mathcal{P}_v}\right] = \frac{O(\log \beta)}{\beta} \cdot \mathbb{E}\left[\frac{1}{\mathcal{P}_{\text{parent}(v)}}\right] \tag{4.2}$$

Let $\tau_\ell \triangleq \sum_{v \in V_\ell} s_v$. Then $\tau_0 = 1/\lambda$, and for level ℓ, summing and using Equation (4.2), we obtain

$$\mathbb{E}[\tau_\ell] = \mathbb{E}\left[\sum_{v \in V_\ell} s_v\right] \leq \beta \cdot \mathbb{E}\left[\sum_{v \in V_{\ell-1}} \frac{O(\log \beta)}{\beta} \cdot s_v\right] \leq O(\log \beta)\, \mathbb{E}\left[\sum_{v \in V_{\ell-1}} s_v\right] = O(\log \beta)\tau_{\ell-1}.$$

Then $\tau_\ell \leq O(\log \beta)^\ell/\lambda$, and we conclude $\mathbb{E}\left[\sum_{v \in V_{\log_\beta(n)}} s_v\right] = O(\log(\beta))^{\log_\beta n}/\lambda$ as needed. □

4.4.1 Tree sampling data structure

We equip each Precision-Tree with a *Sampling Oracle Data Structure* (denoted SODS) that should be seen as a data structure wrapper with access to Precision-Tree, allowing *efficient sampling* of leaves.

First, we show one can simulate uniform samples of elements given a Precision-Tree access.

Lemma 72 (Simulating Random Samples)**.** *Fix $\delta \leq 1$ and Precision-Tree $T = T_\delta(e_1, \ldots, e_n)$, as well as arbitrary $\delta' \geq \delta$, with $1/\delta' \in \mathbb{N}$. Using access to T only, we can generate a set of elements $S \subseteq \{e_1, \ldots, e_n\}$ such that the distribution of S is identical to the distribution where each e_i is included i.i.d. with probability $1/\delta' n$. The runtime to generate S is $O((\log \beta)^{\log_\beta n}/\delta)$.*

We remark that, while two subsequent invocations of the above lemma may give different sets S, S', each with the above distribution, they are dependent between each other.

Proof. The main task here is to show independence, i.e., that we can choose a set of i.i.d. samples by reusing the randomness of the Precision-Tree. We prove by induction on the tree height, starting with single node trees, that for any node with $\mathcal{P}_v \leq 1$, for any $\delta' \geq \mathcal{P}_v$ we can generate a subset of the leaves where each leaf is included with probability at least $1/\delta' n_v$, where n_v to be the number of leaves in the subtree rooted at v.

For leaves, if $\mathcal{P}_v \leq 1$, the node is included in the Precision-Tree, and we can subsample it with any required probability as needed. Now consider a non-leaf node v; by induction the statement holds for its children.

If $\delta' \leq \frac{4}{\beta}$, we are basically done since any child u of v, has score $\mathcal{P}_u \leq \frac{\mathcal{P}_v \beta}{4} \leq \frac{\delta' \beta}{4} \leq 1$, while the number of elements $n_u = \frac{n_v}{\beta}$, so we can simply subsample leaves in u's tree, each with probability $\frac{1}{\delta' n_v} = \frac{1}{\delta' \beta n_u} \leq \frac{1}{\mathcal{P}_u n_u}$ as needed (note that $\delta' n_v / n_u = \beta \delta' \geq \mathcal{P}_u$ as required by the inductive hypothesis).

It remains to show this for $\delta' > \frac{4}{\beta}$. For this case we claim we would like to use the tree-randomness to generate the samples.

For distributions P, Q, we say that P *stochastically dominates* Q (also denoted $P \succ Q$) if $\mathcal{F}_P(t) \leq \mathcal{F}_Q(t)$ for all t, with strict inequality for some t, where $\mathcal{F}_P$ and $\mathcal{F}_Q$ are the cumulative distribution functions (CDFs) of P and Q respectively. We first claim the following:

Claim 73. *Fix $k \in \mathbb{N}$, $0 < p < 1$ and $F \geq 1$ such that $F \cdot pk < 1/4$. Let*

$\mathbf{X}_1, \ldots, \mathbf{X}_k \sim_{i.i.d.} \texttt{Ber}(p)$ *(i.e., i.i.d. Bernoulli random variables with bias* p*) and let* $\mathbf{X} = \sum_i \mathbf{X}_i$*. Let* $\mathbf{Y}_1, \ldots, \mathbf{Y}_k \sim_{i.i.d.} \texttt{Ber}(2\,p\,F)$ *and let* $\mathbf{Y} = \mathbf{Y}^* \cdot \sum_i \mathbf{Y}_i$ *where* $\mathbf{Y}^* \sim \texttt{Ber}(1/F)$*, independent of the other variables. Then,* $\mathbf{Y}$ *stochastically dominates* $\mathbf{X}$*.*

Proof. $\mathbf{X}$ and $\mathbf{Y}$ are both supported on $\{0, 1, \cdots, k\}$. So, it suffices to show that $\Pr[\mathbf{Y} \geq i] > \Pr[\mathbf{X} \geq i]$ for all $i \in [k]$. Note that $2pF \leq 2pFk < 1/2$, so $2pF$ is a valid probability. For $i \in [k]$, let

$$f(i) := \frac{\Pr[\mathbf{Y} = i]}{\Pr[\mathbf{X} = i]} = \frac{\binom{k}{i} \cdot (2\,p\,F)^i \cdot (1 - 2\,p\,F)^{k-i}}{F \cdot \binom{k}{i} \cdot p^i \cdot (1-p)^{k-i}} = 2^i \cdot F^{i-1} \cdot \left(\frac{1 - 2\,p\,F}{1-p}\right)^{k-i}.$$

To show that $\Pr[\mathbf{Y} \geq i] > \Pr[\mathbf{X} \geq i]$ for all $i \in [k]$, it suffices to show that $f(i) > 1$ for all $i \in [k]$. Using the conditions that $2pF < 1/2$, $pFk < 1/4$ and $F \geq 1$, along with the fact that $1 - x \geq 2^{-2x}$ for all $x \in [0, 1/2]$, we obtain:

$$f(i) > 2(1 - 2\,p\,F)^k \geq 2 \cdot 2^{-4pFk} > 2 \cdot 2^{-1} = 1,$$

for all $i \in [k]$. This concludes the proof. □

We show that one can simulate sub-sampling of each leaf of u i.i.d. with probability $1/\delta' n_v$ by the following process. If $\mathcal{P}_u > 1$, then output no elements. Otherwise, we can (by the inductive hypothesis) sub-sample each leaf of u i.i.d. with probability $1/n_u$.

Now, let

$$F \triangleq \frac{1}{\lfloor 1/\delta' \rfloor}\frac{\beta}{4} \geq \frac{1}{\lfloor 1/\mathcal{P}_v \rfloor}\frac{\beta}{4} \overset{Eq.\ 4.1}{\geq} \frac{1}{\Pr_T[\mathcal{P}_u \leq 1]}.$$

This process generates μ i.i.d. leaves of u with distribution

$$\mu \sim \texttt{Ber}(\Pr_T[\mathcal{P}_u \leq 1]) \cdot \sum_{l \text{ leaf of } u} \texttt{Ber}(\tfrac{1}{n_u}) \succ \texttt{Ber}(\tfrac{1}{F}) \cdot \sum_{l \text{ leaf of } u} \texttt{Ber}(\tfrac{1}{n_u}).$$

Noting that $n_u = \frac{n_v}{\beta} \geq \frac{\delta' n_v}{2F}$ and $\frac{n_u F}{\delta' n_v} \leq \frac{1}{4}$, we invoke Claim 73 to obtain that the distribution μ *stochastically dominates* the required sample-size distribution $\sum_{l \text{ leaf of } u} \texttt{Ber}(\frac{1}{\delta' n_v})$, and hence we can simulate the distribution we need over leaves of each child, and hence also for v.

For run-time, we note that this process takes time (at most) proportional to the size of the tree. For the latter, by Lemma 71, the time is $O((\log \beta)^{\log_\beta n}/\delta)$.

□

Next, we show that if each element of the tree is a block of integers, then one can construct a data structure that sub-samples with *interval range restriction*, in time proportional to the sample size.

Corollary 74 (Conditional sub-sampling data structure)**.** *Fix Precision-Tree $T = T_\delta(y)$ with $y \in \mathbb{N}^{n \times k}$. There exists a data structure, that given any interval $Y \in \mathcal{I}$ and sub-sampling probability $\eta \leq 1/\delta n$ with $1/\eta \in E_2(n)$, sub-sample block-coordinates $X' \subseteq [n]$ i.i.d. with probability η and outputs a set of all coordinates $W \subseteq X' \times [k]$ such that $y_W \in Y$, i.e. $W \triangleq \cup_{i \in X'}\{(i,j) \mid y_{i,j} \in Y\}$, in time $\tilde{O}(|W|)$. The preprocessing time is, in expectation, $\tilde{O}\left(\frac{n+\|m\|_1}{\delta n}\right)$, where $m_i \leq k$ is the number of non-null elements in y_i.*

Proof. To preprocess, we prepare for each possible $\eta \leq 1/\delta n$ with $1/\eta \in E_2(n)$ by sub-sampling each block i.i.d. with probability η using Lemma 72. Then we compute a set of coordinates $Z_\eta \subset [n]$ by combining all coordinates from all sampled blocks. Let $U_\eta \triangleq \cup_{w \in Z_\eta \times [k]}\{y_w\}$. Compute U_η and store it as a sorted array with a pointer to $y^{-1}(j) \cap (Z_\eta \times [k])$ for each $j \in U_\eta$. Now, for each (η, Y) query, locate the range of elements in U_η containing precisely the elements in Y using binary search of

$\min(Y), \max(Y)$. This will be our output W.

The correctness follows immediately from the construction. Preprocessing runtime is as claimed as, in expectation, we need to sort $\|m\|_1/\delta n$ elements, and there are $O(\log n)$ different values of η to consider. Runtime is only $O(\log n)$ plus the size of the output (as it is stored as contiguous block). □

4.4.2 Decomposing Precision-Trees

We now show that if one can compute some "local" function over intervals of elements given some local Precision-Tree parameter access to each interval, then we can redefine the tree-access as a tree-access to the global problem assuming all local problems were computed successfully. For the analysis, we assume that the number of leaves is an integral power of β. The results extend to general tree sizes (for example by padding the input).

The algorithm follows the following steps (the full algorithm is described in Algorithm 12):

1. Identify the "correct" level ℓ in T where each node corresponds to the right number of elements.

2. Find all nodes v at level ℓ with enough precision to compute the local function f_v and compute the local function on these nodes by "detaching" the sub-trees rooted at each v.

3. Consider the global tree T, trimmed to levels $\leq \ell$ (i.e., the *top* of the tree). Augment the precision parameters by dividing each score with the precision

we need to use for the local computation. This creates another "simulated" precision tree where we have access to the value at all leaves with $\mathcal{P}_v \leq 1$.

4. Compute the global function g with tree access to the locally computable values.

Overall, we constructively show the following:

Lemma 75 (Tree Decomposition)**.** *Fix $\beta, \tau, n \in \mathbb{N}$ such that n, τ are each a power of β. Fix a domain of elements $e \triangleq \{e_1, \ldots, e_n\} \in \mathcal{E}^n$ and let $X_1, \ldots, X_\tau$ be a partition of $[n]$ into disjoint intervals of equal width. Fix $\delta, \eta \in (0,1)$ and (possibly randomized) functions $g : T_\delta(\mathcal{F}^\tau) \to \mathcal{G}$ and $\{f_i : T_\eta(e(X_i)) \to \mathcal{F}\}_{i\in[\tau]}$ for some spaces $\mathcal{G}$ and $\mathcal{F}$. Then, algorithm TrimTree (see Algorithm 12), given a Precision-Tree $T_{\delta\eta}(e)$, outputs $g(T_\delta(f_1(T_\eta(e(X_1))), \ldots, f_\tau(T_\eta(e(X_\tau)))))$. The algorithm's expected run-time is $\tilde{O}(\tau + t_g + \frac{1}{\delta\tau} \cdot \|t_f\|_1 \cdot \log^{O(\log_\beta(\tau))}(\beta))$, where t_g is the time to compute g, and $(t_f)_i$ is the expected time to compute f_i.*

Proof of Lemma 75. Consider the full Precision-Tree $T_{\delta\eta}(e)$. Take the top $\log_\beta \tau$ levels of the tree to get to the level where each node v has n/τ leaves in its subtree. This top portion (trimmed tree) can be seen as T_δ as follows: the precision $\mathcal{P}'_v$ in T_δ is simply defined to be $\mathcal{P}_v/\eta$. Hence for leaf v in T_δ, we have that $\mathcal{P}'_v \leq 1$ iff $\mathcal{P}_v \leq \eta$. Hence for each such v, if $\mathcal{P}'_v \leq 1$, we have another T_η rooted at v (in the full Precision-Tree), allowing us to compute the corresponding $f_i(T_\eta(e(X_i)))$. Hence we will be able to also compute $g(T_\delta(f_1(T_\eta(e(X_1))), \ldots, f_\tau(T_\eta(e(X_\tau)))))$ as required.

For runtime, we note that by Lemma 71, we have $\frac{1}{\delta} \cdot \log^{O(\log_\beta(\tau))}(\beta)$ leaves v with $\mathcal{P}'_v \leq 1$. We only preprocess SODS and compute the function f_i on those leaves, and all f_i have the same probability to be computed. Also, we note from the independence of $\mathbf{Z}_i$'s, once we fix precision parameter η, the time to compute each $f_i(T_\eta(e(X_i)))$

is independent of the random choice of i where f_i is computed, hence we have the expected runtime is the product of expectations. Finally, computing the function g takes t_g time.

□

Algorithm 12 TRIMTREE

Input: Disjoint consecutive intervals of $[n]$: $\{X_i\}_{i\in[\tau]}$ of length r each, global function $g : T_\delta(\mathcal{F}^\tau) \to \mathcal{G}$, functions $\{f_i : T_\eta(e(X_i)) \to \mathcal{F}\}_{i\in[\tau]}$, Precision-Tree T, local precision parameter η.

Output: $g([f_i(T_\eta(e(X_i)))]_{i\in[\tau]})$

Let $\ell = \log_\beta \tau$ be the level in T such that for $v \in V_\ell$, $|X_v| = r$.

$T_g \leftarrow$ the induced sub-tree of $V_{\le\ell}$.

for $v \in V_\ell$ **do**

  if $\mathcal{P}_v \le \eta$ **then**

    Preprocess SODS for the precision tree rooted at v using Corollary 74.

    Compute $f_i(T_\eta(e(X_i)))$ using sub-tree rooted at v and store the result in v.

  end if

end for

$\mathcal{P}_v \leftarrow \mathcal{P}_v/\eta$ for all $v \in T_g$.

Preprocess SODS for T_g using Corollary 74.

Compute $q \leftarrow g([f_i(T_\eta(e(X_i)))]_{i\in[\tau]})$ using T_g.

return q.

Figure 6: Algorithm TRIMTREE for decomposing Precision-Trees (Lemma 75). Define level ℓ_v of a node v as its distance to the root. We define V_ℓ as the set of nodes of level ℓ, i.e., $V_\ell \triangleq \{v \in T \mid \ell_v = \ell\}$. We use the shorthand $V_{\le\ell} \triangleq \cup_{m\le\ell} V_m$ and similarly for $V_{<\ell}, V_{>\ell}, V_{\ge\ell}$.

Algorithm 13 PREPROCESSPRECISIONTREE

Input: An input $e \in \mathcal{E}^n$, precision parameter δ, and branching parameter $\beta \in \mathbb{N}$.

Output: Precision-Tree data-structure $T_\delta(e)$.

$T \leftarrow$ PREPROCESSNODE$(e, [|e|], \delta, \beta)$.

Preprocess SODS for T using Corollary 74.

return T.

Algorithm 14 PREPROCESSNODE

Input: An input $e \in \mathcal{E}^n$, an interval $I \subseteq [|e|]$, precision parameter δ, and branching parameter $\beta \in \mathbb{N}$.

Output: Precision-Tree data-structure $T_\delta(e)$.

$v \leftarrow$ a new node associated with e_I.
if $\delta \leq 1$ **then**
  if $|I| = 1$ **then**
    Store e_I in v.
  else
    $I_1, \ldots, I_\beta \leftarrow$ a partition of I into β disjoint, equal length intervals.
    for $i \in [\beta]$ **do**
      $\mathbf{Z}_i \sim$ Uniform$(\{1, 2, 3, \ldots, \beta/4\})$.
      $\mathcal{P}_i \leftarrow \delta \cdot \mathbf{Z}_i$.
      $u_i \leftarrow$ PREPROCESSNODE$(e, I_i, \mathcal{P}_i, \beta)$.
      Add u_i as a child of v.
    end for
  end if
end if
return v.

Figure 7: Description of the algorithm PREPROCESSPRECISIONTREE.

4.5 Algorithm for Genuine-LIS: proof of Theorem 68

In this section we describe and analyze the algorithm for the Genuine-LIS problem, which is called EstimateGenuineLIS.

First, we introduce an extension to the Block-LIS algorithm, which we use for our algorithm. In particular, we allow it to ignore a few "heavy blocks" which will not affect the approximation or complexity guarantees of Theorem 68. We use the following definition:

Definition 76. *Fix block sequence $y \in \mathbb{N}^{X\times[k]}$, parameter $\delta < 1$ and value range interval $Y \in \mathcal{I}$. Let X^δ be the $\delta|X|$ blocks in X for which the quantity $|y(X^\delta) \cap Y|$ is maximized. The* δ-heavy trimmed input *$y_{-\delta,Y} \in \mathbb{N}^{X\times[k]}$ is y after setting $y_{-\delta,Y}(X^\delta) = \emptyset$ (i.e., removing all integers from $y(X^\delta)$).*

For $X' \subseteq X$ and a range of y-values Y', we also define $y_{-\delta}(X', Y') \triangleq y(X')_{-\delta,Y'\cap Y'}$ (i.e., the multiset of values in $y(X')_{-\delta,Y'}$ restricted to Y').

The stronger version of Theorem 68 is as follows:

Lemma 77 (EstimateGenuineLIS algorithm, extended)**.** *Fix integers n, k and $\lambda \in (1/n, 1)$. Fix an instance $g \in (\mathbb{N}\times\{0,1\})^{n\times k}$. For some monotone functions $a_s : \mathbb{R}_+^3 \to [1,\infty)$ and $c_s : \mathbb{R}_+^4 \to [1,\infty)$, suppose there exists a randomized algorithm $\mathcal{A}_{BL}$ that, given $X \subseteq [n], Y \subseteq \mathbb{N}$, parameter τ_s, and a Precision-Tree $T_{1/c_s\left(|X|,\tau_s,\lambda,\frac{\lambda\cdot|X|}{|y_{-\lambda/2}(X,Y)|}\right)}(y)$ for some $y \in \mathbb{N}^{n\times k}$, can produce an $(\alpha_s, \lambda|X|)$-approximation for Block-LIS(y, X, Y) where $\alpha_s = a_s\left(|X|, \tau_s, \frac{\lambda\cdot|X|}{|y_{-\lambda/2}(X,Y)|}\right)$ in time t_s, w.h.p.*

Fix any parameter $\gamma, \tau \geq 1$. Then, there exists an algorithm $\mathcal{A}$ that, given complete direct access to $g(*)_1 \in \mathbb{N}^{n\times k}$ and Precision-Tree access to $g(*)_2 \in \{0,1\}^{n\times k}$, $T_{1/c}(g(*)_2)$, produces a $(\alpha_g, \lambda n)$-approximation for Genuine-LIS(g) w.h.p., where $c \geq$

$\tilde{O}(1/\lambda) + c_s(n, \tau, \lambda, \gamma\lambda/k)$ *and* $\alpha_g = \log(k/\lambda) \cdot \max\{\gamma, a_s(n, \tau, \gamma\lambda/k)\}$. *The algorithm* $\mathcal{A}$ *runs in time* $\tilde{O}(nk) + O(t_s)$.

To prove Lemma 77, we use the following reduction:

Lemma 78. *Suppose there exists an algorithm* $\mathcal{A}$ *that* $\left(a\left(\frac{\lambda|X|}{|y(X,Y)|}\right), \lambda n\right)$*-approximates* Block-LIS *w.h.p. given tree access* $T_{1/c}(g)$*, in time* $t = t\left(\frac{\lambda|X|}{|y(X,Y)|}\right)$*, where* $c = c\left(\frac{\lambda|X|}{|y(X,Y)|}\right)$ *and* a, c, t *are some functions. Then there exists an algorithm* $\mathcal{A}'$ *that* $\left(O\left(a\left(\frac{\lambda|X|}{|y_{-\lambda/2}(X,Y)|}\right)\right), \lambda n\right)$*-approximates* Block-LIS *with tree access* $T_{1/c\left(\frac{\lambda|X|}{|y_{-\lambda/2}(X,Y)|}\right)}$ *in time* $O(t) + \tilde{O}(1/\lambda)$ *w.h.p.*

The proof of Lemma 78 is deferred to Section 4.7.

Proof of Theorem 68 using Lemmas 77 and 78. The proof is immediate since the extended algorithm assumed in Lemma 77 reduces to the standard versions using Lemma 78. □

The rest of this section is devoted to prove Lemma 77.

4.5.1 Algorithm description

The algorithm EstimateGenuineLIS is presented in Algorithm 15. It consists of the following high-level steps. First, we perform some preprocessing by extracting *pseudo-solutions*, longest increasing subsequences, ignoring the genuineness flags. Then we partition these sequences according to their lengths. Next we estimate the proportion of genuine elements amongst the union of above subsequences; this gives our first (dense) estimator. We also employ a second (sparse) estimator, when the proportion of genuine elements is too low, to reduce the instance to Block-LIS with improved parameters. We now describe these steps in detail.

4.5.1.1 Extracting pseudo-solutions

For this step, we treat all elements the same, whether they are genuine or not, and greedily compute and extract disjoint increasing subsequences, each of length $\gtrapprox \lambda n$, denoted $\{P_i\}_i$ (which are called *pseudo-solutions*), until there is no longer an increasing subsequence of length λn remaining. For reasons that will become clear later, we also need to guarantee that each time we extract a sequence, such a sequence is approximately proportional to the longest one at that time, i.e., *after removing all previous sequences*, and therefore we use a greedy algorithm.

We note that we cannot afford to use the standard dynamic-programming solution for finding an optimal LIS each time, as the resulting runtime is too large for us. To improve the runtime, we use instead a fast data-structure for approximate greedy extraction, based on the fully dynamic data-structure in [101].

Theorem 79. *[101] Given an input $x \in \mathbb{N}^n$, for any $\epsilon > 0$, there exists a* dynamic *data-structure $\mathcal{D}$ with the following operations: 1) inserting/deleting an element, in time $(\log n/\epsilon)^{O(1)}$, and 2) finding an approximate longest increasing sub-sequence of length within $(1+\epsilon)$-factor of the optimum, OPT, in time $O(|OPT|)$.*

This theorem with $\epsilon = 1$ leads to the following corollary for Block-LIS.

Corollary 80 (Extracting Pseudo-solution, approximate)**.** *There exists a data-structure that given a sequence of integer blocks $y \in \mathbb{N}^{n \times k}$, extracts (i.e., finds and removes) an increasing sequence of length at least $|OPT|/2$, where OPT is the current LIS. The pre-processing time is $\tilde{O}(n \cdot k)$.*

Proof. To preprocess the data-structure, we insert all elements to $\mathcal{D}$ one by one, where elements of each block are inserted in non-increasing order. This takes time $\tilde{O}(nk)$.

Finally, we answer the extraction queries by LIS-querying $\mathcal{D}$, and remove all elements in the 2-approximate LIS solution one by one from $\mathcal{D}$. □

4.5.1.2 Partitioning the pseudo-solutions

Next, we partition $\{P_i\}_i$ into $\log(1/\lambda)$ disjoint buckets denoted $\{\mathcal{P}_\ell\}_\ell$, where for $\ell \in E_2(1/\lambda)$, each $P \in \mathcal{P}_\ell$ is an increasing sub-sequence of length $\approx \ell\lambda n$ (again, ignoring the genuineness of the elements). Our algorithm generates an estimator for Genuine-LIS for each scale ℓ, based purely on the elements of the subsequences $\{P \mid P \in \mathcal{P}_\ell\}$, and outputs the maximal estimator. Clearly, one of the buckets must contain a significant fraction of the Genuine-LIS; we focus on that bucket $\mathcal{P}_\ell$ henceforth.

4.5.1.3 Estimator Computation

We next describe our estimator for each bucket as above, which by itself is generated by taking the maximum of 2 estimators, denoted *Dense Estimator* and *Sparse Estimator*. The *Dense Estimator* is a fairly straightforward one. Here, we consider the union of all pseudo-solutions $U_\ell \triangleq \cup_{P\in\mathcal{P}_\ell} P$, and estimate the rate of genuine elements in U_ℓ by checking the genuineness of a random sample of elements in U_ℓ. The dense estimator is then proportional to the estimated rate of genuine elements in U_ℓ. The main idea is that, if U_ℓ has many genuine elements, then the aforementioned sampling procedure gives a high-fidelity estimate.

The sparse estimator handles the opposite situation: when the genuine elements are sparse within the union U_ℓ of subsequences. Then, if the Genuine-LIS itself is quite large, it must be the case that a high fraction of the genuine elements participate in the

LIS. Here, we invoke the algorithm for Block-LIS, restricted to genuine elements only. Using the fact that Block-LIS approximation is a function of the proportion of LIS elements among all the "relevant" elements (in this case, genuine elements), we make progress in the sparse case by reducing the number of relevant elements, resulting in improved approximation. The overall estimator for length ℓ is the maximum of the 2 estimators.

We highlight the fact that the sparse estimator calls the assumed algorithm $\mathcal{A}_{\mathrm{BL}}$ on the input string g, restricted only to the genuine elements. In this call, we do not compute the genuineness flags explicitly, but rather compute them as needed, i.e., only for the elements that are "read" by the $\mathcal{A}_{\mathrm{BL}}$ algorithm, and if an element turns out to be non-genuine it becomes a "null" entry. Since we are using Precision-Tree to access samples, the algorithm will perform computation in a different (but equivalent) order: first computing the genuineness of the right samples, and then run the $\mathcal{A}_{\mathrm{BL}}$ algorithm on those samples.

Algorithm 15 EstimateGenuineLIS

Input: $g \in (\mathbb{N} \times \{0,1\})^{n \times k}, \lambda \in [1/n, 1], \gamma \geq 1$, Precision-Tree $T = T_{1/c}(g(*)_2)$ for the parameter c defined in Lemma 77.

Output: A number $\widehat{G} \in [0, n]$.

Greedily and iteratively extract 2-approximate maximal *increasing sequences* from $g(*)_1$, as long as they contain at least $\lambda n/4$ elements using Corollary 80, generating pseudo-solutions $P_1, \ldots, P_t$ for some $t = O(k/\lambda)$, where each $P_i \subseteq [n] \times [k]$ is a set of coordinates.

Sample blocks $S \subseteq [n]$ i.i.d. with probability $10\zeta \cdot \frac{1}{\lambda} \cdot \frac{1}{n}$ each, for $\zeta = O(\log n)$, using Corollary 74 and T.

Compute $W \leftarrow \{(i,j) \in S \times [k] \mid g(i,j)_2 = 1\}$ the set of genuine (sampled) coordinates.

for $\ell \in E_2(4/\lambda)/4$ **do**

Let $\mathcal{P}_\ell \triangleq \{P_i : |P_i| \in (\lambda\, n\, \ell/2, \lambda\, n\, \ell]\}$, and let $U_\ell \triangleq \cup_{P \in \mathcal{P}_\ell} P$.

Compute $B_\ell \subseteq S \leftarrow$ the ζ blocks in S for which $|(B_\ell \times [k]) \cap W \cap U_\ell|$ is maximized.

$\kappa_\ell \leftarrow |W \cap U_\ell \cap ((S \setminus B_\ell) \times [k])|$.

end for

return

$$\widehat{G} \triangleq \max_{\ell} \max \left\{ \tfrac{\lambda^2\, \ell\, n\, \kappa_\ell}{100 \cdot k \cdot \zeta} - \tfrac{\lambda n}{4}, \mathcal{A}_{\mathrm{BL}}\left(g(*)_1 \mid_{U_\ell \cap (g(*)_2)^{-1}(1)}, \mathbb{N}, \lambda, T \right) \right\}.$$

Figure 8: Description of the algorithm EstimateGenuineLIS. $g(*)_1 \mid_{U_\ell \cap (g(*)_2)^{-1}(1)}$ denotes the sequence $g(*)_1$ restricted to positions in U_ℓ for which $g(\cdot)_2 = 1$. Recall that $E_2(k) \triangleq \{2^i \mid i \in \{0, \ldots, \lfloor \log k \rfloor\}\}$. $\mathcal{A}_{BL}$ is the assumed algorithm for Block-LIS.

4.5.2 Analysis

We now analyze the algorithm, proving Lemma 77.

Proof of Lemma 77. Let OPT be the coordinates of an optimal solution of length $\textsf{Genuine-LIS}(g)$. Let $U \triangleq \cup_\ell U_\ell$, and $\overline{U} \triangleq [n] \times [k] \setminus U$. Define $\widehat{\mathsf{OPT}} \triangleq \mathsf{OPT} \cap U$ (i.e., restricted to elements in U). Since $\overline{U}$ does not contain an increasing sequence of length $\frac{1}{2}\lambda n$, we have that $|\widehat{\mathsf{OPT}}| \geq |\mathsf{OPT}| - |\mathsf{OPT} \cap \overline{U}| \geq |\mathsf{OPT}| - \frac{1}{2}\lambda n$. For $\ell \in E_2(4/\lambda)$, let us also define $\widehat{\mathsf{OPT}}_\ell \triangleq \mathsf{OPT} \cap U_\ell$.

To bound our estimators, we first bound the quantity κ_ℓ. For this task, we introduce more notation. Let $M \subset [n]$ be the $\lambda n/2$ blocks containing the highest number of genuine elements. We also define the following:

$$
\begin{aligned}
\widehat{\kappa_\ell} &\leftarrow |W \cap U_\ell|. \\
q^{(\ell)} &\triangleq \tfrac{1}{nk} \cdot |U_\ell \cap (g(*)_2)^{-1}(1)|. \\
q^{(\ell)}_{-\lambda/2} &\triangleq \tfrac{1}{nk} \cdot |U_\ell \cap (g(*)_2)^{-1}(1) \cap (([n] \setminus M) \times [k])|. \\
m^{(\ell)}_{-\lambda/2} &\triangleq \tfrac{1}{k} \cdot \min_{i \in M} \{|U_\ell \cap (g(*)_2)^{-1}(1) \cap (i, *)|\}. \\
G_{\ell,1} &\triangleq \tfrac{\lambda^2 \ell n \kappa_\ell}{100 \cdot k \cdot \zeta} - \tfrac{\lambda n}{4}. \\
G_{\ell,2} &\triangleq \text{ the output of } \mathcal{A}_{\mathrm{BL}}\left(g(*)_1 \mid_{U_\ell \cap (g(*)_2)^{-1}(1)}, \mathbb{N}, \lambda, T\right).
\end{aligned}
$$

We have

$$\mathbb{E}\left[\widehat{\kappa_\ell}\right] = q^{(\ell)} \cdot k \cdot \tfrac{1}{\lambda} \cdot 10\zeta = q^{(\ell)} \cdot \tfrac{10k \cdot \zeta}{\lambda}.$$

We need to have a good estimator w.h.p. and therefore use the following statistical fact:

Fact 81. *Fix $\zeta \in \mathbb{N}$. Let $\mathbf{X}_1, \ldots, \mathbf{X}_m$ be non-negative independent random variables and let $\mathbf{X} = \sum_i \mathbf{X}_i$. Let $\mathbf{Y}$ be the sum of the ζ largest (empirical) $\mathbf{X}_i$. Then, $\mathbf{X} - \mathbf{Y} \leq 2\,\mathbb{E}[\mathbf{X}]$ with probability $1 - \exp(-\Omega(\zeta))$.*

Proof. Define $p_i \triangleq \Pr[\mathbf{X}_i > 2\,\mathbb{E}[\mathbf{X}]/\zeta]$, and notice that $\sum_i p_i < \zeta/2$ using Markov's inequality. Define also $\mathbf{Z}_i \triangleq \max\{0, \mathbf{X}_i - 2\,\mathbb{E}[\mathbf{X}]/\zeta\}$, and $\mathbf{Z} = \sum_i \mathbf{Z}_i$. Then $\mathbf{X} - \mathbf{Z}$ is a sum of non-negative independent random variables bounded by $2\,\mathbb{E}[\mathbf{X}]/\zeta$, and hence, by the Chernoff bound (multiplicative form), with probability at most $\exp(-\Omega(\zeta))$, we have $\mathbf{X} - \mathbf{Z} > 2\,\mathbb{E}[\mathbf{X}] \geq \left(1 + \frac{\mathbb{E}[\mathbf{X}]}{\mathbb{E}[\mathbf{X}-\mathbf{Z}]}\right) \cdot \mathbb{E}[\mathbf{X} - \mathbf{Z}]$. Also, notice that $\Pr[\mathbf{Z}_i > 0] = p_i$, and hence, with probability at most $\exp(-\Omega(\zeta))$, $\sum_i \mathbf{1}[\mathbf{Z}_i > 0] > \zeta \geq \sum_i p_i + \zeta/2$, implying that $\mathbf{Z} > \mathbf{Y}$ with this probability as well. By the union bound, we conclude that $\mathbf{X} - \mathbf{Y} \leq 2\,\mathbb{E}[\mathbf{X}]$ with probability $1 - \exp(-\Omega(\zeta))$ as needed. □

Now, notice that $\widehat{\kappa_\ell}$ is a sum of independent random variables and hence from Fact 81, w.h.p. we have

$$\kappa_\ell \leq 2\,\mathbb{E}[\widehat{\kappa_\ell}] = 20q^{(\ell)} \cdot \tfrac{k \cdot \zeta}{\lambda}.$$

For the lower bound, we derive a lower bound for κ_ℓ using $q^{(\ell)}_{-\lambda/2}$. Notice that each $i \in M$ gets sampled i.i.d. with probability $\frac{10\zeta}{\lambda n}$. Hence $|S \cap M| > 2\zeta$ w.h.p, which implies that $|(S \setminus B_\ell) \cap M| > \zeta$. This means that the contribution of blocks in M to κ_ℓ is at least $\zeta \cdot k \cdot m^{(\ell)}_{-\lambda/2}$. Next, consider the contribution ψ_ℓ of blocks in $[n] \setminus M$ to κ_ℓ. Then ψ_ℓ is a sum of independent random variables in $k \cdot [0, m^{(\ell)}_{-\lambda/2}]$ and hence, using the Chernoff bound, we have w.h.p.

$$\psi_\ell \geq 0.9\,\mathbb{E}\left[\psi_\ell\right] - \zeta \cdot k \cdot m^{(\ell)}_{-\lambda/2} = \tfrac{9\zeta}{\lambda} \cdot k \cdot q^{(\ell)}_{-\lambda/2} - \zeta \cdot k \cdot m^{(\ell)}_{-\lambda/2}.$$

Therefore $\kappa_\ell \geq \psi_\ell + \zeta \cdot k \cdot m^{(\ell)}_{-\lambda/2} \geq \frac{9\zeta}{\lambda} \cdot k \cdot q^{(\ell)}_{-\lambda/2}$ w.h.p. as well. Hence we can

finally bound $G_{\ell,1}$, w.h.p., as follows:

$$G_{\ell,1} = \frac{\lambda^2 \ell n \kappa_\ell}{100 \cdot k \cdot \zeta} - \frac{\lambda n}{4} \in \left[\frac{9}{100} \lambda n \ell \cdot q^{(\ell)}_{-\lambda/2} - \frac{\lambda n}{4}, \frac{1}{4} \lambda n \ell \cdot q^{(\ell)} \right].$$

For the second estimator, we have that $G_{\ell,2}$ is a $\left(a_s\left(n, \tau, \frac{\lambda n}{|U_\ell \setminus (M \times [k]) \cap g(*)^{-1}(1)|}\right), \lambda n\right)$-approximation of

$\mathsf{Genuine\text{-}LIS}(g(U_\ell))$.

We now bound the maximum over these quantities.

4.5.2.1 Upper Bound

W.h.p., for all ℓ, we have $G_{\ell,1} \leq \frac{1}{4} \cdot \lambda \, n \, \ell \, q^{(\ell)}$. Now, by construction, all pseudo-solutions $P \in \mathcal{P}_\ell$ are of length $|P| \geq \lambda \, n \, \ell / 4$, and hence $|\mathcal{P}_\ell| \leq \frac{4k}{\lambda \ell}$. Therefore, there must exist some increasing sub-sequence $P^* \in \mathcal{P}_\ell$ such that $|P^* \cap (g(*)_2)^{-1}(1)| \geq \frac{1}{4} \cdot \lambda \, n \, \ell \, q^{(\ell)}$, i.e., that contains at least that many genuine elements. This implies $|\widehat{\mathsf{OPT}}_\ell| \geq G_{\ell,1}$.

Similarly, for $G_{\ell,2}$, the $\mathsf{Block\text{-}LIS}$ approximation algorithm outputs a lower bound on the LIS, and hence $G_{\ell,2} \leq |\widehat{\mathsf{OPT}}_\ell|$.

We conclude that $\max_\ell \max\{G_{\ell,1}, G_{\ell,2}\} \leq \max_\ell |\widehat{\mathsf{OPT}}_\ell| \leq |\mathsf{OPT}|$.

4.5.2.2 Lower Bound

Since $\widehat{\mathsf{OPT}} = \cup_\ell \widehat{\mathsf{OPT}}_\ell$, there exists $\ell^* \in E_2(4/\lambda)$ such that $|\widehat{\mathsf{OPT}}_{\ell^*}| \geq \frac{|\widehat{\mathsf{OPT}}|}{\log(4/\lambda)}$. Fix $U^* \triangleq U_{\ell^*} \setminus (M \times [k])$, $q^* = q^{(\ell^*)}_{-\lambda/2}$, $\widehat{\mathsf{OPT}}^* \triangleq \widehat{\mathsf{OPT}}_{\ell^*}$, $G^*_1 = G_{1,\ell^*}$ and $G^*_2 = G_{2,\ell^*}$.

Consider the following two cases:

Dense Case: First, suppose $q^* \geq \frac{1}{\gamma}$. Observe that from Corollary 80, we may only extract a solution of length at most $\lambda n \ell^*$ once the remaining LIS is at most $2\lambda n \ell^*$

(i.e., after extracting longer subsequences). Hence $|\mathsf{OPT}_{\ell^*}| \leq 2\lambda n\ell^*$. Therefore,

$$G_1^* \geq \frac{\lambda n\ell^*}{20\gamma} - \frac{1}{4}\lambda n \geq \frac{|\widehat{\mathsf{OPT}}^*|}{40\gamma} - \frac{1}{4}\lambda n \geq \frac{|\widehat{\mathsf{OPT}}|}{40\gamma\log(4/\lambda)} - \frac{1}{4}\lambda n > \frac{|\mathsf{OPT}|}{40\gamma\log(4/\lambda)} - \lambda n.$$

Sparse Case: Now, suppose $q^* < \frac{1}{\gamma}$. Then $|U^* \cap (g(*)_2)^{-1}(1) \cap (([n] \setminus M) \times [k])| < \frac{nk}{\gamma}$ and hence for any $\tau = \tau_s$ such that $T = T_{1/c}(g(*)_2)$, we have

$$G_2^* \geq \frac{|\mathsf{OPT}^*|}{\alpha} - \lambda n \geq \frac{|\mathsf{OPT}|}{\alpha\log(4/\lambda)} - \lambda n,$$

where

$$\alpha = a_s\left(n, \tau, \tfrac{\lambda n}{|U^* \cap g(*)^{-1}(1) \cap (([n]\setminus M)\times[k])|}\right) \leq a_s\left(n, \tau, \tfrac{\lambda\gamma}{k}\right).$$
$$c = c_s\left(n, \tau, \lambda, \tfrac{\lambda n}{|U^* \cap g(*)^{-1}(1) \cap (([n]\setminus M)\times[k])|}\right) \leq c_s\left(n, \tau, \lambda, \tfrac{\lambda\gamma}{k}\right).$$

Since we are either in the *dense* or *sparse* case as described above, we obtain that

$$\max\{G_1^*, G_2^*\} \geq \frac{|\mathsf{OPT}|}{A\log(1/\lambda)} - \lambda n,$$

where $A = \max\left\{O(\gamma), a_s\left(n, \tau, \frac{\lambda\gamma}{k}\right)\right\}$.

We conclude that $\widehat{G} = \max\{G_{\cdot,\cdot}\}$ is a $(A\log(1/\lambda), \lambda n)$-approximation for $|\mathsf{OPT}|$, as needed.

Runtime Complexity: The algorithm's runtime is dominated by the first step, i.e., generating increasing subsequences, which requires $\tilde{O}(nk)$ time using Corollary 80. In addition, we require t_s time for computing the second estimator. Thus, the overall algorithm for EstimateGenuineLIS runs in time $\tilde{O}(nk) + t_s$. □

4.6 Algorithm for Block-LIS: proof of Theorem 69

We now present our construction for the Block-LIS problem, and prove Theorem 69. Recall that we are given an interval $X \subseteq [n]$, a sequence $y \in \mathbb{N}^{X\times[k]}$ and a range of

values $Y \subset \mathbb{N}$, and the goal is to approximate Block-LIS(y, X, Y), i.e., the length of a maximal sub-sequence OPT, specified by a set of indices $w_1, w_2, \ldots, w_\ell \subseteq X \times [k]$ such that the set of (first coordinate, value) pairs $\{((w_i)_1, y_{w_i})\}_{i \in [\ell]}$ is a subset of $X \times Y$, and is a monotone set.

Our construction, at a high level, reduces the Block-LIS problem above to a Genuine-LIS instance over τ blocks, where τ is the branching factor of Theorem 69, whose value is given as input and controls the delicate balance between approximation and complexity. The first coordinates of the Genuine-LIS instance are computed directly, while each genuineness flag corresponds to a Block-LIS instance of a block interval X_i of some smaller size $r \triangleq |X|/\tau$, and restricted to a y-interval $Y' \subseteq Y$. We consider the pair X_i, Y' as genuine only if the (recursive) approximation of Block-LIS(y, X_i, Y') is above a certain *threshold* κ. For each x-interval X_i, we compute a set of y-intervals $\mathcal{Y}_i$ (called *candidate intervals*), which together form a *block* in the Genuine-LIS instance. Intuitively, one can think of the Genuine-LIS instance as solving the *global LIS*, while each Block-LIS instance is solving some *local LIS* over (sub-)interval X_i, over some range of y-values. Overall, we will show that such a formulation is equivalent to a composition of functions, and use the Tree Decomposition Lemma to obtain our correctness and complexity guarantees.

4.6.1 Extending the Genuine-LIS problem

Before describing our construction for proving Theorem 69, we introduce a slightly stronger version of it, which will be easier to work with. In particular, we introduce 2 extensions of the requirements of the Genuine-LIS algorithm. First, we allow the input y-values to be *intervals* rather than integers, and second, we require the approximation

to improve if there are many "null" elements, i.e., if most blocks have much less than k elements.[4] In particular, to account for the latter, we note that the input size is $y(X) \cap Y$, which can be much less than $|X| \cdot k$.

Overall, the stronger version is the following:

Lemma 82 (Theorem 69, extended)**.** *Fix monotone functions $a_g : \mathbb{R}_+^2 \to [1,\infty)$, $a_s, c_g : \mathbb{R}_+^3 \to [1,\infty)$ and $c_s : \mathbb{R}_+^4 \to [1,\infty)$, satisfying, for all $r, \tau \in E_\beta$, $m \in \mathbb{N}$, $\lambda < 1$, $\lambda_1, \lambda_2 \in [\lambda, 1]$, and $k' \in [1, 1/\lambda]$ with $\lambda_1\lambda_2 = \Omega(\lambda k')$:*

- $a_s\left(r, \tau, \frac{\lambda}{m}\right) \geq \text{polylog}\left(\frac{k}{\lambda}\right) \cdot a_g\left(\tau, \frac{\lambda_2}{k'}\right) \cdot a_s\left(\frac{r}{\tau}, \tau, \frac{\lambda_1}{m}\right)$*; and*
- $c_s\left(r, \tau, \lambda, \frac{\lambda}{m}\right) \geq \text{polylog}(n) \cdot \frac{\tau}{\lambda} + c_g(\tau, \lambda_2, k') \cdot c_s\left(\frac{r}{\tau}, \tau, \Theta(\lambda_1/k'), \frac{\lambda_1}{m}\right)$.

Fix input $y \in \mathbb{N}^{n\times k}$, a block interval $X \subseteq [n]$, value range interval $Y \subseteq \mathbb{N}$, parameters $\lambda \in (0,1)$, $\beta \in \mathbb{N}$, $\tau \in E_\beta$. Suppose there exist the following algorithms:

$\mathcal{A}_{GL}$*: Given* Genuine-LIS *instance $g \in (\mathbb{N} \times \{0,1\})^{n_g \times k_g}$, with m_g non-null elements total, and β-ary Precision-Tree $T_{1/c_g(n_g,\lambda_g,m_g/n_g)}(g(*)_2)$ access, $\mathcal{A}_{GL}$ outputs $\left(a_g\left(n_g, \frac{\lambda_g n_g}{m_g}\right), \lambda_g n_g\right)$-approximation to* Genuine-LIS(g) *in time $\tilde{O}(n_g + m_g)$ w.h.p.*

$\mathcal{A}_{BL}$*: Fix $t > 1$. For any interval $X' \subset X$, value range $Y' \subset Y$, and any $\lambda_s < 1$, given a β-ary Precision-Tree $T_{1/c_s\left(|X'|,\tau,\lambda_s,\frac{\lambda_s|X'|}{|y_{-\lambda/2}(X',Y')|}\right)}(y(X'))$, $\mathcal{A}_{BL}$ outputs $(\alpha_s, \lambda_s|X'|)$-approximation to* Block-LIS(y, X', Y') *w.h.p., where $\alpha_s \triangleq a_s\left(|X'|, \tau, \frac{\lambda_s|X'|}{|y_{-\lambda/2}(X',Y')|}\right)$. The expected run-time is at most $t_{BL} = t \cdot c_s\left(|X'|, \tau, \lambda_s, \frac{\lambda_s|X'|}{|y_{-\lambda/2}(X',Y')|}\right) \cdot \frac{|y(X',Y')|}{|X'|}$.*

[4] We highlight that "null" is different from "not genuine"; while we need to test to determine that an element is "not genuine", "null" elements can be determined "for free".

Then, given a β-ary Precision-Tree $T_{1/c}(y)$, we can produce a $(\alpha, \lambda|X|)$-approximation for Block-LIS(y, X, Y) *w.h.p., as long as* $\alpha \geq a_s\left(|X|, \tau, \frac{\lambda|X|}{|y(X',Y')|}\right)$ *and* $c \geq c_s\left(|X|, \tau, \lambda, \frac{\lambda|X|}{|y(X',Y')|}\right)$.

The algorithm's expected run-time is at most $c \cdot t \cdot \log^{O(\log_\beta(\tau))}(\beta) \cdot \frac{|y(X,Y)|}{|X|}$.

To prove Theorem 69, we use the reduction from Lemma 78, along with the following reduction:

Lemma 83. *Suppose the algorithm* EstimateGenuineLIS $(a(\lambda/k), \lambda n)$*-approximates* Genuine-LIS *for some function a in time $t(n \cdot k)$ over integers, then there exists an algorithm $\mathcal{A}'$ that $(O(a(\lambda n/m)\log(1/\lambda)), \lambda n)$-approximates* Genuine-LIS *over intervals with m non-null elements. The run-time is $t(m) + O(n + m)$.*

The proof of Lemma 83 is deferred to Section 4.7.

Proof of Theorem 69 using Lemmas 78, 82 and 83. The proof follows by induction. We assume Theorem 69 holds for $X' \subset X$, setting $t = t_s(|X'|, \tau)$ (noting that for the base case, we have $a_s(1, \cdot, \cdot) = 1$ and $c_s(1, \cdot, \cdot, \cdot) = 1$, simply by outputting 1 if and only if the block is not empty). Also note that the extended algorithms assumed in Lemma 82 reduce to the standard version using Lemma 83, with $O(\log 1/\lambda_g)$ additional approximation (which can be absorbed into the polylog(k/λ)-approximation of parameter α). Finally, one can observe that the additive time of $\tilde{O}(n_g + m_g)$ does not change the asymptotic time complexity. □

The rest of this section is devoted to proving Lemma 82.

4.6.2 Algorithm

The algorithm for the Block-LIS problem, named EstimateBlockLIS, is described in Algorithm 18 and uses routines that we describe next.

We first provide some intuition underlying the algorithm. The algorithm is based on a reduction to a Genuine-LIS instance, where checking whether a item is genuine takes one Block-LIS call. We view the Genuine-LIS instance as the "global LIS" and the Block-LIS instances as "local LIS".

Consider the case when $k = 1$, in which case the problem becomes a standard LIS problem. In this case, one visualize the instance as a set of points on a standard two-dimensional grid, with X being an interval on the x-axis, and the element (i, y_i) being represented by the point with x-coordinate i and y-coordinate y_i. In this equivalent formulation, the objective is to determine the maximum length of a subsequence that is increasing with respect to both axes.

Assume that the longest increasing subsequence OPT is of length approximately $\lambda|X|$. Note that OPT can be distributed arbitrarily with respect to X. We split X into sub-intervals: for some length $r > 1$, use the standard, in-order partitioning of X into $\tau = |X|/r$ mutually disjoint and covering intervals $X_1, \cdots, X_\tau$ of length r each. There are two potential extreme scenarios for the distribution of elements from OPT:

1. All elements in OPT belong to approximately $\lambda|X|/r$ intervals X_i, and each such interval consists entirely of elements in OPT; the other intervals do not contribute at all.

2. OPT is uniformly distributed across all intervals, i.e., each interval X_i contributes approximately λr elements to OPT.

Intuitively speaking, the problem of *certifying a "close to optimal" monotone sequence* is more difficult when fewer elements participate in an optimal sequence. Therefore, in the first case above, one has to "work harder" on the global LIS to find the participating intervals X_i; however, little effort is required to verify the local LIS in each interval of interest, i.e., to get a lower bound on the LIS within each interval. In contrast, in the second case, it should be easier to determine the global LIS, but it is more difficult to approximate the local LIS (within each interval X_i).

Of course, the distribution of OPT among the intervals X_i can be arbitrary, between these extreme scenarios. However, one can show that there must exist some $\rho \in [1, 1/\lambda]$, such that there are approximately $1/\rho$ fraction of intervals X_i, each having approximately $\lambda\rho r$ "local" contribution to OPT.

4.6.2.1 Algorithm Overview

Now we describe the algorithm for the general Block-LIS problem. Before describing the main algorithm, we describe its main subroutines and mechanism, setting up some concepts and notation along the way. The notation is summarized in Table 4. The algorithm assumes no element repeats itself in the sequence. To remove such assumption, one can reconfigure the element values setting $y_{i,j} \leftarrow n \cdot y_{i,j} - i$.

Sampling. The algorithm starts by sampling uniformly random blocks (w_j, y_{w_j}) and then accumulating all y_{w_j}-samples in Y within the sampled blocks (we note that sampling is accomplished via the SODS over the input Precision-Tree). The main idea is to generate a set $\mathcal{S}_i$ of sampled values, such that the "distance in $\mathcal{S}_i$" between any 2 values in $\mathcal{S}_i$ is approximately proportional to the number of integers in the range over the entire $y(X_i)$. To obtain such a guarantee, we simply use members of $\mathcal{S}_i$ whose

rank is a multiple of $O(\log n)$. We also want to make sure there are not too many total samples in all blocks, i.e., that $\sum_i |\mathcal{S}_i|$ is close to its expectation. To control the tail bounds of such a quantity, we also discard the $\Theta(\log n)$ blocks with the largest number of samples in Y. The sampling algorithm is presented in Algorithm 16, and the guarantees are formally stated in Claim 88, as part of the analysis.

Algorithm 16 SAMPLEANDPARTITION

Input: A Precision-Tree $T = T_{1/c}(y)$, where $y \in \mathbb{N}^{X \times [k]}$ consists of $|X|$ blocks with at most k integers each, value range $Y \in \mathcal{I}$, branching parameter τ, and sample size s.

Output: Partition of X with sampled y-values of each part.

$\zeta \leftarrow O(\log n)$.

$X_1, \ldots, X_\tau \leftarrow$ a partition of X into τ consecutive intervals, each consisting of $r \triangleq |X|/\tau$ blocks.

Using Corollary 74 and T, sub-sample blocks $S \subseteq X$ i.i.d. with probability $\zeta \frac{s}{|X|}$, and let $W \leftarrow \{(i,j) \in S \times [k] \mid y_{i,j} \in Y\}$.

Let $B \subseteq S$ be the ζ largest blocks in S, i.e., for which $|B \times [k] \cap W|$ is maximized.

for $i \in [\tau]$ **do**

$\{p_1^i, p_2^i, \ldots\} \leftarrow y\left(W \cap ((X_i \setminus B) \times [k])\right)$ where $p_m^i \le p_{m+1}^i$ for all m.

$\mathcal{S}_i \leftarrow \{p_m^i\}_{m \in [\lfloor p_*^i \rfloor] \cap \zeta\mathbb{N}}$.

end for

return $(X_*, \mathcal{S}_*)$.

Figure 9: Description of the sampling and partitioning subroutine.

Generation of Candidate Intervals. Now we describe our construction of the candidate intervals $\mathcal{Y}_i$ using the sampled y-values $\mathcal{S}_i$, for some fixed i. The high-

level goal, as in [185], is to cover an optimal LIS solution using a small number of monotone interval sequences denoted as *pseudo-solutions*, so that the largest LIS within all pseudo-solutions would be a good approximation to the optimum. However, we need to choose such candidates more carefully, since, in addition to "capturing the local LIS", we would like to ensure the following efficiency guarantees, for our delicate bounds:

1. There are not too many candidate intervals, i.e., $|\mathcal{Y}_i| \lessapprox \frac{|y(X_i,Y)|}{\lambda|X|}$; and more importantly,

2. The total number of integers in range over all candidate intervals do not cause significant overhead. In particular: $\sum_{Y' \in \mathcal{Y}_i} |y(X_i, Y')| \lessapprox |y(X_i, Y)|$.

In particular, we argue that given a value set $\mathcal{S}$ (corresponding to the sampled values in each X_i) and a sub-additive set function ψ, we can generate a near-linear set of candidate intervals which approximates, up to a logarithmic factor, all candidate intervals. Here we note that the set function $\psi_{X_i,y}(Y') \triangleq \textsf{Block-LIS}(X_i, Y', y)$ is sub-additive in Y'. While we did not manage to create a single set of intervals $\mathcal{Y}_i$ as above, we are able to create a *small family of sets* $\{\mathcal{Y}_{i,\Delta}\}_{\Delta \in E_2(|\mathcal{S}_i|)}$ with similar guarantees. We show the following:

Lemma 84. *Fix a finite set of values $\mathcal{S} \subseteq \mathbb{N}$ of size m. There exist sets of intervals $\{\mathfrak{C}^{\mathcal{S}}_{\Delta}\}_{\Delta \in E_2(m)} \subseteq \mathcal{I}$ such that:*

1. *For all Δ and all $I \in \mathfrak{C}^{\mathcal{S}}_{\Delta}$: $|I \cap \mathcal{S}| = \Delta$.*

2. *For all Δ: $|\mathfrak{C}^{\mathcal{S}}_{\Delta}| = |\mathcal{S}|/\Delta$.*

3. *Fix a sub-additive set function* $\psi : \mathcal{I} \to \mathbb{R}$. *For any* $I \in \{[a,b] \in \mathcal{I} \mid a,b \in \mathcal{S}\}$, *there exists some* $J \in \cup_\Delta \mathfrak{C}_\Delta^{\mathcal{S}}$ *with* $J \subset I$ *such that* $\psi(I) = O(\log|\mathcal{S}|) \cdot \psi(J)$.

We prove Lemma 84 in Section 4.6.4. For now, for each set $\mathcal{S}_i = \{p_1^i, p_2^i, \ldots\}$, we construct, on average, $O(\log(k/\lambda))$ *candidate interval sets* $\{\mathcal{Y}_{i,\Delta}\}_\Delta$ where $\Delta \in E_2(|\mathcal{S}_i|)$ and each $\mathcal{Y}_{i,\Delta}$ contains precisely Δ (consecutive) elements of $\mathcal{S}_i$. In particular, $\mathcal{Y}_{i,\Delta}$ are the dyadic intervals of $\mathcal{S}_i$, i.e., $\mathcal{Y}_{i,\Delta}$ is the set of all the intervals $[p^i_{k\Delta+1}, p^i_{(k+1)\Delta}]$ for all possible integer $k \geq 0$.

Discretization. The algorithm performs exponential discretization over the following parameters:

ρ: inverse of the fraction of intervals participating in OPT. This parameter characterizes the relation between the "global" and "local" LIS. In particular, for the two extreme scenarios outlined above, the parameter ρ would be $1/\lambda$ and 1 respectively.

Δ: a quantity proportional to the "candidate interval size" of the local-LIS (namely, to $|y_{-\lambda/2}(y, X_i, Y')|$) and inversely proportional to the "average block size" of the global-LIS problem (namely, to m_g/n_g[5]).

Decomposition and Recursion. The next step is to determine, for each choice of parameters (ρ, Δ), the largest set of monotone candidate intervals $\{(i, Y') : Y' \in \mathcal{Y}_{i,\Delta}\}_i$ such that each candidate interval has local LIS at least $\kappa_\rho \triangleq r/\rho$. For this purpose, the algorithm defines the pair (i, Y') as "genuine" iff the local LIS is long enough

[5]Recall from Lemma 82 that m_g is the total number of non-null elements of the instantiated Genuine-LIS problem and n_g is the number of blocks.

(determined via recursive call to Block-LIS). Then the problem becomes to solve a "global" Genuine-LIS instance over all genuine (X_i, Y') pairs. In other words, we formulate the problem as a composition of a Genuine-LIS problem over many smaller Block-LIS problems, where the Precision-Tree decomposition algorithm (Lemma 75) is used to access the sequence question.

Optimization. Finally, we output the solution with the maximum value among all combinations of the parameter ρ and Δ.

Algorithm 17 DECOMPOSE

Input: A Precision-Tree $T = T_{1/c}(y)$, where $y \in \mathbb{N}^{X \times [k]}$ consists of $|X|$ blocks with at most k integers each, x-partition $\{X_i\}_i$ of X, y-partition sets $\{\mathcal{Y}_i\}_i$ (for a fixed Δ) of some interval $Y \in \mathcal{I}$.

Output: An integer $\widehat{L} \in [0, \text{Block-LIS}(y, X, Y)]$.

Let $\mathcal{A}_{\text{GL}}$, $\mathcal{A}_{\text{BL}}$ be the aforementioned algorithms from Lemma 82.

Define $\kappa_\rho \triangleq \frac{r}{\rho}$.

For each i, define $\Lambda_\rho(i, *) \triangleq \left[\left(Y', \mathbf{1}\left[\mathcal{A}_{\text{BL}}\left(y(X_i), Y', \frac{1}{\rho}, \tau\right) > \kappa_\rho\right]\right)\right]_{Y' \in \mathcal{Y}_i}$.

for $\rho \in E_2(1/\lambda)$ **do**

Compute $G_\rho \leftarrow \mathcal{A}_{\text{GL}}(\Lambda_\rho, \lambda\rho) \cdot 0.5\kappa_\rho$ using Lemma 75 and T.

end for

return $\widehat{L} \leftarrow \max_\rho G_\rho$.

Figure 10: Description of the decomposition subroutine. Note that lines $2 - 4$ are merely definitions for $\rho \in E_2(1/\lambda)$, and not computational steps.

Symbol, Definition	**Description**
λ	additive error parameter (as fraction); given as input
τ	branching factor; given as input
X_i	sub-interval of X
$r := \lvert X\rvert/\tau$	*radius*, i.e., number of blocks in each interval X_i
ρ	inverse of fraction of intervals participating in *global* OPT
$\kappa_\rho := r/\rho$	the local-LIS threshold that qualifies a candidate interval as "genuine"
Y	permissible range of y-values
$\mathcal{S}_i$	y-values in Y found in sampled blocks in X_i
$\mathcal{Y}_{i,\Delta}$	candidate intervals for X_i using samples of $\mathcal{S}_i$ of "distance" $\approx \Delta$
$\ell_{i,Y'} :=$ Block-LIS(y, X_i, Y')	the length of the longest sub-sequence of $y(X_i)$ using values in Y'
$T := T_{1/c}(y)$	β-ary Precision-Tree with starting complexity/precision c
$a_g(\cdot,\cdot), c_g(\cdot,\cdot)$	approximation and complexity functions for Genuine-LIS
$a_s(\cdot,\cdot,\cdot), c_s(\cdot,\cdot,\cdot)$	approximation and complexity functions for Block-LIS
$y_{-\delta,Y'}$	block sequence y, where the heaviest δ fraction of blocks (considering only values in Y') are emptied
$y_{-\delta}(X', Y') := y(X')_{-\delta,Y'} \cap Y'$	multiset of values in $y(X')_{-\delta,Y'}$ restricted to Y'

Table 4: Block-LIS Algorithm: Notation Table

Algorithm 18 EstimateBlockLIS

Input: A Precision-Tree $T = T_{1/c}(y)$, where $y \in \mathbb{N}^{X \times [k]}$ consists of $|X|$ blocks with at most k integers each, value range $Y \subseteq \mathbb{N}$, error parameter $\lambda \in (1/n, 1)$, branching parameter τ.

Output: An integer $\widehat{L} \in [0, \mathsf{Block\text{-}LIS}(y, X, Y)]$.

if $|X| < c$ **then**

  return $\widehat{L} \leftarrow \mathsf{Block\text{-}LIS}(y, X, Y)$ through exact computation.

end if

$(X_*, \mathcal{S}_*) \leftarrow$ SampleAndPartition$(T, \tau, \frac{\tau}{\lambda})$.

$\mathcal{Y}_{i,*} \leftarrow$ sets of intervals by invoking Lemma 84 using each $\mathcal{S}_i \in \mathcal{S}_*$.

return $\widehat{L} \leftarrow \max_\Delta\{$Decompose$(T, X_*, \mathcal{Y}_{*,\Delta})\}$.

Figure 11: Description of the algorithm EstimateBlockLIS.

4.6.3 Main analysis

Before analyzing the estimator $\widehat{L}$, we show several important properties. First, we claim that partitioning X into intervals of smaller size and matching each one with a value range Y monotonically cannot over-estimate the overall LIS:

Lemma 85 (Upper Bound)**.** *Fix an interval $Y \subset \mathbb{N}$. Let $X_1, \ldots, X_\tau$ be a partition of X into consecutive, disjoint, in-order block intervals (i.e., $X_1 < X_2 < \ldots < X_\tau$) and let $P \subset [\tau] \times \mathcal{I}$ be an arbitrary* monotone *set. Then, we have $\sum_{(i,Y_i)\in P} \mathsf{Block\text{-}LIS}(y, X_i, Y_i \cap Y) \le \mathsf{Block\text{-}LIS}(y, X, Y)$.*

Proof. For each $(i, Y_i) \in P$, let $\mathsf{OPT}_i \subseteq (X_i \times [k]) \cap y^{-1}(Y_i \cap Y)$ be a set of coordinates of an optimal increasing subsequence for $\mathsf{Block\text{-}LIS}(y, X_i, Y_i \cap Y)$ Then $\sum_{(i,Y_i)\in P} |\mathsf{OPT}_i| = \sum_{(i,Y_i)\in P} \mathsf{Block\text{-}LIS}(y, X_i, Y_i \cap Y)$. On the other hand, since P

is a monotone set, we have $y(p) < y(q)$ for any $p \in \mathsf{OPT}_i, q \in \mathsf{OPT}_j$ with $i < j$. We conclude that the sequence $\cup_i y(\mathsf{OPT}_i) \subseteq Y$ forms an increasing subsequence of $y(X)$ of length $\sum_{(i,Y_i)\in P} \textsf{Block-LIS}(y, X_i, Y_i \cap Y)$ and the claim follows. □

Next, we argue three essential invariants when constructing candidate intervals. These guarantees will be used later to lower bound our estimator:

Lemma 86 (Candidate intervals guarantees)**.** *Fix input X, Y, y. With high probability, all the following invariants hold for the* EstimateBlockLIS *algorithm:*

1. *For all i, Δ and all $Y' \in \mathcal{Y}_{i,\Delta}$,*

$$|y_{-\lambda/2}(X_i, Y')| = O(\lambda \Delta r).$$

2. *For all Δ,*

$$\mathbb{E}_{i\in[\tau]}\left[|\mathcal{Y}_{i,\Delta}|\right] = O\left(\frac{|y(X,Y)|}{\lambda\Delta|X|}\right),$$

3. *There exist candidate intervals $Y' \in \oplus_{i\in[\tau]} \cup_{\Delta\in E_2(\mathrm{poly}(k/\lambda))} \mathcal{Y}_{i,\Delta}$ such that $\{(i, Y'_i)\}_i$ is a monotone set; and*

$$\sum_i \textit{Block-LIS}(y, X_i, Y'_i) = \Omega\left(\frac{\textit{Block-LIS}(y, X, Y) - \lambda|X|}{\log(k/\lambda)}\right).$$

We analyze the correctness and complexity of our algorithm, proving Lemma 82 assuming Lemma 86. We prove Lemma 86 in Section 4.6.5.

Proof of Lemma 82 using Lemma 86. For all i, Δ and all $Y' \in \mathcal{Y}_{i,\Delta}$, let $\ell_{i,Y'} \triangleq \textsf{Block-LIS}(y, X_i, Y')$. By Lemma 86 (3), we have that for some Δ^*, there exists a monotone set of pairs $P^* \in \oplus_i(\{i\} \times \mathcal{Y}_{i,\Delta^*})$ such that:

$$\sum_{(i,Y')\in P^*} \ell_{i,Y'} \geq \Omega\left(\frac{\textsf{Block-LIS}(y,X,Y)-\lambda|X|}{\log(k/\lambda)\cdot\log(k/\lambda)}\right) > \frac{\Omega(\textsf{Block-LIS}(y,X,Y))-\lambda|X|}{\log^2(k/\lambda)}.$$

Therefore, we have that, for any $\alpha_s > 1$, there is some $\rho^* \in E_2(1/\lambda)$ for which, using $\kappa_{\rho^*} = r/\rho^*$:

$$\sum_{(i,Y')\in P^*} \mathbf{1}\left[\ell_{i,Y'} \geq 2\alpha_s \cdot \kappa_{\rho^*}\right] \geq \frac{1}{2\alpha_s \kappa_{\rho^*}} \cdot \frac{\Omega(\textsf{Block-LIS}(y,X,Y)) - \lambda|X|}{\log^3(k/\lambda)} \tag{4.3}$$

Let us first argue correctness. Define $\kappa^* \triangleq \kappa_{\rho^*} = \frac{r}{\rho^*}$ and $\mathcal{Y}_i^* \triangleq \mathcal{Y}_{i,\Delta^*}$. Recall the following guarantees of $\mathcal{A}_{\text{BL}}$ and $\mathcal{A}_{\text{GL}}$ which hold for each pair $(i, Y') \in P^*$:

1. Let $\widehat{\ell_{i,Y'}}$ be the output of $\mathcal{A}_{\text{BL}}(y(X_i), Y', \frac{1}{\rho^*}, \tau)$. Then $\widehat{\ell_{i,Y'}}$ is a $\left(\alpha_s, \frac{r}{\rho^*}\right)$ - approximation of $\ell_{i,Y'}$, where $\alpha_s \triangleq a_s\left(r, \tau, \frac{r/\rho^*}{|y_{-\lambda/2}(X_i,Y')|}\right)$.

2. Let $\Lambda^* \triangleq \Lambda_{\Delta^*,\rho^*}$. The output of $\mathcal{A}_{\text{GL}}(\Lambda^*, \lambda\rho^*)$ is a $(\alpha_g, \lambda\rho^*\tau)$-approximation of $\sum_{(i,Y')\in P^*} \mathbf{1}\left[\widehat{\ell_{i,Y'}^*} \geq \kappa^*\right]$, where $\alpha_g \triangleq a_g\left(\tau, \frac{\lambda\rho^* n_g}{m_g}\right)$, where $n_g = \tau$ and $m_g = \sum_i [|\mathcal{Y}_i^*|]$, are, respectively, the number of blocks and non-null elements in the Genuine-LIS instance.

We use the approximation guarantees from above to obtain a lower bound on $L^* \triangleq L_{\Delta^*,\rho^*}$. Assuming $c \geq c_s(\cdot) \cdot c_g(\cdot)$, we invoke Lemma 75 to get that the quantity L^* is a $(\alpha_g, \lambda\rho^*\tau)$-approximation for a Genuine-LIS instance, where each block i is defined by all possible candidate intervals $\mathcal{Y}_i^*$, and each candidate interval $Y' \in \mathcal{Y}_i^*$ is genuine iff a $\left(\alpha_s, \frac{r}{\rho^*}\right)$-local approximation $\widehat{\ell_{i,Y'}}$ passes the κ_{ρ^*} threshold.

Note that the approximations are:

$$\alpha_s = a_s\left(r, \tau, \frac{r}{\rho^*|y_{-\lambda/2}(X_i,Y')|}\right)$$
$$\alpha_g = a_g\left(\tau, \frac{\lambda\rho^*}{\mathbb{E}_i\left[|\mathcal{Y}_i^*|\right]}\right).$$

We let $\lambda_s = \frac{|y(X,Y)|}{|X|} \cdot \frac{r}{\rho^*|y_{-\lambda/2}(X_i,Y')|}$, and $\lambda_g = \frac{m_g}{n_g} \cdot \frac{\lambda\rho^*}{\mathbb{E}_i\left[|\mathcal{Y}_i^*|\right]} = \lambda\rho^*$, in which case:

$$\alpha_s \cdot \alpha_g = a_s\left(r, \tau, \frac{\lambda_s|X|}{|y(X,Y)|}\right) \cdot a_g\left(\tau, \frac{\lambda_g n_g}{m_g}\right). \tag{4.4}$$

We need to bound $\lambda_s \lambda_g$. For this, we use Lemma 86 (1) and (2), to obtain for each $Y' \in \mathcal{Y}_i^*$, for all i:

$$|y_{-\lambda/2}(X_i) \cap Y'| = O(\lambda \Delta r); \tag{4.5}$$

$$\frac{m_g}{n_g} = \mathbb{E}_i[|\mathcal{Y}_i^*|] = O\left(\frac{|y(X,Y)|}{\lambda \Delta |X|}\right). \tag{4.6}$$

We obtain that

$$\lambda_s \lambda_g \cdot n_g/m_g = \Omega\left(\frac{|y(X,Y)|}{|X|} \cdot \frac{r}{\rho^* \lambda \Delta r} \cdot \lambda \rho^* \cdot \frac{\lambda \Delta |X|}{|y(X,Y)|}\right) = \Omega(\lambda),$$

as required in the theorem statement.

We can finally derive the following overall bound on $\widehat{L}$, and in particular, show it is a $(\alpha, \lambda|X|)$-approximation, for some $\alpha = O(\alpha_s \cdot \alpha_g \cdot \text{polylog}(k/\lambda))$. For the lower bound, recall $\widehat{L} = \max_{\rho,\Delta} L_{\rho,\Delta} \geq L^*$, and therefore we have that:

$$\begin{aligned}
\widehat{L} &\geq L^* \\
&\geq \left(\frac{1}{\alpha_g} \sum_{(i,Y') \in P^*} \mathbf{1}\left[\widehat{\ell_{i,Y'}} \geq \kappa^*\right] - \lambda \rho^* \tau\right) \cdot 0.5\kappa^* \\
&\geq \sum_{(i,Y') \in P^*} \frac{1}{\alpha_g} \mathbf{1}\left[\ell_{i,Y'} \geq \alpha_s \cdot \left(\kappa^* + \frac{r}{\rho^*}\right)\right] \cdot 0.5\kappa^* - \lambda \rho^* \tau \cdot 0.5\kappa^* \\
&\geq \frac{0.5\kappa^*}{\alpha_g} \cdot \sum_{(i,Y') \in P^*} \mathbf{1}\left[\ell_{i,Y'} \geq \alpha_s \cdot \left(\kappa^* + \frac{r}{\rho^*}\right)\right] - 0.5\lambda|X| \\
&\geq \frac{0.5\kappa^*}{\alpha_g} \cdot \sum_{(i,Y') \in P^*} \left(\mathbf{1}\left[\ell_{i,Y'} \geq \alpha_s \cdot (\kappa^* + \kappa^*)\right]\right) - 0.5\lambda|X| \\
&\geq \frac{0.5\kappa^*}{\alpha_g} \cdot \frac{1}{2\alpha_s \kappa_{\rho^*}} \cdot \frac{\Omega(\textsf{Block-LIS}(y,X,Y)) - \lambda|X|}{\log^3(k/\lambda)} - 0.5\lambda|X| \\
&> \Omega\left(\frac{\textsf{Block-LIS}(y,X,Y)}{\alpha_s \alpha_g \log^3(k/\lambda)}\right) - \lambda|X|.
\end{aligned}$$

For the upper bound, we use Lemma 85 to claim that for any set of parameters, and any choice of monotone pairs, $L_{\rho,\Delta}$ is upper bounded by $\textsf{Block-LIS}(y, X, Y)$, and hence

any approximation of such quantity is guaranteed to be bounded by Block-LIS(y, X, Y) as well.

Proving the assumed bound for c. For correctness, it is left to show the lower bound for c suffices to obtain $c \geq c_{sampling} + c_g(\cdot) \cdot c_s(\cdot)$ for every set of potential parameters from above, and where $c_{sampling}$ is the complexity for sampling the set S_*. Note that $c_{sampling} = \zeta s = \zeta\tau/\lambda$. As regards c_s, c_g, in order to obtain the desired approximations α_g, α_s, we need complexities $c_g = c_g(\tau, \lambda_g, m_g/n_g)$ and $c_s = c_s(r, \tau, 1/\rho^*)$ respectively. Now, noting $\frac{m_g}{n_g} \overset{Eq.\ (4.6)}{=} O\left(\frac{|y(X,Y)|}{\lambda\Delta|X|}\right)$, we have

$$\frac{1}{\rho^*} = \frac{\lambda\cdot\lambda_s\cdot\Delta\cdot|X|}{|y(X,Y)|} = \Omega\left(\frac{\lambda_s n_g}{m_g}\right),$$

as needed. This concludes the correctness proof.

Runtime Analysis. The algorithm consists of the following procedures:

First, **sub-sampling** of blocks, with expected runtime $\tilde{O}(\frac{\tau}{\lambda}) \cdot \frac{|y(X,Y)|}{|X|}$ by Corollary 74.

Second, **computation of candidate intervals**, takes time $\tilde{O}(|S|) = \tilde{O}(\frac{\tau}{\lambda}) \cdot \frac{|y(X,Y)|}{|X|}$.

Last, and most important, are the recursive calls invoked by the **decomposition procedure**. Here, for each set of parameters Δ, t, the expected runtime by Lemma 75 is

$$\tilde{O}\left(\tau + t_{GL} + \frac{1}{\delta\tau} \cdot \|t_f\|_1 \cdot \log^{O(\log_\beta(\tau))}(\beta)\right),$$

where $\delta = \frac{1}{c_g(\cdot)}$. The Lemma assumption provides the bound

$$t_{GL} = \tilde{O}(m_g + n_g) = \tilde{O}(\tau + \sum_i |\mathcal{Y}_{i,\Delta}|) = \tilde{O}(\tau + |S|) = \tilde{O}(\tfrac{\tau}{\lambda}) \cdot \tfrac{|y(X,Y)|}{|X|},$$

and hence the main quantity to analyze is $\frac{1}{\delta\tau} \cdot \|t_f\|_1 \cdot \log^{O(\log_\beta(\tau))}(\beta)$.

Notice that computing each f_i requires $|\mathcal{Y}_{i,\Delta}|$ invocations of $\mathcal{A}_{\mathrm{BL}}$, where for each $Y' \in \mathcal{Y}_{i,\Delta}$, we pay (expected) time $t \cdot c_s(\cdot) \cdot \frac{|y(X_i,Y')|}{|X_i|}$. Furthermore, note that candidate intervals in $\mathcal{Y}_{i,\Delta}$ are disjoint and contained in Y, and hence $\sum_{Y' \in \mathcal{Y}_{i,\Delta}} |y(X_i, Y')| \leq |y(X_i, Y)|$. Therefore:

$$\mathbb{E}_T\left[\|t_f\|_1\right] \leq \sum_i \sum_{Y' \in \mathcal{Y}_{i,\Delta}} t \cdot c_s \cdot \frac{|y(X_i,Y')|}{|X_i|} \leq \sum_i t \cdot c_s \cdot \frac{|y(X_i,Y)|}{|X_i|} = t \cdot c_s \cdot \frac{|y(X,Y)|\tau}{|X|}.$$

Now, since $c \geq c_s \cdot c_g$, then,

$$\mathbb{E}_T\left[\frac{1}{\delta\tau} \cdot \|t_f\|_1\right] = \frac{c_g}{\tau} \cdot \mathbb{E}[\|t_f\|_1] \leq t \cdot c_s \cdot c_g \cdot \frac{|y(X,Y)|}{|X|} \leq t \cdot c \cdot \frac{|y(X,Y)|}{|X|}.$$

Counting the time taken by all procedures above, the overall expected runtime is:

$$\tilde{O}\left(\frac{\tau}{\lambda} + t \cdot c \cdot \log^{O(\log_\beta(\tau))}(\beta)\right) \cdot \frac{|y(X,Y)|}{|X|} = t \cdot c \cdot \log^{O(\log_\beta(\tau))}(\beta) \cdot \frac{|y(X,Y)|}{|X|},$$

as needed.

□

4.6.4 Dyadic construction for candidate intervals: proof of Lemma 84

We now describe the algorithm for generating candidate intervals, proving Lemma 84. First, we show how to generate a small "covering" set of candidate intervals over some fixed set $[m]$.

Claim 87. *For any $m \in \mathbb{N}$, there exists a clustering of $[m]$ into intervals $\{\mathcal{J}_{\Delta,1}, \mathcal{J}_{\Delta,2}, \ldots, \mathcal{J}_{\Delta,m/\Delta}\}_{\Delta \in E_2(m)}$ where each $\mathcal{J}_{\Delta,j} \subseteq \mathcal{I} \cap 2^{[m]}$ and further:*

1. $|\mathcal{J}_{\Delta,j}| = \Delta$ *for all* Δ, j.

2. *For any interval* $I \in [m] \cap \mathcal{I}$, *there exists a set of intervals* $\mathfrak{C}_I \subseteq \{\mathcal{J}_{\Delta,j}\}_{\Delta,j}$ *of size* $|\mathfrak{C}_I| = O(\log m)$, *such that* I *is precisely covered by* $\mathfrak{C}_I$; *i.e.,* $I = \cup_{\mathcal{J}\in\mathfrak{C}_I}\mathcal{J}$.

Proof. Consider a binary tree T_m with leaves indexed by integers in $[m]$ and internal nodes indexed by the union of its children. The set of clusters corresponding to the indexing of all nodes in each level of the tree provides the above guarantees. For (1), we have a node at each height $h \in [\log_2 m]$ of length $\Delta = 2^h$ hence that bound is immediate.

For (2), we reconstruct I as follows: let $a \triangleq \min\{I\}$ and $b \triangleq \max\{I\}$. We start with v_a (the leaf with index a) and add it to $\mathfrak{C}_I$. Now we go up the tree level by level, each time adding the *sibling* of the current node if and only if it is contained in I. Similarly, we do the same process with v_b. We note that this process generates $\mathfrak{C}_I \subseteq \{\mathcal{J}_{\Delta,j}\}_{\Delta,j}$ of size $O(\log m)$. It is left to show that $I = \cup_{\mathcal{J}\in\mathfrak{C}_I}\mathcal{J}$.

For this task, consider any $c \in [m]$. One one hand, if $c \notin I$, no ancestor of c will be added to $\mathfrak{C}_I$ by construction. On the other hand, for $c \in I$, let v be the "lowest" node such that either a, c or b, c are common descendants of v. Assume w.l.o.g. that a, c are the common descendants (otherwise consider the symmetric argument). Now, if $c = a$, then we add $v_a = v_c$ to $\mathfrak{C}_I$. Otherwise, $c \in (a, b)$, then $c \in \text{right}(v) \subseteq I$, and hence right$(v)$ will be added to $\mathfrak{C}_I$. We conclude that $c \in I$ if and only if $c \in \cup_{\mathcal{J}\in\mathfrak{C}_I}\mathcal{J}$ as needed.

□

Second, We prove Lemma 84, by simple ordering of the set $\mathcal{S}$, and reducing any interval over $[m]$ to an interval over $\mathcal{S}$.

Proof of Lemma 84. Index all values by rank such that $\mathcal{S}_1 \leq \mathcal{S}_2 \leq \ldots \leq \mathcal{S}_m$. We now construct a clustering of $[m]$ as per Claim 87, and output $\mathfrak{C}_\Delta^{\mathcal{S}} \triangleq \{[\mathcal{S}_a, \mathcal{S}_b] \mid [a,b] \in \{\mathcal{J}_{\Delta,i}\}_i\}$. Then all the above guarantees are immediate (the last guarantee is obtained by sub-additivity). □

4.6.5 Analysis of Block-LIS invariants: proof of Lemma 86

To prove Lemma 86, we state the following claim:

Claim 88. *The algorithm* SampleAndPartition *generates* $\{(X_i, \mathcal{S}_i)\}_i$ *pairs which, with high probability, satisfy the following for all* $i \in [\tau]$ *and* $Y' \in \mathcal{I}$*:*

1. $\sum_{w \in X_i} \mathbf{1}[y(w) \cap Y' \neq \emptyset] \leq \lambda r \cdot (|Y' \cap \mathcal{S}_i| + 2)$
2. $|y_{-\lambda/2}(X_i, Y')| \leq 4\lambda r \cdot (|Y' \cap \mathcal{S}_i| + 2)$.

We now prove Lemma 86 using Claim 88.

Proof of Lemma 86. Fix i, Δ. By Lemma 84 (1), we have for all $Y' \in \mathcal{Y}_{i,\Delta}$ that $|Y' \cap \mathcal{S}_i| = \Delta + 1$, and we use Claim 88 (2), to obtain the first invariant.

For the second invariant, we bound $\mathbb{E}_{i \in [\tau]}[|\mathcal{Y}_{i,\Delta}|]$ by arguing about the random sets $W, B, \mathcal{S}_*$ computed in the SampleAndPartition algorithm. Let $\mathcal{S} = \cup_i \mathcal{S}_i$, we would like to show high probability bound on $|\mathcal{S}|$. Note that $|\mathcal{S}| \cdot \zeta = \Theta(|W \setminus (B \times [k])|)$. Now, each block is subsampled independently with probability $\left(\frac{\zeta\tau}{\lambda|X|}\right)$. Then, $\mathbb{E}[|W|] = \frac{\zeta\tau|y(X,Y)|}{\lambda|X|}$. For our task, we invoke Fact 81, which implies $|S| = O\left(\frac{\tau|y(X,Y)|}{\lambda|X|}\right)$ w.h.p. We also invoke Lemma 84 (2) to obtain $|\mathcal{Y}_{i,\Delta}| = \frac{|S_i|}{\Delta}$, which implies

$$\mathbb{E}_i[|\mathcal{Y}_{i,\Delta}|] = \mathbb{E}_i\left[\frac{|S_i|}{\Delta}\right] = \frac{|S|}{\Delta\tau} = O\left(\frac{|y(X,Y)|}{\lambda\Delta|X|}\right),$$

as needed.

For the third invariant, let OPT be (the indices of) an optimal increasing subsequence of $\mathsf{Block\text{-}LIS}(y, X, Y)$. Let $M_i \triangleq \max\{y(X_i \cap \mathsf{OPT})\}$ be the maximal value of $y(\mathsf{OPT})$ in X_i (by convention, $M_i = \perp$ if $\mathsf{OPT} \cap X_i = \emptyset$). Let $U_i \triangleq (M_j, M_i]$ for the largest $j < i$ such that M_j is defined (similarly, let $U_i = \perp$ if $\mathsf{OPT} \cap X_i = \emptyset$). Last, define

$$R_i \triangleq \operatorname*{argmax}_{[j,k] \subseteq U_i : j,k \in \mathcal{S}_i} (k - j),$$

i.e., the largest interval in U_i using the values of $\mathcal{S}_i$ only. We have the following bound on $\mathsf{Block\text{-}LIS}(y, X, Y)$:

$$\begin{aligned}\mathsf{Block\text{-}LIS}(y, X, Y) &= \sum_i \mathsf{Block\text{-}LIS}(y, X_i, U_i) \\ &\leq \sum_i \left(\mathsf{Block\text{-}LIS}(y, X_i, R_i) + \mathsf{Block\text{-}LIS}(y, X_i, U_i \setminus R_i)\right).\end{aligned}$$

Now, notice that by definition of R_i, $U_i \setminus R_i \cap \mathcal{S}_i = \emptyset$. Notice also that $U_i \setminus R_i$ consists of at most 2 intervals. We invoke Claim 88 (1) on each one of those intervals to obtain that $\sum_{w \in X_i} \mathbf{1}[y(w) \cap U_i \setminus R_i \neq \emptyset] \leq 4\lambda r$, implying that $\mathsf{Block\text{-}LIS}(y, X_i, U_i \setminus R_i) \leq 4\lambda r$ as well. We therefore have:

$$\sum_i \mathsf{Block\text{-}LIS}(y, X_i, R_i) \geq \mathsf{Block\text{-}LIS}(y, X, Y) - 4\lambda r \cdot \tau = \mathsf{Block\text{-}LIS}(y, X, Y) - 4\lambda |X|$$

It remains to bound $\sum_i \mathsf{Block\text{-}LIS}(y, X_i, R_i)$. Here, we use Lemma 84 (3), with $\psi(J) = \mathsf{Block\text{-}LIS}(y, X_i, J)$ and $I = R_i$, to obtain $\mathsf{Block\text{-}LIS}(y, X_i, R_i) \leq O(\log |\mathcal{S}_i|) \cdot \mathsf{Block\text{-}LIS}(y, X_i, Y_i)$, for some $R_i \supseteq Y_i \in \mathcal{Y}_{i,*}$. Last, to avoid "too large Δ", we simply do not count intervals with many sampled pivots. In particular, note that since $|\mathcal{S}| \leq 2k\tau/\lambda$ whp (by the immediate Chernoff bound), we have that at most $\lambda\tau$

blocks satisfy $|\mathcal{S}_i| \geq 2k/\lambda^2$. Overall:

$$\begin{aligned}\sum_{i:|S_i|\leq k/\lambda^2} \textsf{Block-LIS}(y, X_i, Y_i) &\geq \sum_{i:|\mathcal{S}_i|\leq 2k/\lambda^2} \frac{\textsf{Block-LIS}(y,X_i,R_i)}{O(\log|\mathcal{S}_i|)} \\ &\geq \frac{1}{O(\log(k/\lambda))}\left(\sum_{i\in[\tau]} \textsf{Block-LIS}(y, X_i, R_i) - \lambda|X|\right) \\ &= \Omega\left(\frac{\textsf{Block-LIS}(y,X,Y)-\lambda|X|}{\log(k/\lambda)}\right).\end{aligned}$$

We also note that since $\{(i, R_i)\}_i$ are monotone, so are Y_i. This concludes the proof. □

It remains to prove Claim 88.

Proof of Claim 88. Fix $Y' \in \mathcal{I}$. Assume w.l.o.g $Y' \in \{(j,k) \mid \{j,k\} \in y(X_i) \cup \{-\infty, \infty\}\}$ (otherwise, consider the smallest interval in this set, which contains Y'). Notice that since there are at most poly(n) such intervals, it suffices to prove this holds w.h.p. for a single Y' and the claim follows by the union bound.

To prove the claim, we also define $\gamma \triangleq |Y' \cap \mathcal{S}_i| + 2$ and use γ to get our bounds.

Concentration Analysis for γ. First, we show concentration for γ using the quantities we care to bound. We observe that $\gamma - 2 \in \frac{|p^i_* \cap Y'|}{\zeta} + [0,1]$, and hence $\gamma \in \frac{|p^i_* \cap Y'|}{\zeta} + [2,3]$. Now, the quantity $|p^i_* \cap Y'|$ can be lower bounded by $\texttt{Bin}\left(\sum_{w\in X_i} \mathbf{1}[y(w) \cap Y' \neq \emptyset], \zeta\frac{\tau/\lambda}{|X|}\right) - \zeta$, which w.h.p. is at least:

$$\frac{\tau\zeta}{\lambda} \cdot \frac{\sum_{w\in X_i} \mathbf{1}[y(w)\cap Y'\neq\emptyset]}{|X|} - 2\zeta = \zeta \cdot \left(\frac{\sum_{w\in X_i} \mathbf{1}[y(w)\cap Y'\neq\emptyset]}{\lambda r} - 2\right).$$

Hence $\gamma \geq \sum_{w\in X_i} \mathbf{1}[y(w) \cap Y' \neq \emptyset] / (\lambda r)$, and since $|Y' \cap \mathcal{S}_i| + 2 = \gamma$, we obtain part (1) of the claim.

To prove (2), we show that our sampling procedure guarantees that most blocks $w \in X_i$ will have bounded number of values in Y'.

All blocks in $y(X_i)_{-\lambda/2,Y'}$ contain $O(\gamma)$ elements in Y'. Let $T \triangleq \max_{w \in y(X_i)_{-\lambda/2,Y'}}\{|y(w) \cap Y'|\}$. We argue $T = O(\gamma)$. By construction, we have $\frac{\lambda r}{2}$ blocks in $y(X_i)$ with at least T integers in Y'. So, the quantity $|p^i_* \cap Y'|$ is lower bounded by $T \cdot \texttt{Bin}\left(\frac{\lambda r}{2}, \zeta\frac{\tau/\lambda}{|X|}\right)$. Hence, by Chernoff bound, we have

$$\zeta\gamma \geq |p^i_* \cap Y'| \geq T \cdot \left(\tfrac{\zeta}{3} - O(\log n)\right) > \tfrac{\zeta T}{4}$$

w.h.p. (for ζ with a high enough constant). This implies that $T \leq 4\gamma$.

Finally, observe that the quantity $|p^i_* \cap Y'|$ can also be lower bounded by viewing it as sampling blocks from $y(X_i)_{-\lambda/2,Y'}$, whose max-size (in Y') is T. Hence, we have, by the Chernoff bound, w.h.p.:

$$|p^i_* \cap Y'| \geq \zeta\tfrac{\tau/\lambda}{2|X|} \cdot |y_{-\lambda/2}(X_i, Y')| - \tfrac{1}{4}\zeta T.$$

This implies that $|y_{-\lambda/2}(X_i, Y')|/2(\lambda r) \leq \gamma + \gamma$, which further implies part (2) as well.

□

4.7 Genuine-LIS and Block-LIS extensions: proof of Lemmas 78 and 83

In this section we prove the reductions regarding the Genuine-LIS and Block-LIS extensions, namely Lemmas 78 and 83.

4.7.1 Block-LIS extension

For the Block-LIS problem, all we need to show is that one can "ignore" not too many exceptionally large blocks. The algorithm is the natural one:

1. Using Corollary 74 and T, sample blocks $S \subseteq X$ with i.i.d. probability $\frac{\zeta}{\lambda|X|}$, and let $W \leftarrow \{(i,j) \in S \times [k] \mid y_{i,j} \in Y\}$.

2. Compute $\Upsilon \leftarrow$ the $(1-0.6\lambda)$-quantile of $\{|\{(i,j) \in W\}|\}_{i \in S}$.

3. Run $\mathcal{A}$ with $\lambda' = 0.3\lambda$. Whenever $\mathcal{A}$ tries to access a block i using T, we check first if $|y_{i,*} \cap Y| > \Upsilon$, and if so, we treat the block as "empty" (say, by override).

Proof of Lemma 78. First, we argue that each block in $y(X)_{-\lambda/2,Y}$ has at least Υ elements in Y. Indeed, that holds w.h.p.by the Chernoff bound. Therefore, the instance we approximate is contained in $y_{-\lambda/2}(X,Y)$. It remains to show the additive error bounds. Here again, by the Chernoff bound, we "omit" at most $0.7\lambda n$ blocks, and hence the total additive error is at most $0.3\lambda n + 0.7\lambda n = \lambda n$ w.h.p. This concludes the proof. □

4.7.2 Genuine-LIS extensions

For the Genuine-LIS problem, we need to show two extensions:

4.7.2.1 Sparse instances

When the instance g consists of $m \ll nk$ non-null elements[6], we would like to improve the approximation and runtime bounds to be a function of m/n instead of k.

To obtain the improved bound, we first exponentially discretize the blocks based on the number of non-null elements (denoted the *block size* m_i) and output the maximum over all instances. For each scale of block size $b \in E_2(k)$, we use the following

[6]Note that unlike "sparse genuine" instances where we have few genuine elements, here we have few elements overall.

extended version of Genuine-LIS:

Lemma 89 (EstimateGenuineLIS algorithm, sparse version)**.** *Suppose the algorithm* EstimateGenuineLIS $(a(\lambda/k), \lambda n)$*-approximates Genuine-LIS for some function a in time $t(n \cdot k)$, then there exists an algorithm $\mathcal{A}$ that $(O(a(\lambda n/\mathsf{d}k)), \lambda n)$-approximates Genuine-LIS with d non-empty blocks. The run-time is $t(\mathsf{d}k) + O(n)$.*

We adapt algorithm EstimateGenuineLIS as follows. Let D be the set of non-empty blocks. We proceed by following the same algorithmic steps, using parameters $n' \triangleq \mathsf{d}$ and $\lambda' \triangleq \lambda n/\mathsf{d}$.

Proof. It suffices to show that the algorithm output guarantees do not deteriorate from the case where the input is $g \in \mathbb{N}^{\mathsf{d} \times k}$ with parameter λ' as above and does not include the empty blocks. Note that the additive error is $\lambda n = \lambda' \mathsf{d}$.

Indeed, for the dense estimator, we only count $i \in D$ which are subsampled, and each one is independently subsampled with the same probability as above.

For the sparse estimator, we note that the guarantees of Theorem 95 are with respect to $\frac{|Y \cap y|}{|X|}$, and increasing n can only help, since it gives us greater flexibility for choosing $\tau \in E_\beta$.

The runtime is immediate, as the only overhead is skipping empty blocks. □

4.7.2.2 Genuine-LIS over interval space

We also need to overcome the obstacle that the space $\mathcal{I}$ admits a partial order relation only, as opposed to $\mathbb{N}$, which admits a total order relation. Our ideal solution involves a mapping $\varphi : \mathcal{I} \to \mathbb{N}$ that approximately preserves the overall LIS over all subsequences (with probability 1), and therefore also preserves the overall LIS over

the genuine elements. We first note that such a mapping is applied a priori on the first coordinates of the input, and is not affected by the genuineness flags, nor does it require access to any genuineness samples.

While we are unable to exhibit a single mapping for the entire space, we show that one can discretize an arbitrary Genuine-LIS instance over intervals into $\log(k/\lambda)$ "nicer" instances $G_1, \ldots, G_{\log(k/\lambda)}$, which are mutually disjoint and whose union covers most of the original Genuine-LIS intervals (the unused intervals are discarded, incurring some additive error). Each such "nicer" instance is then mapped to a Genuine-LIS instance over integers, incurring a constant factor approximation over all its subsequences. The final output is the maximum Genuine-LIS over all integral instances $\{\varphi_i(G_{i,1}), G_{i,2}\}_i$. This costs us another $\log(k/\lambda)$-factor approximation.

In particular, we use the following claim:

Claim 90. *For $\ell \in \mathbb{N}$, define $\mathcal{I}_\ell$ to be the collection of all intervals in $\mathcal{I}$ whose length is in $[\ell, 2\ell)$. Then, there exists a mapping $\varphi_\ell : \mathcal{I}_\ell \to \mathbb{N}$, which can be computed in constant time, such that:*

1. *For $I, J \in \mathcal{I}_\ell$: $I < J \Rightarrow \varphi_\ell(I) < \varphi_\ell(J)$; and,*
2. *For all increasing sequences $y \in \mathbb{N}^m$ and all $q \in \mathcal{I}_\ell^m$ such that $\varphi_\ell(q) = y$, q must have an increasing subsequence of length $m/3$.*

Proof. We use the mapping $\varphi_\ell(I) \triangleq \lfloor \min(I)/\ell \rfloor$. First we show (1). We have

$$I < J \Rightarrow \min(I) \leq \max(I) - \ell + 1 \leq \min(J) - \ell,$$

which implies that $\varphi_\ell(I) < \varphi_\ell(J)$.

For (2), fix y as above, and for $i \in \{0,1,2\}$, let y^i be the sequence y restricted to integer values y_w satisfying $y_w \pmod 3 = i$. Let q^i be the corresponding subsequences of q. Then each subsequence q^i is increasing. To see this, consider two intervals q^i_j and q^i_{j+1}. Then $\max(q^i_j) < \min(q^i_j)+2\ell$. On the other hand, $\min(q^i_{j+1}) > \min(q^i_j)+2\ell$, as $y^i_{j+1} \geq y^i_j + 3$. So, $\max(q^i_j) < \min(q^i_{j+1})$.

Since $|q^0| + |q^1| + |q^2| = m$, at least one of them satisfies the length promise as well. □

Algorithm 19 Genuine-LIS extension reduction

Input: $g \in (\mathcal{I} \times \{0,1\})^{n \times k}, \lambda \in [1/n, 1]$, algorithm $\mathcal{A}$ for Genuine-LIS over integers.

Output: A number $\widehat{G} \in [0, n]$.

Let $y \leftarrow g(*)_1$ be the first coordinates of g.

Compute the set of integers $Q \leftarrow \cup_{i,j}\{\min(y_{i,j}), \max(y_{i,j})\}$.

Compute $a \in \mathcal{I}^{n \times k}$ such that $a_{i,j} \leftarrow [\text{rank}_Q(\min(y_{i,j})), \text{rank}_Q(\max(y_{i,j}))]$ for all i, j.

for $(\ell, i) \in E_2(2k/\lambda) \times [n]$ **do**

  Compute $U_\ell \leftarrow \{(i,j) \mid |a_{i,j}| \in [\ell, 2\ell)\}$.

  Compute $b^{(\ell)} \leftarrow \varphi_\ell(a(U_\ell)) \oplus g(U_\ell)_2$, where φ_ℓ is the transformation of Claim 90.

  for $e \in E_2(k)$ **do**

    Compute $B \leftarrow \{i \mid |b_i| \in [e, 2e)\}$.

    $L_{\ell,e} \leftarrow \frac{1}{3} \cdot$ { the output of algorithm from Lemma 89 with inputs $b^{(\ell)}_B, \lambda/2$ }.

  end for

end for

return $\max_{\ell,e} L_{\ell,e}$.

Figure 12: Description of the Genuine-LIS extension algorithm (Lemma 83). Here, $\oplus$ denotes direct sum.

Proof of Lemma 83. Let $\mathcal{A}$ be such that $\mathcal{A}(b^{(\ell)})$ is a $(\alpha, \lambda n/2)$-approximation for Genuine-LIS$(b^{(\ell)})$ in time t. We need to show the reduction algorithm provides a $(O(\log(1/\lambda)\alpha), \lambda n)$-approximation for g.

For the upper bound, we use Claim 90, item (1), to map each interval to an integer preserving its strict order relation with all other intervals. This implies that we always output an upper bound of the original interval LIS, as we only use subsets of elements of the original sequence, and do not add elements.

For the lower bound, first consider intervals of length at least $2k/\lambda$, which do not participate in any $b^{(\ell)}$. Any increasing subsequence of intervals contained in $[1, 2nk]$ can have at most $\lambda n/2$ intervals of such magnitude, hence by ignoring such intervals we "omit" at most $\lambda n/2$ LIS elements (additive error). Now, consider all intervals of length at most $2k/\lambda$. Each such interval participates in some instance $b^{(\ell)}$, and hence there exists an instance such that the intervals participating in it $(\log(k/\lambda), \frac{\lambda}{2}n)$-approximates the original instance. Now, using Claim 90, item (2), this instance is mapped to an integer instance which $(O(\log(k/\lambda)), \frac{\lambda}{2}n)$-approximates the original instance. Therefore, the lower bound holds for at least one $b^{(\ell)}$.

Next, we argue that partitioning $b^{(\ell)}$ into $\log(k)$ instances of similar block size guarantees that at least one instance has $1/\log(k)$ fraction of the LIS. Finally, we use the guarantees of Lemma 89, together with the above bounds on $b_B^{(\ell)}$, to conclude that $\mathcal{A}(b_B^{(\ell)})$ outputs a $(\alpha \cdot O(\text{polylog}(k/\lambda)), \lambda n)$-approximation of the original LIS. This concludes the proof.

□

4.8 Proof of Theorem 95

The proof is by recursive application of Theorems 68 and 69 for solving Block-LIS, up to approximation $\left(a\left(|X|,\tau,\frac{\lambda\cdot|X|}{|y(X,Y)|}\right),\lambda|X|\right)$, with access to a Precision-Tree $T_{1/c}$, for $c=c\left(|X|,\tau,\lambda,\frac{\lambda\cdot|X|}{|y(X,Y)|}\right)$.

We will establish the recurrences for quantities $a(\cdot,\cdot,\cdot)$ and $c(\cdot,\cdot,\cdot,\cdot)$.

Claim 91. *Fix $\epsilon<1$ and $\xi,\theta,n\in\mathbb{N}$. Consider the following recursive formulas:*

$$a(m,\tau,1/L)\leq\xi\cdot\max_{t\in[1,O(L)]}\max\left\{\gamma,a(\tau,\tau^{\epsilon},\tfrac{\gamma}{t})\right\}\cdot a(m/\tau,\tau,\Omega(t/L));$$

$$c(m,\tau,1/D,1/L)\leq\theta\cdot\tau\cdot D+\max_{r\in[1,D]}\left(\tilde{O}(r)+c(\tau,\tau^{\epsilon},1/r,\gamma/t)\right)\cdot c(m/\tau,\tau,\Omega(r/D),\Omega(t/L)).$$

With the following base cases:

$$a(\cdot,\cdot,1)=1 \qquad a(m,\cdot,1/m)=1 \qquad a(1,\cdot,\cdot)=1$$

$$c(\cdot,\cdot,\cdot,1)=1 \qquad c(1,\cdot,\cdot,\cdot)=1 \qquad c(m,\cdot,\cdot,1/m)=m$$

Then, there exists a function $\gamma(t,n,m,\epsilon)$, such that substituting $\gamma=\gamma(t,n,m,\epsilon)$ in the recursive formulas above yield the following upper bounds, for $H\triangleq\epsilon^{\Theta(1/\sqrt{\epsilon}\cdot\log 1/\epsilon)}$:

$$a(n,n^{\epsilon},1/L)=L^{\sqrt{\epsilon}}\cdot\xi^{O(H^2)}$$

$$c(n,n^{\epsilon},1/D,1/L)=D\cdot m^{O(\epsilon\log_{(1/\epsilon)}(H))}\cdot\theta^{O(H)}.$$

The proof of Claim 91 is deferred to Section 4.8.1.

To prove Theorem 95, we deduce the recurrences on approximation and Precision-Tree complexity showing they match the ones from Claim 91, from which we directly

obtain the final bounds on approximation. Then we analyze the runtime, where we use the bound established on $c(\cdot,\cdot,\cdot,\cdot)$ as well.

Proof of Theorem 95. To deduce the recurrences, we first note that for base cases, we have $a(\cdot,\cdot,1)=1$, $c(\cdot,\cdot,\cdot,1)=1$ as in this case $\lambda=1$ and additive approximation $|X|$ is trivial. Similarly, for $|X|=1$ (i.e., 1 element string), we have $a(1,\cdot,\cdot)=1$, $c(1,\cdot,\cdot,\cdot)=1$, as we can compute if LIS is 1 or 0 with access to a 1 element string. Similarly, we need full access for exact approximation, and hence: $a(m,\cdot,1/m)=1$, $c(m,\cdot,\cdot,1/m)=m$. For the other cases, we show the following:

Approximation. For convenience, let $m \triangleq |X|$, $L \triangleq \frac{|y(X,Y)|}{\lambda\cdot|X|} \leq k/\lambda$. We apply Theorem 69. Define $t \triangleq k_g/\lambda_g$, in which case $\frac{\lambda_s|X|}{|y(X)\cap Y|} = \Omega(t/L)$. Note that if $t > \Omega(L)$, then, we get (base case) $a(|X|,\cdot,1)=1$, and hence $a(m,\tau,1/L) \leq (\log k/\lambda)^{O(1)}\cdot \max_{t\leq 1/\lambda} a_g(\tau,1/t)$. Thus, including when t is smaller, we have that:

$$a(m,\tau,1/L) \leq (\log k/\lambda)^{O(1)} \cdot \max_{t\in[1,O(L)]} a_g(\tau,1/t)\cdot a(m/\tau,\tau,\Omega(t/L)).$$

Plugging in a_g from Theorem 68, (with $\tau_s = n^\epsilon$), we get, for our choice of γ:

$$a(m,\tau,1/L) \leq (\log k/\lambda)^{O(1)} \cdot \max_{t\in[1,O(L)]} \max\left\{\gamma, a(\tau,\tau^\epsilon,\tfrac{\gamma}{t})\right\}\cdot a(m/\tau,\tau,\Omega(t/L)).$$

Precision-Tree Complexity. We now deduce the complexity bound c. Note that the recursion for c will be using same parameters as above. We need to keep track of new parameters, in particular let $D = 1/\lambda$. We have, where λ_s,λ_g,k_g are as in approximation bound above, with $\lambda_s\lambda_g = \Omega(k_g\lambda)$:

$$c(m,\tau,1/D,1/L) \leq \text{polylog}(n)\cdot\tau D + (c_g(\tau,\lambda_g,k_g)\cdot c(m/\tau,\tau,\Theta(\lambda_s/k_g),\Omega(t/L))$$

Here we set $r = 1/\lambda_g \in [1, D]$ and hence:

$$c(m, \tau, 1/D, 1/L) \leq \text{polylog}(n) \cdot \tau L + c_g(\tau, 1/r, k_g) \cdot c(m/\tau, \tau, \Omega(r/D), \Omega(t/L)).$$

Plugging in c_g from Theorem 68 (again, with $\tau_s = n^\epsilon$), we get:

$$c(m, \tau, 1/D, 1/L) \leq \text{polylog}(n) \cdot \tau D + \max_{r \in [1,D]} \left(\tilde{O}(r) + c(\tau, \tau^\epsilon, 1/r, \gamma/t) \right) \cdot c(m/\tau, \tau, \Omega(r/D), \Omega(t/L)),$$

We now apply Claim 91, with $\theta = \text{polylog}(n)$ and $\xi = \text{polylog}(k/\lambda)$ to obtain the bounds we need.

Runtime complexity. The recurrence relation $t_s(n, n^\epsilon) = \kappa \cdot t_s(n^{1-\epsilon}, n^\epsilon)$, where $\kappa = \tilde{O}\left(\log^{O(\log_\beta(\tau))}(\beta)\right)$, with base case $t_s(n^\epsilon, n^\epsilon) = \kappa$, can be expanded to obtain $T(n, n^\epsilon) = \kappa^{1/\epsilon}$.

Therefore, the total expected runtime is $\kappa^{1/\epsilon} \cdot c(m, \tau, 1/D, 1/L) \cdot \frac{L}{D} = L \cdot m^{O(\sqrt{\epsilon} \log 1/\epsilon)} \cdot (\log n)^{2^{O(1/\sqrt{\epsilon} \cdot \log^2 1/\epsilon)}} \cdot \tilde{O}\left(\log^{O(\log_\beta(n^\epsilon))}(\beta)\right)^{1/\epsilon} = \log^{O(\log_\beta(n))}(\beta) \cdot \log(n)^{O(1/\epsilon)}$. The pre-processing of the Precision-Tree takes time $c \cdot \tilde{O}\left(\log^{O(\log_\beta(n^\epsilon))}(\beta)\right)$, which is a lower order term.

□

4.8.1 Recursion Analysis

We will prove each recursion separately. Let $b = n^{1/H}$. Our choice of $\gamma = t^{\sqrt{\epsilon}}$, except for $\tau = b$, then we set $\gamma = t$ and get the base cases $a(m, b, 1/L) \leq \xi \cdot \max_{t \in [1, O(L)]} t \cdot a(m/b, b, \Omega(t/L))$ and $c(m, b, 1/D, 1/L) \leq \theta \cdot b \cdot D + \max_{r \in [1,D]} \left(\tilde{O}(r) + c(m/b, b, \Omega(r/D), \Omega(t/L)) \right)$.

Proof of Claim 91, function a. Setting $\tau = m^\epsilon$, the recursion formula can be written as:

$$a(m, m^\epsilon, 1/L) \leq \xi^\Upsilon \cdot \max_{\substack{t_1, \ldots, t_{1/\epsilon} \geq 1 \\ t_1 \cdot t_2 \cdot \ldots \cdot t_{1/\epsilon} \leq 2^{O(\Upsilon)} L}} \prod_i \max\{\gamma_i, a(m^\epsilon, m^{\epsilon^2}, \gamma_i/t_i)\},$$

where $\Upsilon = 1/\epsilon$ and $\gamma_i = \gamma(t_i)$. Let $f = \Upsilon \cdot \log(\xi)$. Setting $q_h(l) = \log a(m^{\epsilon^h}, m^{\epsilon^{h+1}}, 1/2^l)$, we obtain from above, and substituting γ_i with $\log \gamma_i = \sqrt{\epsilon} t_i$:

$$q_h(l) \leq f + \max_{\substack{t_1,\ldots,t_\Upsilon \geq 0 \\ t_1+t_2+\ldots+t_\Upsilon \leq l+O(\Upsilon)}} \sum_i \max\{\gamma_i, q_{h+1}(t_i - \gamma_i)\}.$$

Now let $H' = \log_\Upsilon(H)$ and hence $b = n^{1/H} = n^{\epsilon^{H'}}$. Then $q_{H'}(l) \leq f + l + O(\Upsilon)$ by the base case.

Now for $h \leq H'$ we prove, by induction, that

$$q_h(l) \leq (f + O(\Upsilon)) \cdot \Upsilon^{2(H'-h)} + (1 - \sqrt{\epsilon})^{H'-h} \cdot l.$$

Indeed, we first check the base case for $h = H'$: $q_{H'}(l) \leq f + l + \Upsilon$. Now note that $q_h(t_i - \gamma(t_i)) \leq f \cdot \Upsilon^{H'-h} + (1 - \sqrt{\epsilon})^{H'-h+1} \cdot t_i$. Hence, we can verify the inductive step for $h < H'$:

$$\begin{aligned}
q_h(l) \leq & f + \max_{\substack{t_1,\ldots,t_\Upsilon \geq 0 \\ t_1+t_2+\ldots+t_\Upsilon \leq l+O(\Upsilon)}} \sum_i \max\{\gamma(t_i), q_{h+1}(t_i - \gamma(t_i))\}. \\
\leq & f + \max_{\substack{t_1,\ldots,t_\Upsilon \geq 0 \\ t_1+t_2+\ldots+t_\Upsilon \leq l+O(\Upsilon)}} \sum_i (f + O(\Upsilon)) \cdot \Upsilon^{2(H'-h-1)} + (1 - \sqrt{\epsilon})^{H'-h} \cdot t_i \\
\leq & f + (f + O(\Upsilon)) \cdot \Upsilon^{2(H'-h)-1} + (1 - \sqrt{\epsilon})^{H'-h}(l + O(\Upsilon)) \\
\leq & (f + O(\Upsilon)) \cdot \Upsilon^{2(H'-h)} + (1 - \sqrt{\epsilon})^{H'-h} \cdot l
\end{aligned}$$

where we omitted $\gamma(t_i)$ from the max in the 2nd inequality since $\sqrt{\epsilon} t_i$ is smaller than the inductive hypothesis on $q_{h+1}(t_i)$ (using the particular choice on the value of H).

In particular, this bound implies $q_0(l) = O(f) \cdot \Upsilon^{2H'} + \sqrt{\epsilon} l = f \cdot O(H^2 \cdot \Upsilon) + \sqrt{\epsilon} l$.

Substituting back, we obtain that $a(m, m^\epsilon, 1/L) \leq 2^{O(H\Upsilon) \cdot f + \sqrt{\epsilon} \log(L)} \leq L^{\sqrt{\epsilon}} \cdot \xi^{O(H^2)}$.

□

Proof of Claim 91, (function c). Similar to the previous proof, setting $\tau = m^{\epsilon}$ with t_i, Υ from before too:

$$c(m, m^{\epsilon}, 1/D, 1/L) \leq (\log(n))^{O(\Upsilon)} \cdot \left(\sum_{i=1}^{\Upsilon} \frac{m^{\epsilon} D/r_1..r_{i-1}}{\log n} \prod_{j<i} c(\tau, \tau^{\epsilon}, 1/r_j, \gamma(t_j)/t_j) \right),$$

where the $r_i's$ satisfy $r_1, \ldots r_{\Upsilon} \geq 1$ and $r_1 r_2 \ldots r_{\Upsilon} = D$. We further can simplify by dividing by D:

$$\tfrac{1}{D} c(m, m^{\epsilon}, 1/D, 1/L) \leq (\log(n))^{O(\Upsilon)} \cdot \tfrac{m^{\epsilon}}{\log n} \cdot \left(\sum_{i=1}^{\Upsilon} \prod_{j<i} \tfrac{1}{r_j} c(\tau, \tau^{\epsilon}, 1/r_j, \gamma(t_j)/t_j) \right).$$

As before, let $c_h(D) = \frac{1}{D} c(m^{\epsilon^h}, m^{\epsilon^{h+1}}, 1/D, 1/L)$, for which we get:

$$c_h(D) = (\log n)^{O(\Upsilon)} \cdot m^{\epsilon^{h+1}} \cdot \max_i \prod_{j<i} c_{h+1}(r_j),$$

with the base case $c_{H'}(D) = (\log n)^{O(1)}$ (when we are taking the maximal γ). Hence we obtain that

$$c_0(D) \leq (\log n)^{\Upsilon^{O(H')}} \cdot m^{\epsilon \cdot H'}.$$

Plugging back into c we obtain that:

$$c(m, m^{\epsilon}, 1/D, 1/L) \leq D \cdot (\log n)^{\Upsilon^{O(H')}} \cdot m^{\epsilon \cdot H'}.$$

□

Chapter 5

Longest Common Subsequence

Given a pair of strings $x, y \in \Sigma^n$, a common subsequence between x and y is a subsequence which occurs in both strings. The Longest Common Subsequence (LCS) of x and y, is a common subsequence between x and y of maximal length. We denote by $\mathsf{LCS}(x, y)$ the size of the Longest Common Subsequence.

The (length of the) LCS has a classical dynamic programming solution in quadratic time [202].

LCS is also a fundamental string similarity measure of key importance both in theory and practice, and therefore has been extensively studied during the course of the past 50 years [199; 116; 117; 35; 93; 194; 185; 186], with the main goal of improving the run-time from polynomial towards close(r) to linear.

Despite significant attention, little progress was made, and the runtime has improved by logarithmic factors only [150]. More recently, fine-grained hardness results helped explaining the lack of progress, showing that any truly subquadratic solution for LCS would refute the Strong Exponential Time Hypothesis (SETH)[1; 65]. Moreover, for the LCS problem in particular, it was later shown that even a high

polylogarithmic speedup would be surprising [3; 2].

Even before the above hardness results, researchers started considering faster algorithms that approximate the LCS, but there as well the progress has been slow. The trivial $\sqrt{n}$-factor approximation was only recently improved to $\approx n^{0.497}$ [112] and even more recently to $n^{0.4}$ [61]. In parallel, a parametrized version was developed which gives a subquadratic solution with improved approximation when the longest common subsequence is promised to be above a certain threshold [185].

In this chapter, we show an algorithm which runs in $O(n)$ time, and outputs a $n^{o(1)}$-factor approximation to $\mathsf{LCS}(x, y)$, with high probability, for any pair of length n input strings, proving Theorem 5, restated below:

Theorem 92 (Theorem 5, restated)**.** *LCS_n can be approximated up to $n^{o(1)}$-factor, with high probability, and in $O(n)$ time.*

While we do not track the exact approximation factor in our analysis, we note it is $\exp\left(\frac{\log(n)}{\log^c \log(n)}\right)$ for some absolute constant c.

Our entire algorithm is merely an efficient black-box reduction to the Block-LIS problem, introduced very recently in [29], and solving the Block-LIS problem directly.

5.1 Preliminaries

5.1.1 Definitions

Sequences. Fix alphabet Σ. We denote $x \in \Sigma^n$ as a length n sequence.

For a sequence x, we denote by $\psi(x) : \Sigma \to \mathbb{N}$, the count vector of letters in x, i.e., $\psi(x)_c \triangleq |x^{-1}(c)|$ for any $c \in \Sigma$.

We also define integer block sequence $z \in \mathbb{N}^{n \times k}$, and denote $|z|$ as the count of all integers in all the blocks (with repetitions).

Monotone sets. We define monotone sets as follows:

Definition 93. *We say that a set $P \subseteq \mathbb{N} \times \mathbb{N}$ is* monotone *iff for all $((i,k),(j,l)) \in P \times P$, we have (i) $i = j \Leftrightarrow l = k$; and (ii) $i < j \Leftrightarrow k < l$.*

One can observe that

$$\mathsf{LCS}(x,y) = \max_{\substack{P \subseteq [n] \times [n] \\ P \text{ is monotone}}} \sum_{(i,j) \in P} \mathbf{1}[x_i = y_j]$$

The Block-LIS problem. The Block-LIS problem was defined in [29][1]. In Block-LIS, the main input consists of n blocks of (at most) k elements each, and each block can contribute at most one of its elements to a subsequence. Formally, for an integer block sequence $z \in \mathbb{N}^{n \times k}$, $\mathsf{Block\text{-}LIS}(z)$ is the length of a maximal sub-sequence $\mathsf{OPT} \triangleq \{(w_1, z_w)\}_{w \in [n] \times [k]}$ such that OPT is monotone.

Other notations. Fix vectors $a, b \in \mathbb{R}^n$, then $\min(a,b)$ (respectively $\max(a,b)$) is defined, as the vector of minimums (respectively, maximums) for each (a_i, b_i) pair. Also, the notation $\oplus$ means the direct sum.

5.1.2 Main technique - Block-LIS reduction

Our construction boils down to a formulation of the LCS problem as a (larger) LIS problem, and solve the LIS problem directly. Our reduction at its core is reminiscent to ideas introduced in [117; 35], with two important differences:

[1]The original definition of Block-LIS included another extension which is not relevant for our reduction, and hence was omitted for exposition.

1. We provide a simplified approach by direct "black-box" reduction to a LIS problem which is then solved directly. In particular, we reduce to the Block-LIS problem. As a result, the entire construction and proof takes roughly 1.5 pages.

2. We show that the number of "matching pairs" is always bounded by $2n \cdot \mathsf{LCS}(x, y)$ using norm inequalities, implying the linear runtime from Lemma 4.3 in [29].

In particular, we restate the following Definition and Theorem:

Definition 94. *[29] For $\alpha \geq 1$ and $\beta > 0$, an (α, β)-approximation $\hat{q}$ of a quantity q is an α-multiplicative and β additive estimation of q, i.e., $\hat{q} \in [q/\alpha - \beta, q]$.*

Theorem 95. *[29] Fix $n, k \in \mathbb{N}$ and $z \in \mathbb{N}^{n \times k}$. For any $\lambda = o(1)$, there exists a randomized non-adaptive algorithm $\mathcal{A}_{BL}$ that solves Block-LIS up to $(\alpha, \lambda n)$-approximation, where $\alpha = \left(\frac{|z|}{\lambda n}\right)^{o(1)}$ using $\left(\frac{|z|}{\lambda n}\right) \cdot n^{o(1)}$ time.*

Block-LIS Formulation for LCS. To prove Theorem 92, we formulate the LCS problem as a single string Block-LIS. Given $x, y \in \Sigma^n$, we define an integer block sequence z such that each z_i is a block of all $j \in [n]$ indices where $y_j = x_i$. Formally:

$$z \triangleq \oplus_{i \in [n]} \left\{ y^{-1}(x_i) \right\}. \tag{5.1}$$

We claim the following equivalence:

Claim 96. *Block-LIS(z) = LCS(x, y).*

Proof. First, observe that any common subsequence of x and y has a monotone set $P \subseteq [n] \times [n]$ where $x_i = y_j$ for all $(i, j) \in P$. Now, since $j \in y^{-1}(x_i)$, then the second coordinates of P are also an increasing subsequence of z.

Similarly, consider the coordinates of an increasing subsequence $W \subseteq [n] \times [k]$. Then $\{(w_1, z_w)\}_{w \in W}$ is a monotone set where $x_{w_1} = y_{z_w}$, and hence x and y has a common subsequence of the same length. □

Our entire algorithm is merely an efficient implementation of the above reduction, and solving it using Theorem 95. In particular, we show the following:

Lemma 97. *Suppose there exists an algorithm $\mathcal{A}_{BL}$ which $\left(f\left(\frac{|z|}{\lambda n}\right), \lambda n\right)$ approximates Block-LIS in time $t\left(\frac{|z|}{\lambda n}, n\right)$. Then, there exists an algorithm $\mathcal{A}_{LCS}$ approximating LCS_n up to a factor $(1 + o(1)) \cdot f(n \cdot \log n)$ in $t\left(O(n \cdot \log(n)), n\right) + O(n)$ time.*

Proof of Theorem 92 using Theorem 95 and lemma 97. The proof follow almost immediately from Theorem 95 and lemma 97. However, a direct invocation of the statements only yields $n^{1+o(1)}$ time algorithm. To obtain linear time, we first subsample $U \subseteq [n]$ i.i.d. at rate $n^{-o(1)}$, and then apply the statements on x_U and y_U, noting $\mathsf{LCS}(x_U, y_U) \in [n^{-o(1)}, 1] \cdot \mathsf{LCS}(x, y)$. The resulting runtime is $|U|^{1+o(1)} = O(n)$ as needed. □

5.1.3 Useful lower bound for the LCS using Hölder's inequality

Hlder's inequality. A classical norm inequality which says that $\langle x, y \rangle \leq \|x\|_{1/p} \cdot \|y\|_{1/q}$ for any $p + q = 1$. In particular, $\langle x, y \rangle \leq \|x\|_\infty \cdot \|y\|_1$. Such inequality can be extended to show the following bound:

Claim 98. *Fix $x, y \in \mathbb{R}^n$. Then, $\langle x, y \rangle \leq \|\min\{x, y\}\|_\infty \cdot (\|x\|_1 + \|y\|_1)$.*

Proof. Let $a = \min\{x, y\}$ and $b = \max\{x, y\}$. Then, by Hlder's inequality,

$$\langle x, y\rangle = \langle a, b\rangle \leq \|a\|_\infty \cdot \|b\|_1 = \|\min\{x, y\}\|_\infty \cdot \|\max\{x, y\}\|_1 \leq \|\min\{x, y\}\|_\infty \cdot (\|x\|_1 + \|y\|_1) .$$

□

Simple lower bound for the LCS. We next state a useful fact about the minimal LCS of two strings:

Fact 99. $LCS(x, y) \geq \|\min\{\psi(x), \psi(y)\}\|_\infty$.

Proof. That fact is immediate, since if each of x and y contain k copies of the same letter c, then c^k is a common subsequence to both x and y. □

We combine the above to show the following lower bound on the LCS:

Lemma 100. $LCS(x, y) \geq \frac{|z|}{2n}$.

Proof. Observe that by construction $|z| = \langle \psi(x), \psi(y)\rangle$. Now, invoking Claim 98 and fact 99, we obtain:

$$|z| \leq \|\min\{\psi(x), \psi(y)\}\|_\infty \cdot (\|\psi(x)\|_1 + \|\psi(y)\|_1) \leq 2n \cdot \mathsf{LCS}(x, y),$$

which implies the Lemma. □

Remark 101. *One direct implication of the above is a simple deterministic and exact algorithm for LCS in time $O(n\ell \log(\ell))$, where $\ell = LCS(x, y)$. While we do not derive it in this paper, one can observe this can be done by simply reducing the LCS problem to Block-LIS, and solving it directly as a standard LIS problem, which takes $O(|z| \log(LIS(z)))$ time.*

5.2 Reduction algorithm

In this section we prove Lemma 97. Namely, we are given an algorithm $\mathcal{A}_{\mathrm{BL}}$ for solving $\mathsf{Block\text{-}LIS}(z)$ with additive error parameter λ, and we use it to estimate $\mathsf{LCS}(x, y)$.

The algorithm is the straight-forward one:

EstimateLCS(x, y):

1. Store the function y^{-1} over Σ (say, in the sparse matrix representation).
2. Set $d \leftarrow |z|/2n$, where z is defined as in Equation (5.1).
3. Run $\mathcal{A}_{\mathrm{BL}}$ with input z and $\lambda \leftarrow \frac{d}{n \log n}$.
4. Output $\widehat{\ell} \leftarrow$ the maximum between d and the output of $\mathcal{A}_{\mathrm{BL}}$.

Figure 13: Description of the algorithm for estimating the LCS between x and y

5.2.1 Analysis

We show the above algorithm provides the guarantees of Lemma 97.

Proof of Lemma 97. First, let us argue $\widehat{\ell} \in \left[\frac{1-o(1)}{f(n \log n)}, 1\right] \cdot \mathsf{LCS}(x, y)$.

Upper Bound. The upper-bound is immediate since the output of $\mathcal{A}_{\mathrm{BL}}$ is promised to be at most $\mathsf{Block\text{-}LIS}(z) = \mathsf{LCS}(x, y)$, and since $d \leq \mathsf{LCS}(x, y)$ from Lemma 100.

Lower Bound. Let $\widehat{\ell_{\mathcal{A}}}$ be the output of $\mathcal{A}_{\mathrm{BL}}$. Then, we are guaranteed

$$\widehat{\ell_{\mathcal{A}}} \geq \frac{\mathsf{Block\text{-}LIS}(z)}{f(|z|/\lambda)} - \lambda n = \frac{\mathsf{LCS}(x,y)}{f(n \log n)} - \frac{2d}{\log n}.$$

Therefore, $\widehat{\ell} = \max\{\widehat{\ell_{\mathcal{A}}}, d\} \geq (1 - o(1)) \cdot \frac{\mathsf{LCS}(x,y)}{f(n \log n)}$ as needed.

Last, we show the runtime is at most $t\left(O(n \cdot \log(n)), n\right) + O(n)$.

Runtime Analysis. The runtime for the first 2 steps is clearly $O(n)$. It remains to show the runtime on the Block-LIS algorithm, which is promised to be at most

$$t\left(\frac{|z|}{\lambda n}, n\right) = t\left(\frac{2d}{\lambda}, n\right) = t\left(O(n \cdot \log(n)), n\right).$$

This concludes the proof.

□

Part II

Nearest Neighbour Search

Chapter 6

LSH Forest - Bit Sampling

In the Nearest Neighbor Search problem (NNS), we are given a set P of n points in a d-dimensional space, and the goal is to build a data structure that, given a query point q, reports any point from P within a given distance r to the query q. In the (c, r)-approximate near neighbor problem (ANN), the data structure may return any data point whose distance from the query is at most cr, for an approximation factor $c > 1$ (provided that there exists a data point within distance r from the query).

As discussed in Chapter 1, there is a longstanding gap between the classical, data-oblivious LSH algorithms, which are very practical but with limited theoretical guarantees, and the data-dependent ones, which are fastest asymptotically, but so far has been mostly impractical.

In this chapter, we show new, simple data-dependent algorithms that are *provably* better than data-oblivious LSH algorithms, even for *worst case* instances, proving Theorem 6, restated below:

Theorem 102 (Theorem 6, informal, restated)**.** *For every $k \geq 1$, there exists a procedure of choosing k pivots in each node of the tree of LSH Forest, such that, in*

order to get a data structure for (c, r)-ANN for the Hamming distance with constant probability of success, it is enough to sample $O(n^\rho)$ tries, where:

$$\rho \leq \frac{1}{\ln 4} \cdot \frac{1}{c} \cdot \left(1 + O\left(\frac{1}{k}\right)\right) + O\left(\frac{1}{c^2}\right).$$

Choosing k to be large enough, the exponent is $\rho = 0.73/c + O(1/c^2)$. The resulting query time is $O(n^\rho d^2 k)$ and space is $O(n^{1+\rho} k + nd)$.

While the above does not improve the bounds of existing DDH algorithms such as [31] quantitatively (in fact they are worse), the main advantage is simplicity and relation to algorithms used in practice.

Our improved algorithm is based on *LSH Forest*, a heuristic introduced in [43] and now popular in practice (see, e.g., [47]). It can be applied as a black box to any LSH hash family, but we will need it instantiated to the Bit Sampling LSH of [124; 114] for the Hamming space. We show that while using LSH Forest directly does not lead to improved algorithms, a simple modification of it does!

It will be convenient to assume that there is exactly one data point—the near neighbor—within distance cr from the query (and, thus, the data structure must recover the near neighbor). This assumption is adequate for practice: for many real-world datasets there are not too many data points that are not much further from the query than the near neighbor. At the same time, later we will show a modification of our algorithm that can handle the general case of (c, r)-ANN.

Our new, improved algorithm is based on the LSH Forest algorithm, which proceeds as follows when instantiated to the Bit Sampling LSH. We choose a random permutation of the coordinates $\pi : [d] \to [d]$ and apply it to each vector $x \in P$. Then

we build a *trie* on the permuted vectors.[1] For a given query q, we descent down the trie until we reach precisely one leaf and compare against the point stored in that leaf. As before, we boost the probability of success by building n^ρ such tries (hence the name of "forest").

While the above algorithm still requires $\rho \geq 1/c$ in the worst case, here is how we modify it to bypass the $1/c$ bound. For a constant parameter $k = O(1)$, and for every tree node v, we choose a fixed set of k *pivots* T_v. Then, when answering a query, while descending down the tree, we check against all the pivots T_v of every node v we visit on the way to the leaf. Overall, we compare the query point with at most dk data points. The following theorem summarizes the main result of this paper assuming that there is only the near neighbor within distance cr from the query.

To get rid of the assumption, we need to augment the set of pivots further. Namely, in addition to k carefully chosen pivots, we need to include $\Theta(\log n)$ *uniformly random pivots*. If we do this, Theorem 102 will hold for the original (c, r)-ANN with a logarithmic loss in space and query time.

Discussion of the theorem statement. Our resulting bound of, essentially, $\rho \approx \frac{1}{\ln 4 \cdot c}$ corresponds to the bound that follows from using the Bit Sampling LSH of [124; 114] on *a random dataset* in $\{0, 1\}^d$. Hence the algorithm can be seen as a worst-case–to–random-case reduction in the case of Bit Sampling LSH. While there exist space partitions that are better than Bit Sampling on a random dataset [17; 31; 16], they are less efficient, and, in particular require a *larger number of parts*, requiring a tree with a large branching factor. This seems required at least in some cases: in

[1] This can be also seen as applying kd-tree on P with for a *random order* of the coordinates, stopping the splitting until we have exactly one point.

particular, for the *Euclidean space*, [16] prove that, for any space partition where ρ differs from the optimal exponent by ϵ, the number of parts must be exponential in $1/\epsilon$. We conjecture a similar phenomenon happens for the Hamming case; indeed, in all the *known* constructions, we need at least 10^6 parts to improve upon the Bit Sampling for random instances, for $c = 2$. Furthermore, in all known space partitions, the decoding time is proportional to the number of parts, and hence it is a very important consideration in practice.

Discussion of the algorithm. We note that LSH Forest by itself is already a data-dependent algorithm, in that it auto-tunes to the dataset. In contrast to Bit Sampling that uses a fixed number of (random) coordinates, LSH Forest uses a variable number of coordinates, which may differ from one part of the tree to the other. Informally speaking, local granularity of a partition depends on the local density of a region being partitioned, which is beneficial for irregular datasets. Nonetheless, despite the vanilla LSH Forest working well in practice, one can still construct a bad example, where one has to sample $n^{\frac{1}{c}-o(1)}$ trees. In particular, in the *worst case* LSH Forest by itself does not offer any advantage over the regular Bit Sampling. It is precisely the carefully selected pivots that allow LSH Forest to achieve the improved bound.

We note that our algorithm can also be seen as the LSH Forest essentially instantiated to the *min-hash* scheme [66; 67], another very popular (data-independent) hash function for the Hamming space, used extensively in practice. In particular, the min-hash function h chooses a random permutation π and, on point $x \in \{0,1\}^d$, outputs the smallest coordinate i in π where $x_i = 1$. The resulting space partition corresponds to a caterpillar tree with the permutation π. Hence, if we concatenate a few minhashes, and simplify the tree, we obtain precisely the above LSH Forest.

Again, pivots are essential to obtain improved bounds.

We also comment on the use of *randomness* in the tree. In the later sections, we use LSH Forest where different parts of the trie use *independent* randomness: i.e., in each node we choose a new random coordinate to split the dataset by, and recurse in each of the two parts corresponding to the dataset conditioned on the value in that coordinate. This does not affect the analysis (due to the linearity of expectation) Not sure that it's quite the reason., but is more general and likely amenable to further improvements.

A final remark regarding the analysis of our algorithm is that this is the first time when the analysis of random space partitions is with respect to the entire dataset. This is in contrast to the standard analysis of LSH [114], data-dependent hashing [31], or the analysis of [89], where the analysis of collision probability is with respect to *two* or *three* points. In this paper, we perform a global analysis, which turns out to be, technically, the most involved part of the paper.

6.1 Preliminaries

As is the case with most ANN algorithm, we use random partition of a metric space. For a point x and a partition $\mathcal{R}$, we denote $\mathcal{R}(x)$ the part of $\mathcal{R}$ the point x lies in.

Definition 103. *The* (c, r)-Approximate Near Neighbor problem (ANN) *is to construct a data structure over an* n*-point dataset* $P \subset X$ *lying in a metric space* X *supporting the following queries: given any fixed query point* $q \in X$*, if there exists* $p^* \in P$ *with* $d_X(q, p^*) \leq r$*, then report some* $p' \in X$ *such that* $d_X(q, p') \leq cr$*, with probability at least* 0.9*.*

It will be convenient to introduce the following simplification of Definition 103.

Definition 104. *The* (c, r)-Gap Approximate Near Neighbor problem *is to construct a data structure over an* n*-point dataset* $P \subset X$ *lying in a metric space* X *supporting the following queries: given any fixed query point* $q \in X$ *with the promise that there is a data point* $p^* \in P$ *with* $d_X(q, p^*) \leq r$ *and other data points are further than* cr *from the query* q*, the goal is to find* p^* *with probability at least* 0.9*.*

Definition 105 ([124; 114])**.** *We call a random partition* $\mathcal{R}$ *of a metric space* X *an* (r_1, r_2, p_1, p_2)-sensitive LSH partition*, if for every* $x_1, x_2 \in X$ *with* $d_X(x_1, x_2) \leq r_1$*, we have* $\Pr_{\mathcal{R}}[\mathcal{R}(x_1) = \mathcal{R}(x_2)] \geq p_1$*, while for every* $x_1, x_2 \in X$ *with* $d_X(x_1, x_2) > r_2$*, we have* $\Pr_{\mathcal{R}}[\mathcal{R}(x_1) = \mathcal{R}(x_2)] \leq p_2$*.*

Theorem 106 ([124; 114])**.** *Suppose that* $\mathcal{R}$ *is a* (r, cr, p_1, p_2)*-sensitive LSH partition of a metric space* X*. We assume that* $0 < p_1, p_2 < 1$ *and denote* $\rho = \log(1/p_1)/\log(1/p_2)$*. Then, there exists a data structure for* (c, r)-*ANN over a dataset* $P \subset X$ *that consists of at most* n *points such that:*

- *The query procedure requires* $O(n^\rho/p_1 \cdot \log_{1/p_2} n)$ *point locations in partitions sampled according to* $\mathcal{R}$ *and* $O(n^\rho/p_1)$ *distance computations and other operations;*
- *The data structure uses at most* $O(n^{1+\rho}/p_1)$ *words of space, in addition to the space needed to store the dataset* P *and to the space needed to store* $O(\log_{1/p_2} n)$ *partitions sampled according to* $\mathcal{R}$*.*

The failure probability of the data structure is at most 0.1*.*

In particular, the original LSH paper used the following Bit Sampling partition.

Definition 107. *For a Hamming space* $\{0,1\}^d$ *the* Bit Sampling LSH *partition [124; 114] is as follows: one first chooses a random coordinate* $i \in [d]$*, and the partition is* $\{x \mid x_i = 0\} \cup \{x \mid x_i = 1\}$*. It is easy to show that for every* r *and* c *it is* $(r, cr, 1 - r/d, 1 - cr/d)$*-sensitive. In particular,*

$$\rho = \frac{\log(1/p_1)}{\log(1/p_2)} = \frac{\log\left(1 - \frac{r}{d}\right)^{-1}}{\log\left(1 - \frac{cr}{d}\right)^{-1}} \leq \frac{1}{c}.$$

For the Bit Sampling LSH, the worst-case exponent of $\rho = 1/c$ occurs when $r \ll d$. On the other hand, for $r = d/2c$, one obtains $\rho = \frac{\log 1/2}{\log 1 - 1/2c} \approx \frac{1}{\ln 4} \cdot \frac{1}{c} + O(1/c^2)$ when $c \to \infty$. This case of $r = d/2c$ corresponds to the canonical random instance in the Hamming space: where the dataset is i.i.d. random in $\{0,1\}^d$ and the query point is planted within distance $d/2c$ from a random dataset point.

6.2 Generic LSH Forest with pivots

In this section we describe a generic construction and analysis of LSH Forest. This can be seen as a different way to use random partitions than the classical one, as in Theorem 106. We instantiate the general LSH Forest algorithm in later sections. LSH Forest was first introduced in [43] as a heuristic.

LSH Forest consists of several decision trees. Each tree is built and queried using Algorithm 20, which works as follows. In the root of the tree, we start with the entire dataset $S = P \subset X$. We sample a random partition $\mathcal{R}$ of X and group the points from S according to the parts of $\mathcal{R}$. We recurse on each non-empty group and add the resulting trees as children of the root node. We stop as soon as a current subset $S \subset P$ becomes of size 1. In each node v we store the pivot set, denoted T_v, the random partition sampled while building v, denoted by $\mathcal{R}_v$, and the list of children. During

the query stage, we descend down the tree, where at each node v, we follow the child that captures the query q in the partition $\mathcal{R}_v$. We also compute the distance to the points $p \in T_v$ for each traversed node v. One such tree succeed with a relatively low probability, and hence we build several independent trees by calling BUILDTREE(P), thus boosting this probability to 0.9. During the query stage, we query every tree.

Algorithm 20 Generic LSH Forest with pivots

```
1: function BuildTree(S)                          ▷ S is a subset of the dataset
2:     create a node v with associated set S
3:     compute pivots ∅ ≠ T_v ⊆ S, store them in v
4:     if |S| = 1 then
5:         return v.
6:     end if
7:     sample a partition R_v and store it in v
8:     for U ∈ R_v do
9:         if |S ∩ U| ≠ 0 then
10:            add BuildTree(S ∩ U) as a child of v
11:        end if
12:    end for
13:    return v
14: end function
15: function QueryTree(v, q)                      ▷ v is a tree node, q is a query point
16:    if ∃ p ∈ T_v within distance cr from q then
17:        return p
18:    end if
19:    if there is a child v' of v associated with R_v(q) then
20:        return QueryTree(v', q)
21:    else
22:        return ⊥
23:    end if
24: end function
```

Figure 14: Generic LSH Forest with pivots algorithm

To analyze LSH Forest, we need to understand two quantities: how many trees do we need to get probability of success 0.9, say, and how deep the built trees are. The latter is quite easy, while the former is much more delicate and is the main focus of this paper.

Lemma 108. *For a dataset P of size n, let $q \in X$ be a query point, and $p^* \in P$ be a near neighbor of q (that is, $d_X(q, p^*) \leq r$). Let $0 < \rho \leq 1$ be the smallest real number such that, for every subset $S \subseteq P$ where $p^* \in S$, the following holds:*

- *either q is within distance cr from one of the pivots* BuildTree *computes for S,*
- *or one has*

$$\Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)] \times \tag{6.1}$$

$$\times \mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho} \middle| \mathcal{R}(q) = \mathcal{R}(p^*)\right] \geq 1. \tag{6.2}$$

Then the tree from Algorithm 20 returns a point within distance cr for a query q with probability at least $n^{-\rho}$. Hence, if one samples $O(n^{\rho})$ trees, the entire data structure solves (c, r)-ANN for q with probability at least 0.9.

Proof. Consider any tree node we encounter answering the query q for which $p^* \in S$. We would like to prove that the probability of success in this case is at least $|S|^{-\rho}$, which, when applied to the root of the tree, gives the desired claim.

If the first bullet point from the lemma statement holds, the desired probability is equal to one. Similarly, if $|S| = 1$, the probability is also one, since the set of pivots is non-empty, and, by the assumption, $p^* \in S$. Otherwise, let us use the induction on the size of S.

We have:

$$\begin{aligned}
\Pr[\text{success}] &= \Pr_{\mathcal{R}}[\text{success for } S \cap \mathcal{R}(q)] \\
&\geq \Pr_{\mathcal{R}}[\text{success for } S \cap \mathcal{R}(q), \mathcal{R}(q) = \mathcal{R}(p^*)] \\
&= \Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)] \cdot \\
&\cdot \Pr_{\mathcal{R}}[\text{success for } S \cap \mathcal{R}(q) \mid \mathcal{R}(q) = \mathcal{R}(p^*)] \\
&\geq \Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)] \cdot \\
&\cdot \mathrm{E}_{\mathcal{R}}[|S \cap \mathcal{R}(q)|^{-\rho} \mid \mathcal{R}(q) = \mathcal{R}(p^*)] \\
&= \Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)] \cdot \\
&\cdot \mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho} \middle| \mathcal{R}(q) = \mathcal{R}(p^*)\right] \cdot \\
&\cdot |S|^{-\rho} \\
&\geq |S|^{-\rho},
\end{aligned}$$

where the fourth step is by the induction assumption (since we condition on $p^* \in S \cap \mathcal{R}(q)$) and the last step is by the lemma assumption. Technically, if $S \cap \mathcal{R}(q) = S$, then we can not quite use the induction assumption, but nevertheless, the required bound follows from the immediate claim "If for $0 < p < 1$ one has $x = px + y$, then $x \geq z$ iff $pz + y \geq z$". □

Note that one tree requires $O(nk)$ space: there are at most $O(n)$ nodes in the tree and each stores at most $k + 1$ nodes. The query time depends on the height of tree: for a maximum height h, the query time per tree is $O(h \cdot dk)$. We note that, in our applications, the height will be easy to bound. Move the preceding text before the Lemma.

6.3 LSH Forest and the bit sampling LSH

In this section we present our main algorithm claimed informally in Theorem 102. The algorithm is an instantiation of Algorithm 20 for the Hamming space. In particular, we use the Bit Sampling LSH from Definition 107 as the random partition $\mathcal{R}$ as well as a certain procedure for computing pivots, described below.

The formal statement of our main theorem is the following:

Theorem 109. *Fix $r \in \mathbb{N}, k \in \mathbb{N} \geq 2$, and $c > 2$. Then for given $P \subseteq \{0,1\}^d$ of size n, the LSH Forest with k pivots solves (c,r)-ANN with the following guarantees:*

- *The space is $O(n^{1+\rho}d + nd)$;*
- *The worst-case query time is $O(n^{\rho} \cdot d^2)$;*
- *where*

$$\rho = \frac{1}{\ln 4} \cdot \frac{1}{c} \cdot \left(1 + O\left(\frac{1}{k}\right)\right) + O\left(\frac{1}{c^2}\right).$$

We first show how to prove Theorem 109 for the case of (c,r)-Gap ANN (see Definition 104). In this case, it is enough to simply choose k pivots as described below. To handle the general case of (c,r)-ANN, we add $\Theta(\log n)$ *uniformly random pivots* to the existing set of pivots. It is not hard to show that if there are $\Omega(n)$ data points closer than cr to the query, then at least one them will end up in a random subset with very high probability, and, in that case, we are done. Otherwise, there are $o(n)$ data points within distance cr, and the analysis for the Gap ANN goes through: namely, the close points do not affect the quantity (6.2) too much. From now on, we focus on solving the (c,r)-Gap ANN problem.

6.3.0.1 Construction

The algorithm for choosing k pivots is very simple: see Algorithm 21 for the pseudo-code. In words, for a subset $S \subseteq P$ of the dataset, we sort all the points according to the distance from the *mean* of S, then we add first k points in this order which are *sufficiently diverse*, namely, we require them to be pairwise $\big((c-1)r\big)$-separated.

6.3.0.2 Intuition

Let us provide some intuition why this procedure helps.

As Lemma 108 states, we need to find the smallest ρ such that

$$\begin{aligned}&\Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)]\cdot \\ &\cdot \mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho} \middle| \mathcal{R}(q) = \mathcal{R}(p^*)\right] \geq 1.\end{aligned}$$

The first term is easy: $\Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)] \geq 1 - \frac{r}{d}$ (remember that $\mathcal{R}$ is the Bit Sampling LSH). For the sake of this discussion, let us ignore the effect of conditioning on the event "$\mathcal{R}(q) = \mathcal{R}(p^*)$" (as we will see later, the effect is provably small provided that c is large enough). Thus, we need to understand the following quantity:

$$\mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho}\right].$$

For $i \in [d]$, we denote w_i the fraction of points $p \in S$ for which $p_i \neq q_i$. Then,

$$\mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho}\right] = \mathrm{E}_{i\in[d]}\big[(1-w_i)^{-\rho}\big].$$

Thus, we need to find the smallest ρ for which

$$\left(1 - \frac{r}{d}\right)\mathrm{E}_{i\in[d]}\big[(1-w_i)^{-\rho}\big] \geq 1.$$

A naïve argument using Jensen's inequality gives:

$$\begin{aligned}\left(1-\frac{r}{d}\right)\mathrm{E}_{i\in[d]}\left[(1-w_i)^{-\rho}\right] &\geq \left(1-\frac{r}{d}\right)\mathrm{E}_{i\in[d]}\left[1-w_i\right]^{-\rho} \\ &\gtrsim \left(1-\frac{r}{d}\right)\left(1-\frac{cr}{d}\right)^{-\rho},\end{aligned}$$

where the second step follows from the fact that all except one points from S are at distance more than cr from the query point q. This merely allows us to set $\rho \approx \frac{1}{c}$, so we have not obtained any improvement compared to the classical application of the Bit Sampling LSH using Theorem 106.

Is the above simple analysis tight? The application of Jensen's inequality is (almost) tight if all w_i's are (almost) the same (around $\frac{cr}{d}$). This is indeed possible, moreover this is precisely a hard example from [174]. That is exactly where pivots come to the rescue! By using the fact that the distance from q to all the pivots computed by Algorithm 21 is more than cr, we are able to show that w_i *can not be* almost the same, and thus the application of Jensen can be sharpened.

For example, consider the case of $k = 1$ pivot. W.l.o.g. we can assume that $p^* = 0^d$. If p^* was a pivot, we would be done, so w.l.o.g. we can assume that the pivot is $\widetilde{p} \neq p^*$. Since $\|\widetilde{p} - p^*\|_1 > (c-1)r \approx cr$ (q is at least cr apart from $\widetilde{p}$), we can assume w.l.o.g. that $\widetilde{p} = 1^{cr}0^{d-cr}$. Now comes the crucial insight: since $\widetilde{p}$ is closer to the mean of S than p^*, we have that, at least on the first cr coordinates, weights w_i must be at least $1/2$ on average, which is much larger than the average $\frac{cr}{d}$. This allows us to claim that the ℓ_1-mass of w_i's is somewhat concentrated on the first cr coordinates, and hence the application of Jensen can be significantly improved.

Introducing more pivots allows us to obtain *even better* concentration of the weights w_i. More precisely, for k pivots we can get as many as $\big(2 - O(1/k)\big)cr$

bits of substantial average weight. Thus, if k is large, almost all the ℓ_1-mass of w_i's is concentrated on $\approx 2cr$ coordinates, which corresponds to a *random instance*, for which the Bit Sampling LSH yields $\rho \approx \frac{0.73}{c}$. We obtain an increased number of large weights by noticing that we have k pivots which are at distance $\approx cr$ from each other, thus there must be a large number of coordinates where at least one pivot is equal to one.

Algorithm 21 Computing pivots for the Hamming space

```
function ComputePivots(v, k)    ▷ v is a tree node, k is the desired number of pivots
    S ← data points corresponding to v
    T_v ← ∅
    p^c ← the mean of S
    for p ∈ S in the order of increasing ℓ_1 distance from p^c do
        if the ℓ_1-distance between p to T_v is at least (c − 1)r then
            T_v ← T_v ∪ {p}
            if |T_v| = k then
                return T_v
            end if
        end if
    end for
    return T_v
end function
```

Figure 15: Computing pivots for the Hamming space

6.3.0.3 Analysis

To prove Theorem 109, we will use the lemma for the generic LSH Forest, Lemma 108. Let us fix some query q and assume that $p^* \in P$ is its near neighbor ($\|q - p^*\|_1 \leq r$). We need to establish (6.2) assuming that q is far from all the pivots.

Let us first show the following combinatorial lemmas which will be useful later.

Lemma 110. *Fix $s, n \geq 1$, $k \geq 2$ and small $\epsilon > 0$. For $i \in [n]$, let $a_i \in [1, k]$ and $x_i \in [0, 1]$. Suppose we have that:*

- $\sum_i a_i x_i \geq \frac{1}{2} \sum_i a_i \, (1 - \epsilon)$, *and*
- $\sum a_i(k - a_i) \geq \frac{k(k-1)\cdot s}{2}$.

Then we have that:

$$\prod_i (1 - x_i) \leq \exp\Big(s \cdot (1 - 1/k) \cdot (1 - \ln(4)) - \sum_i x_i + O(\epsilon s) \Big).$$

Proof appears in Appendix 6.4

Lemma 111. *Let $x \in [0, 1]^d$, $s \in \mathbb{N}$, and let $\mathcal{U}$ be a family of k subsets of coordinates $[d]$ such that:*

- *For each $U \in \mathcal{U}$, $|U| \geq s$*
- *For each $U \in \mathcal{U}$ one has $\sum_{i \in U} x_i \geq \frac{|U|}{2} \cdot \big(1 - O(\frac{1}{c})\big)$*
- *For every distinct $U_1, U_2 \in \mathcal{U}$ one has $|U_1 \triangle U_2| \geq s$.*
- $\sum_{i \in [d]} x_i \geq s$.

Then:

$$\prod_i (1 - x_i) \leq \exp\left(-s \cdot \frac{\ln(4)k + 1}{k+1} + O(s/c)\right)$$

.

Proof. For $i \in [d]$, let $a_i \in \{0, 1, .., k\}$ denote the number of subsets in $\mathcal{U}$ containing coordinate i. The above constraints imply that:

- $\sum_i a_i \geq ks$.
- $\sum_i a_i x_i \geq \frac{1}{2} \sum_i a_i \cdot \left(1 - O(\frac{1}{c})\right)$,
- $\sum_i a_i(k - a_i) \geq \frac{k(k-1)s}{2}$.

The last constraint follows from the double counting for the sum $\sum_{\substack{U,U' \in \mathcal{U} \\ U \neq U'}} |U \triangle U'|$, which is, on the one hand, is at least $\binom{k}{2}s$ by the statement of the lemma, and, on the other hand is equal to $\sum_i a_i(k - a_i)$, which follows from the decomposition of the sum over coordinates.

Summing up the first and last constraints gives:

$$\sum_i a_i(k + 1 - a_i) \geq \frac{k(k+1)s}{2}.$$

Let $S = \bigcup_{U \in \mathcal{U}} U$. We decompose objective function as follows:

$$\prod_{i \in [d]} (1 - x_i) = \prod_{i \in S} (1 - x_i) \cdot \prod_{i \in [d] \setminus S} (1 - x_i)$$

We now invoke Lemma 110 over the first part:

$$\prod_{i\in S}(1-x_i) \leq \exp\Big(s\cdot(1-1/(k+1))\cdot(1-\ln(4)) \\ -\sum_{i\in S}x_i + O(s/c)\Big).$$

The second part can be simplified to:

$$\prod_{i\in[d]\setminus S}(1-x_i) \leq \prod_{i\in[d]\setminus S} e^{-x_i} \leq \exp\left(\sum_{i\in S}(x_i) - s\right).$$

The last inequality follows the condition that $\sum_{i\in[d]} x_i \geq s$. Finally, we multiply the 2 parts together and get:

$$\begin{aligned}\prod_{i\in[d]}(1-x_i) &\leq \exp\left(s\cdot(1-1/(k+1))\cdot(1-\ln(4))\right. \\ &\quad \left. - s + O(s/c)\right) \\ &\leq \exp\left(-s\cdot\left(\ln(4)\cdot(1-\frac{1}{k+1}) + \right.\right. \\ &\quad \left.\left.\frac{1}{k+1}\right) + O(s/c)\right) \\ &= \exp\left(-s\cdot\frac{\ln(4)k+1}{k+1} + O(s/c)\right)\end{aligned}$$

as needed.

□

We use the above lemmas to calculate the probability of success for each hash function as follows:

Lemma 112. *Let $P \subset \{0,1\}^d$ be a set of n points, $q \in \{0,1\}^d$ be a selected point and p^* be the near neighbor. The probability to find $p \in P$ such that $\|p-q\|_1 \leq cr$ in a tree is at least $n^{-\rho}$, where $\rho \leq \frac{1}{\ln(4)\cdot(c-2)}\cdot(1+\frac{2}{7k+5}) + O(\frac{1}{c^2})$.*

Proof. As we have shown earlier in Lemma 108, all we need to prove is that for any $S \subseteq P$ with $p^* \in S$ such that q is further than cr from all the pivots, one has:

$$\Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)] \cdot$$
$$\cdot \mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho} \middle| \mathcal{R}(q) = \mathcal{R}(p^*)\right] \geq 1.$$

For $i \in [d]$, let w_i denote the fraction of $p \in S$ for which $p_i \neq p_i^*$, and let p^c denote the mean of S. Since we partition using the Bit Sampling LSH, it is easy to see that:

$$\frac{|S \cap \mathcal{R}_i(p^*)|}{|S|} = 1 - w_i,$$

where $\mathcal{R}_i$ is the partition corresponding to splitting the space according to the bit i.

If there exists a close enough pivot point, we are done. Otherwise, there exists a set of k points $\{p^1, .., p^k\} \subseteq S$ s.t. :

- For each $i \in [k]$: $\|p^i - p^c\|_1 \leq \|p^* - p^c\|_1$;
- For every distinct $i, j \in [k]$: $\|p^i - p^j\|_1 \geq (c-1)r$.

For each $i \in [k]$, let $U_i \subseteq [d]$ denote the set of coordinates in which p^i differ from p^*. Since the distance of both p^*, p^i and p^c are equal on all coordinates outside U_i, and since for each $j \in U_i$, $p_j^* - p_j^c = w_j$ and $p_j^i - p_j^c = 1 - w_j$, then the first constraint gives:

- For each U_i: $\sum_{j \in U_i} w_j \geq \sum_{j \in U_i} 1 - w_j \Rightarrow \sum_{j \in U_i} w_j \geq |U_i|/2$;

The second constraint gives:

- For every distinct U_i, U_j : $|U_i \triangle U_j| \geq (c-1)r$.

While the above conditions are enough to invoke Lemma 111, we need to condition on the fact that q and p^* collide. Let $D \subseteq [d]$ denote the set of coordinates for which $p^*_i = q_i$, we deduce similar properties when restricted to the set D:

- For each U_i: $|U_i \cap D| \geq (c-1)r$ since $\|p^i - q\|_1 \geq cr$;
- For each U_i: $\sum_{j \in U_i \cap D} w_j \geq \frac{|U_i \cap D| - r}{2} = \frac{|U_i \cap D|}{2} \cdot \left(1 - O(\frac{1}{c})\right)$ since $\|p^* - q\|_1 \leq r$ and hence $|U_i \setminus D| \leq r$;
- For every distinct $U_i, U_j : |(U_i \triangle U_j) \cap D| \geq (c-1)r - r = (c-2)r$.

We can finally apply Lemma 111 with vector $\vec{x} = \vec{w}$ over bits in set D and $s = (c-2)r$, obtaining that:

$$\prod_{i \in D}(1 - w_i) \leq \exp\left(-(c-2)r \cdot \frac{\ln(4)k + 1}{k+1} + O(r)\right)$$

Since we are sampling uniformly then:

$$\begin{aligned}
&\mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho} \middle| \mathcal{R}(q) = \mathcal{R}(p^*)\right] \\
&= \mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(p^*)|}{|S|}\right)^{-\rho} \middle| \mathcal{R}(q) = \mathcal{R}(p^*)\right] \\
&= \mathbb{E}_{i \in D}\left[(1 - w_i)^{-\rho}\right] \\
&\geq \left(\prod_{i \in D}(1 - w_i)^{-\rho}\right)^{1/|D|} \\
&= \left(\prod_{i \in D}(1 - w_i)\right)^{-\rho/|D|} \\
&\geq \exp\left(\frac{(c-2)r \cdot \rho \cdot (\ln(4)k + 1)}{|D| \cdot (k+1)} - O(\frac{\rho \cdot r}{|D|})\right)
\end{aligned}$$

where the third step is by the AM–GM inequality. Thus we can set:

$$\rho = \frac{k+1}{(\ln(4)k+1)\cdot(c-2)} + O(\tfrac{1}{c^2}) < \frac{1}{\ln(4)\cdot(c-2)} \cdot (1 + \tfrac{2}{7k+5}) + O(\tfrac{1}{c^2})$$

(as $1/\ln(4) > 5/7$). For which we obtain:

$$\mathrm{E}_{\mathcal{R}}\left[\left(\frac{|S \cap \mathcal{R}(q)|}{|S|}\right)^{-\rho} \middle| \mathcal{R}(q) = \mathcal{R}(p^*)\right] \geq e^{r/|D|}.$$

The probability that p^* and q collide is:

$$\Pr_{\mathcal{R}}[\mathcal{R}(q) = \mathcal{R}(p^*)] \geq \frac{|D|}{|D|+r} \geq e^{-r/|D|}.$$

We have thus established that (6.2). □

We are now ready to complete the proof to our main theorem:

Proof of Theorem 109. The proof follows immediately from Lemma 112. In particular, we prepare $O(n^\rho)$ trees using the generic LSH Forest algorithm, Algorithm 20 with the pivot set, of size k, chosen as in Algorithm 21. □

6.4 Proof of Lemma 110

We prove the following lemma.

Lemma 113. *Fix $s, n \geq 1$, $k \geq 2$ and small $\epsilon > 0$. For $i \in [n]$, let $a_i \in [1, k]$ and $x_i \in [0, 1]$. Suppose we have that:*

- $\sum_i a_i x_i \geq \frac{1}{2} \sum_i a_i \, (1 - \epsilon)$, *and*
- $\sum a_i(k - a_i) \geq \frac{k(k-1)\cdot s}{2}$.

Then we have that:

$$\prod_i (1 - x_i) \leq \exp\left(s(1 - \tfrac{1}{k})(1 - \ln 4) - \sum_i x_i + O(\epsilon s) \right).$$

Proof. First we make a change of variables: $y_i = 1 - x_i \in [0, 1], t = \frac{1+\epsilon}{2}$ and $f_i = a_i/k \in [1/k, 1]$.

Now our problem can be reformulated as maximizing $F(y) = \prod_{i \in [n]} (y_i \cdot \exp(1 - y_i))$, subject to the following constraints:

- $\sum_i f_i(y_i - t) \leq 0$;
- $\sum f_i(1 - f_i) \geq \frac{(1-1/k)\cdot s}{2}$.

Note that, there is an optimum solution where both are in fact equalities. In particular, to see this for the second part, note that for i such that $y_i < t$, we can assume $f_i > 1/2$ (otherwise take $f_i := 1 - f_i$) and increase f_i. Similarly, for $y_i > t$, we can assume $f_i < 1/2$ and we decrease it further.

We now analyze the Lagrangian of the logarithm of F. In particular for Lagrangian multipliers $\lambda, \eta \in \Re$, taking the derivatives with respect to y_i gives us that for all i:

$$1/y_i - 1 = \eta f_i, \tag{6.3}$$

and with respect f_i, for all i:

$$0 = \lambda(1 - 2f_i) + \eta(y_i - t). \tag{6.4}$$

Plugging-in one equation into the other, we get an equation which is polynomial of degree 2 in y_i (or, alternatively in f_i), and coefficients that depend on the Lagrange multipliers λ, η. Hence, the optimal solution can have only at most 2 distinct values for (y_i, f_i). Let these values be (y_1, f_1) and (y_2, f_2).

Assume w.l.o.g $y_1 \leq y_2$, and suppose there are $\alpha_1 s(1-1/k)$ pairs (y_1, f_1) and $\alpha_2 s(1-1/k)$ pairs (y_2, f_2), for $\alpha_1, \alpha_2 \geq 0$. Then, the first equation becomes:

$$\alpha_1 f_1(t - y_1) = \alpha_2 f_2(y_2 - t) \tag{6.5}$$

The second equation becomes $\alpha_1 s(1-1/k) \cdot f_1(1-f_1) + \alpha_2 s(1-1/k) \cdot f_2(1-f_2) = \frac{(1-1/k)s}{2}$, which simplifies to:

$$\alpha_1 f_1(1 - f_1) + \alpha_2 \cdot f_2(1 - f_2) = \tfrac{1}{2}, \tag{6.6}$$

The objective $\log F$ can be rewritten as maximizing:

$$\left[\alpha_1(\ln(y_1) + 1 - y_1) + \alpha_2(\ln(y_2) + 1 - y_2)\right](1 - 1/k)s$$

It is enough to maximize $G = \alpha_1(\ln(y_1) + 1 - y_1) + \alpha_2(\ln(y_2) + 1 - y_2)$, subject to the above constraints.

Consider the Langrangian L_G of G, its derivative with respect to each α_i gives:

$$\frac{\partial L_G}{\partial \alpha_i} = \ln(y_i) + 1 - y_i - \eta f_i(y_i - t) - \lambda f_i(1 - f_i)$$

Note that from Eqn. (6.3) and (6.4) (which hold true when considering L_G as well):

$$\begin{aligned}
\eta f_i &= 1/y_i - 1 \\
\lambda f_i &= \frac{\eta f_i(y_i - t)}{(2f_i - 1)} \\
&= \frac{(1/y_i - 1)(y_i - t)}{(2f_i - 1)}
\end{aligned}$$

Therefore, we have:

$$\begin{aligned}\frac{\partial L_G}{\partial \alpha_i} &= \ln(y_i) + 1 - y_i - (\tfrac{1}{y_i} - 1)(y_i - t) - \frac{(\frac{1}{y_i} - 1)(y_i - t)(1 - f_i)}{(2f_i - 1)} \\ &= \ln(y_i) + 1 - y_i + \left(\frac{f_i - 1}{2f_i - 1} - 1\right)(1/y_i - 1)(y_i - t) \\ &= \ln(y_i) + 1 - y_i - \frac{f_i(1/y_i - 1)(y_i - t)}{2f_i - 1} \\ &= \ln(y_i) + 1 - y_i + \frac{\lambda}{\eta^2}(y_i^{-1} - 1)^2\end{aligned}$$

The last step follows from exchanging variables according to Eqn. (6.3) and (6.4). At the optimum of G, for each α_i, either $\frac{\partial L_G}{\partial \alpha_i} = 0$ or α_i cannot be decreased or increased which can only happen at the boundaries, i.e., where either $y_1 = 0$ or $t, y_2 = 1$ or t (otherwise we can always perturb y_i and increase/decrease α_i).
One can observe that $\frac{\partial L_G}{\partial \alpha_i} = 0$ where $y_i = 1$. We can further analyze $\frac{\partial L_G}{\partial \alpha_i}$ by taking a second derivative with respect to y_i:

$$\begin{aligned}\frac{\partial^2 L_G}{\partial \alpha_i \partial y_i} &= y_i^{-1} - 1 - 2\lambda\eta^{-2}y^{-2}(y^{-1} - 1) \\ &= (1 - 2\lambda\eta^{-2}y^{-2})(y^{-1} - 1)\end{aligned}$$

Since there can be at most 1 assignment of $0 < y_i < 1$ for which $\frac{\partial^2 L_G}{\partial \alpha_i \partial y_i} = 0$, and since $\frac{\partial L_G}{\partial \alpha_i}$ is monotonic elsewhere in the range and equals 0 where $y_i = 1$, then there can be at most 1 assignment of $0 < y_i < 1$ for which $\frac{\partial L_G}{\partial \alpha_i} = 0$, which means that any optimal solution we cannot have distinct y_1, y_2 which both lie in the interior, and therefore we must have at least of the following: $y_1 = 0, y_1 = t, y_2 = 1$ or $y_2 = t$.

Now, let us analyze the above cases separately:

- If $y_1 = 0$, the function is at minimum.

- If either $y_i = t$, then according to Eqn. (6.5) the other y_j must also be t (or otherwise $f_j = 0$ for which in optimum $\alpha_j = 0$).
- If $y_2 = 1$, then according to Eqn. (6.3): $1 = \eta f_2 + 1$ and therefore $f_2 = 0$, and according to Eqn. (6.5): $y_1 = t$ (and since $f_2 = 0$, α_2 is irrelevant).

In either of the last 2 scenarios, it is enough to optimize:

$$F(\alpha) = (t^{\alpha} e^{(1-t)\alpha})^{(1-1/k)s} = \exp\left(\alpha \cdot s \cdot (1 - 1/k) \cdot (1 - t - \ln(1/t))\right)$$

subject to:

- $\alpha f(1 - f) = \frac{1}{2}$.

The above is optimized at $f = 1/2; \alpha = 2$, and hence the optimal value of F is:

$$F^* = \exp\left(s \cdot (1 - 1/k) \cdot 2(1 - t - \ln(1/t))\right)$$

Therefore, the required inequality holds:

$$\prod_i (1 - x_i) \leq \exp\left(s \cdot (1 - 1/k) \cdot (1 - \ln(4)) - \sum_i x_i + O(\epsilon s)\right).$$

□

Chapter 7

Average Similarity Embeddings

Bi-Lipschitz metric embeddings (see Definition 7) are frequently used to reduce ANN under some fixed "hard" metric to ANN under, say, ℓ_1/ℓ_2, or ℓ_∞, for which ANN algorithms are well understood.

Such embeddings have been studied for a number of metrics including edit distance, Ulam distance, Earth-Mover Distance (Wasserstein-1 distance), ℓ_p-product spaces, general symmetric norms, Fréchet distance, and others. Alas, for many of these metrics, the best possible distortion remains a long-standing open question. A notorious example is edit (Levenshtein) distance over length-k strings: early results showed that the best distortion c_1 for embedding into ℓ_1 is $\Omega(\log k) \leq c_1 \leq 2^{O(\sqrt{\log k \log\log k})}$ [175; 134; 144] — improving over these bounds is a 20 years old open question [153].

More recently, researchers discovered that a weaker form of embeddings, namely *average* embeddings, yield ANN algorithms as well. Very recently, [38] extended the above notion to average sketching, which should be seen as a generalization of embedding into ℓ_1. The authors obtained a new trade-off between space and approximation

for ℓ_p.

In this chapter, we develop the notion of metric embeddings into *similarity measures* over $\mathbb{R}^m_+$, such as the weighted Jaccard coefficient. We show *average* embeddings into such similarity measures for a number of metric spaces, with (appropriately defined) distortion that is smaller than the best possible distortion of embedding into ℓ_1 or ℓ_2 spaces (bi-Lipschitz or average).

We complement our new embeddings with a new algorithm for Approximate Nearest Neighbor Search (ANN) that leverages such an embedding in a black box fashion. Combining these results, we obtain new efficient algorithms for ANN under the following two classic metrics, achieving an exponential improvement to longstanding prior work:

- Edit distance over length-k strings: $\mathrm{poly}(\log k)$ approximation;
- ℓ_p over $\mathbb{R}^d$, for $p > 2$: $O(\log p)$ approximation (known to be asymptotically optimal in relevant models of computation).

The main embeddings are UASE (see Definition 9, and complete version Definition 115) and WASE (see Definition 10, and complete version Definition 116).

We primarily consider the target similarity measure $\mathcal{S}(x, y)$ being the *Sørensen-Dice coefficient*, denoted by $M(x, y) \triangleq \frac{2\sum_i \min\{x_i, y_i\}}{\sum_i x_i + y_i}$.

One of our main contributions is to show that a WASE embedding yields efficient ANN algorithms (for similarity measure $\mathcal{S} = M$), as stated informally in Chapter 1 (see Theorem 11) and will be formalized later in this chapter (see Theorem 148).

Combining the above, we obtain concrete results on WASE embeddings for the edit distance and ℓ_p, proving Theorems 12 and 13 (from Chapter 1).

Organization The rest of the paper has the following order. We first present a formal definition of our similarity embeddings, together with a technical overview of the results (Section 7.2). Then we present our embedding for the ℓ_p, and hence ℓ_∞, norms (Section 7.5). Next, we present an embedding for a variant of the Earth Mover Distance, which is used for the edit distance (Section 7.6). Next, we present the embedding for the edit distance (Section 7.7). We present our constructions for the ANN data structures, based on the existence of embeddings into similarity measures (Section 7.8). Finally, we put together the embeddings and ANN constructions to obtain our main theorems in Section 7.8.4.

7.1 Discussion of Definitions and Techniques

In this section, we discuss the notion of similarity embeddings and present the main technical ideas. While there are a number of new technical ideas, we start by highlighting three ideas that lay the foundations of embedding into similarity measures: 1) defining the right notion of similarity embedding, 2) showing that such an embedding exists for ℓ_∞^d with distortion $O(\log \log d)$, and 3) identifying a certain dense/sparse duality for datasets. We will then discuss designing an embedding for the edit distance specifically.

7.1.1 Embedding into similarity measures

The new type of embedding is motivated directly by the ANN application, in particular that all (high-dimensional) c-approximate ANN are based on (derivatives of) Locality Sensitive Hashing (LSH). Namely, for a fixed scale $r > 0$, a randomized hash function

$h : \mathcal{X} \to U$ is (r, cr, ρ)-LSH if probability of collision of points at distance τ is function $p(\tau)$, and $\rho = \frac{\log p(r)}{\log p(cr)} < 1$. One can then obtain c-approximate ANN algorithms with runtime $\sim n^{\rho}$ [114]. For example, for a runtime of $\leq n^{0.1}$, it must be that $p(r) \geq p(cr)^{0.1}$.

An embedding of a metric $\mathcal{X}$ into, say, ℓ_1 yields an LSH scheme for $\mathcal{X}$ (using LSH for ℓ_1). Notably, if the distortion of the embedding is c_1, then one can obtain $(r, O(c_1 r), 0.1)$-LSH with $p(r) = 1/2$. Conversely, any $(r, cr, 0.1)$-LSH scheme with $p(r) = 1/2$ for $\mathcal{X}$ implies a (weaker type of) embedding into ℓ_1 that preserves the distances at the scale r up to factor c (see [74]). In other words, LSH where $p(r) = \Omega(1)$ is essentially equivalent to an embedding into ℓ_1!

Alas, many spaces of interest require large distortion c_1 to embed into ℓ_1, and thus, to avoid this non-embeddability barrier, the only remaining option is to have LSH with $p(r) \ll 1/2$ — which is precisely what our new type of embedding aims to do. Instead of talking about probability of collision of a random hash function, it is more natural to talk about a deterministic object. Indeed, we can always transform an LSH function h into a map $\varphi : \mathcal{X} \to \{0,1\}^m$ such that $\Pr_h[h(p) = h(q)] = M(\varphi(x), \varphi(y))$, where the M is, say, the Sørensen-Dice coefficient (similarity measure): $M = 2\frac{\sum_i \min\{\varphi(p)_i, \varphi(q)_i\}}{\|\varphi(p)\|_1 + \|\varphi(q)\|_1}$. Since we want to get $\rho \leq 0.1$, we want that, for any pair (q, p) at distance $\leq r$ and pair (q', p') at distance $\geq cr$ we have

$$0.1 \geq \rho = \frac{\log 1/p_1}{\log 1/p_2} = \frac{\log 1/M(\varphi(q),\varphi(p))}{\log 1/M(\varphi(q'),\varphi(p'))}.$$

Now, we can think of the target space as $\{0,1\}^d$ with "distance" $\mathsf{d}_M = \log 1/M(\varphi(q), \varphi(p))$. We can consider various notions of embedding, such as the natural bi-Lipshitz embedding into $(\{0,1\}^d, \mathsf{d}_M)$, which corresponds to "data-independent" hashing algo-

rithms. We do not know if there are interesting examples of such bi-Lipshitz embeddings. Indeed, all known LSH schemes with $p(r) \ll 1/2$ are data-dependent [120; 27]. This leads us to consider the "average embedding" version, which naturally corresponds to data-dependent hashing (as the hash function h, and hence φ depend on the dataset) — which gives us UASE from Definition 9:

$$\frac{\log 1/M(\varphi(p),\varphi(q))}{\log \mathbb{E}_{p',p''\in P}[1/M(\varphi(p'),\varphi(p''))]} \leq c \cdot \frac{\mathsf{d}(p,q)}{\mathbb{E}_{p',p''\in P}[\mathsf{d}(p',p'')]}.$$

Alas, we don't know whether this definition suffices for obtaining the desired distortion bounds, such as $O(\log\log d)$ for ℓ_∞^d — requiring us to consider a more involved notion. We note that the "right" definition of such an embedding has to balance two criteria:

- The embedding notion should allow for improved bounds on distortion: e.g., obtain $O(\log\log d)$ for ℓ_∞^d.
- We should still be able to use the embedding to obtain ANN data structures. While we showed how LSH leads to the first definition of similarity embedding, it is not immediately clear that the inverse direction holds, especially for a more involved notion.

We manage to balance these two criteria by introducing the *Weighted Average Similarity Embedding* (Def. 10), which can be seen as a convex combination (ℓ_1, outer product) of UASE embeddings.

We note that the considered "distance" d_M is does not satisfy triangle inequality[1] and hence poses additional challenges in building ANN using average similarity em-

[1] As mentioned above, d_M can be $O(1)$-embedded into the Bhattacharyya distance, but the latter does not satisfy the triangle inequality either.

beddings (for WASE or UASE). Namely, the ANN data structure needs an upper bound on the expected number of points from a dataset P that collide with a given query q, corresponding to lower bounding the quantity $\log 1/\mathbb{E}_{p\in P}[M(\varphi(q),\varphi(p))]$. Intuitively, this corresponds to lower bounding the average "M-distance" from q to $p \in P$. In contrast, an average embedding gives a bound on average distance between points $p', p'' \in P$. This difference appears already for the average embeddings into ℓ_1 but can be dealt with easily using the triangle inequality: $\mathbb{E}_p[\mathsf{d}(q,p)] \geq \frac{1}{2}\mathbb{E}_{p',p''}[\mathsf{d}(p',p'')]$. We show that a similar triangle inequality for M-distance holds as well, after a suitable padding of the embedding.

The actual ANN algorithm for WASE is based on the LSH Forest [44], whose theoretical analysis is given in [33].

7.1.2 Embedding of ℓ_∞ into similarity measures

Our first proof of concept result is to show that d-dimensional ℓ_∞ admits an average similarity embedding (WASE) with distortion $O(\log\log d)$, and more generally ℓ_p with distortion $O(\log p)$. This matches the celebrated result of [120] who designed an ANN data structure with the same approximation. Despite lots of subsequent work including showing that the approximation is optimal in a number of computational models [12; 129; 179; 57], the original algorithm remains somewhat poorly understood. Indeed, 1) there has been no alternative algorithm achieving the same approximation, and 2) the natural extension to ℓ_p^d, for $p > 2$, with $O(\log p)$ approximation ANN, has eluded the researchers until now (see best current results in [38; 145; 27; 167; 40]).

As the best average embedding of ℓ_∞ into ℓ_1 is $\Omega(\log d)$ [165], our embedding of

ℓ_∞ shows that WASE is provably stronger than average embeddings into ℓ_1. Indeed, the ℓ_∞ algorithm of [120] is based on certain partitions that intuitively correspond to $p(cr)$ being as small as $1/d$.

The advantage of a WASE embedding for ℓ_∞ is also that it can be used further down the line, i.e., in combination with other embeddings.

Our embedding for ℓ_p proceeds as follows at a high level. Consider a pointset $P \in \mathbb{R}^d$. We embed each coordinate $e \in [d]$ of the dataset P separately. For this we devise a data-dependent scalar embedding $\phi_e : \mathbb{R}^d \to [2^{-p}, \infty)$ (that also depends on p). Then each point q is mapped to $(\phi_e(q_e))_{e\in[d]}$ thought of as WASE with outer product of dimension d (i.e., "inner dimension" is only 1).

Each 1-coordinate embedding ϕ_e is embedded as a function of the e^{th} coordinate of the dataset, denoted by the vector $x \in \mathbb{R}^n$. At a high-level, construction of ϕ can be thought of as a certain combination ANN construction from [120] as well as the Mazur map of ℓ_p into ℓ_2 [167; 40]. In particular, [120] showed that, for r the ANN closeness parameter, unless the dataset is "dense" on that coordinate (most points are in an interval of length $O(r \log\log d)$), there exists two overlapping intervals $(-\infty, m + r]$ and $[m, \infty)$ for some m, with (roughly) the property that $\mu_x([m, \infty)) \leq \mu_x([-\infty, m + r))^2$ where μ_x is just a measure that count the number of coordinates in the interval. We note that it is far from clear how [120]'s method can be applied — the embedding does not even have a parameter r to start with!

Nevertheless, for our embedding, we find a well-chosen interval I_e with the following special properties. First of all, we "project" entries of x to I_e: namely, $y = \text{proj}_{I_e}(x) = \mu([-\infty, x] \cap I)$ (under the uniform measure μ). Then I_e is such that 1) vector y is sparse (few non-zeros), and 2) for some $q_e < p$ the q_e-norm of y is

a sufficiently large fraction of the p-norm of y (and of x), which intuitively promotes a lower bound on the sparsity of y. On the resulting vector y we now apply a version of the Mazur map to obtain the final embedding ϕ_e. Namely, up to scaling and other caveats, $\phi_e(x) = (\text{proj}_{I_e}(x) + 0.5)^{q_e}$.

7.1.3 Dense/sparse dataset duality

While we managed to design a direct WASE for ℓ_∞, it would be convenient to have extra tools for designing WASE for other metrics. We introduce one such tool, based on a certain dense/sparse dataset duality.

In particular, consider a dataset $P \subset \mathcal{X}$ and let the average distance be $\hat{\mathsf{d}} = \mathbb{E}_{p',p'' \in P} \mathsf{d}(p', p'')$. Suppose the dataset is what we call "dense": at least $\Omega(n)$ points have at least δn points at distance $\leq \hat{\mathsf{d}}/6$. Then we claim that we can directly design a WASE for the point-set P, with approximation $O(\log \log 1/\delta)$! In particular, in this case, we use a version of the classic Fréchet embedding into ℓ_∞ [152], and then use our WASE for ℓ_∞. Specifically, we sample a set $S \subset P$ at random, of size $m = O(1/\delta)$. Then, the embedding $\psi : \mathcal{X} \to [0, \infty]^m$ is simply $\psi(p) = \{\mathsf{d}(p, s)\}_{s \in S}$. The embedding is an average embedding into ℓ_∞ with $O(1)$ distortion, as shown in the following claim.

Claim 114 ([152])**.** *Consider set P points in metric $(\mathcal{X}, \mathsf{d})$. Let $\hat{\mathsf{d}} = \mathbb{E}_{p',p'' \in P} d(p', p'')$ and suppose, for $\epsilon, \delta > 0$, at least ϵn points have at least δn points at distance $< \hat{d}/6$. Then the above Fréchet embedding $\psi : P \to \mathbb{R}^m$ for $m = O(1/\delta)$ satisfies:*

$$\mathbb{E}_{p',p'' \in P} \|\psi(p') - \psi(p'')\|_\infty \geq \Omega_\epsilon(\hat{\mathsf{d}}).$$

As a result, we primarily need to consider datasets that satisfy the complement "sparse" property: most pairs of points are at distance $\Omega(\hat{\mathsf{d}})$. Note that this means

that the metric is somewhat uniform (from the lower bound perspective). The latter property can be leveraged to build embeddings (now, in way that more explicitly uses the properties of the metric d).

7.1.4 Embedding for the edit distance

The WASE embedding for the edit distance is our most technically involved part, and leverages the WASE for ℓ_∞. In fact, our embedding for the edit distance is composed of two parts. The first part is a WASE embedding for yet another classic metric, the Earth-Mover Distance (EMD) over some metric $(\mathcal{X}, \mathsf{d})$ (under certain constraints), assuming that $(\mathcal{X}, \mathsf{d})$ itself admits a WASE. The second part is an average embedding from the edit distance to EMD over the ℓ_∞ metric, for which we use the WASE construction from above.

Earth-Mover Distance (EMD, Section 7.6). $\mathsf{EMD}_{\mathcal{X}}$ is defined for two k-sized sets $A, B \subset \mathcal{X}$ as $\mathsf{EMD}(A, B) = \min_{\pi: A \to B} \sum_{a \in A} \mathsf{d}(a, \pi(a))$ where π ranges over all permutations. Embeddings for EMD have been studied extensively for low- and high-dimensional spaces $\mathcal{X}$ [126; 134; 168; 14] (and many more results exist for related problems of fast evaluation, sketching, ANN; see, e.g., the references in the recent [127]). Overall, for high-dimensional spaces, $\mathsf{EMD}_{\mathcal{X}}$ when $\mathcal{X}$ is d-dimensional ℓ_1 has an ℓ_1 distortion of $\approx O(\log k \log d)$; recently, [127] also showed that, for ANN, one can improve the approximation to $\tilde{O}(\log k)$ via a data-dependent hashing algorithm. However, when $\mathcal{X}$ is ℓ_∞ — which is what we are interested in — the best distortion is still $\Omega(\sqrt{d})$.

One tempting avenue for developing embeddings for EMD under ℓ_∞ is to embed ℓ_∞ into ℓ_1 and then use the result for EMD over ℓ_1. While a bi-Lipschitz embedding

requires $\Omega(\sqrt{d})$ distortion, there exists an *average embedding* of ℓ_∞ to ℓ_1 with $O(\log d)$ distortion [165]. Alas, an average embedding of $\mathcal{X}$ is not sufficient for EMD over $\mathcal{X}$. This is because, say, for two sets $A, B \subset \mathcal{X}$ it is not enough to preserve the average distance $\mathsf{d}(a, b)$ over k^2 pairs (a, b) — we need to make sure that the best matching between A and B doesn't drop in value. We do not see a method to rectify this issue.

Instead, we develop WASE for $\mathsf{EMD}_{\mathcal{X}}$ directly, assuming 1) $\mathcal{X}$ admits a good WASE (which we have for ℓ_∞ from above) , and 2) the dataset is "sparse". Let us expand on the latter condition by introducing the following notation. For $\delta > 0$ and a set $S \subset \mathcal{X}$ as well as $q \in \mathcal{X}$, let $D_\delta(q, S)$ be the δ-quantile distance from q to S (i.e., there are $\delta|S|$ points at distance at most $D_\delta(q, S)$). Furthermore, we define $D_\delta(S)$ be the average of $D_\delta(q, S)$ over $q \in S$.

Now, suppose the dataset is $P \subset \binom{\mathcal{X}}{k}$ of size n. Let $S \subset \mathcal{X}$ be the union of all sets in P. We say the dataset is sparse if $D_{1/k^2}(S) \geq t$ for some threshold, where the final WASE distortion will depend on t (and average EMD inside P). The intuition why this is helpful is as follows: for two typical sets $A, B \in P$ almost all their inter-distances are $\geq t$. At the end of the day, the embedding proceeds as follows. We first leverage WASE for $\mathcal{X}$ to obtain data-dependent hashing (DDH) for $\mathcal{X}$, which we then use to obtain UASE for $\mathcal{X}$ for a fixed sparsity scale Δ. Then, the embedding of a set $A \subset \mathcal{X}$ is roughly the *sum* of the latter UASE embeddings of $a \in A$ (and concatenated over all sparsity levels Δ).

From edit distance to EMD (Section 7.7). As many modern constructions for edit distance, our embedding of edit distance is via a recursive decomposition of a string into substrings. In particular, for a string p, $i \in [n]$, we consider the following tree T_i: the root is the entire string p, and then for each node with an associated

substring $p_{i..i+w-1}$, the children have associated substrings $p_{i..i+w/2-1}$ and $p_{i+w/2..i+w-1}$. Fundamentally, we use the following property of the edit distance, for any length w. For strings $x, y \in \Sigma^k$, let X_w, Y_w respectively be all $\approx k$ substrings of length w (at all starting positions). Then $\mathsf{EMD}(X_w, Y_w) \leq O(w \cdot \mathtt{ed}(x, y))$. Alas, this is only an upper bound and the above EMD distance may drop a lot (e.g., think $w = 1$).

Given a dataset P of strings, the embedding proceeds as follows. First of all, if $D_\delta(P)$ is only a small fraction of the average distance $\widehat{\mathtt{ed}}$ inside P, then we are done by the Fréchet embedding from Section 7.1.3! So it is enough to focus on the case $D_\delta(P) = \Omega(\widehat{\mathtt{ed}})$.

Using the tree concept described above, the high level idea is that the "leaves" representing letters in the string would eventually be dense (there are not too many of them), and therefore we can hope to associate each such "leaf" with an ℓ_∞ vector that is far from $1 - O(\delta)$ fraction of all other leaves, since it must become "dense" at some level. Since different leaves can become dense at different levels, we construct ℓ_∞ vectors and concatenate all of them to reach the final embedding.

In order to keep our closest pair close, we need to carefully distribute ℓ_∞ vectors among its descendants (by assigning weights to edges and multiplying the ℓ_∞ vectors by such weights). Here we use a randomized construction for choosing the weights reminiscent of traditional LSH constructions.

The eventual embedding, for the case when $D_\delta(P) = \Omega(\widehat{\mathtt{ed}})$ is as follows. We start with the roots of the aforementioned trees $\{T_i\}_i$ for all strings $x \in P$. For each node v of each tree we build an ℓ_∞ embedding, in a top-bottom fashion. In particular, the embedding for a child v' of v will append some fraction $t \in [0, 1]$ of the embedding of v together with a Fréchet embedding of all substrings at the current

level, corresponding to some length w, while the other child will be weighted by $1-t$. The main technical induction will ensure that the sparsity D_{δ_w} of the vectors at the current level w does not drop too much (as long as δ_w goes a bit up as w goes down). At the end of the day, the embedding of a concrete string q produces $\Theta(k^2)$ vectors in ℓ_∞ corresponding to the leaves of the trees of q — these vectors will form the set corresponding to q in $\mathsf{EMD}_{\ell_\infty}$.

7.2 Definitions and preliminaries

7.2.1 Basic notation

For a vector $x \in S^n$, for a set of objects S, the notation x_* refers to the multi-set $\{x_i\}_{i=1\ldots n}$. For example, for a matrix $x \in (\mathbb{R}^k)^n$, $x_* \subset \mathbb{R}^k$ is the n-multi-set of the k-dimensional vectors. Furthermore, $x_{*,*}$ refers to the multi-set of nk real values.

We will often consider product spaces of metrics. For metric $(\mathcal{X}, \mathsf{d}_\mathcal{X})$ and $(\mathcal{Y}, \mathsf{d}_\mathcal{Y})$, and for pointsets $X \subset \mathcal{X}, Y \subset \mathcal{Y}$ both indexed by $[n]$, we use $X \oplus_p Y$, for $p \in [1, \infty]$ to denote the pointset $(x_i, y_i) \in X \times Y$ under the metric $\ell_p(\mathsf{d}_\mathcal{X}, \mathsf{d}_\mathcal{Y})$ (i.e., compute the distance in each metric and then take the ℓ_p norm of these 2 numbers). This can be generalized to multi-dimensional products as well. We will sometimes drop the subscript p when clear from the context.

For runtime/space bounds, we use standard notation $\tilde{O}(f(n))$ to denote $O(f(n) \cdot (\log f(n))^{O(1)})$.

7.2.2 Similarity embeddings

While we state the embeddings into similarity measures using a general similarity measure $\mathcal{S} : \mathbb{R}_+^m \times \mathbb{R}_+^m \to [0,1]$, we will always use the concrete case of $\mathcal{S}(x,y) = M(x,y) \triangleq \frac{2\sum_i \min\{x_i,y_i\}}{\sum_i x_i+y_i}$, Sørensen-Dice coefficient.

We first restate the UASE definition from the Introduction.

Definition 115 (Uniform Average Similarity Embedding (UASE))**.** *Fix a metric $(\mathcal{X},\mathsf{d})$, similarity measure $\mathcal{S}$ and a point-set $P \subseteq \mathcal{X}$. We say P admits a (c,δ,m)-Uniform Average Similarity Embedding (UASE) if there exists a (data-dependent) transformation $\varphi_P : \mathcal{X} \to \mathbb{R}_+^m$, such that:*

1. $\mathbb{E}_{p',p''\in P}[\mathcal{S}(\varphi_P(p'),\varphi_P(p''))] \leq \delta$.
2. *For any $p,q \in \mathcal{X}$, we have:*

$$\frac{\log\left(1/\mathcal{S}(\varphi_P(p),\varphi_P(q))\right)}{\log(1/\delta)} \leq c \cdot \frac{\mathsf{d}(p,q)}{\mathbb{E}_{p',p''\in P}[\mathsf{d}(p',p'')]}.$$

We say a metric $(\mathcal{X},\mathsf{d})$ admits a (c,δ,m)-UASE if each $P \subseteq \mathcal{X}$ admits a (c,δ,m)-UASE.

Definition 116 (Weighted Average Similarity Embedding, WASE)**.** *Fix a metric $(\mathcal{X},\mathsf{d})$, similarity measure $\mathcal{S}$ and a point-set $P \subseteq \mathcal{X}$. We say P admits a (c,m)-Weighted Average Similarity Embedding (WASE) if there exists a (data-dependent) transformation $\varphi_P : \mathcal{X} \to \mathbb{R}_+^{m/s\times s}$ for some $s \in \mathbb{N}$, such that for any $p,q \in \mathcal{X}$, we have:*

$$\sum_{i\in[m/s]} \frac{\|\varphi_P(p)_i + \varphi_P(q)_i\|_1}{\|\varphi_P(p)+\varphi_P(q)\|_1} \cdot \min\left\{1 + 1/l_i, \frac{\log 1/\mathcal{S}(\varphi_P(p)_i,\varphi_P(q)_i)}{l_i}\right\} \leq c \cdot \frac{\mathsf{d}(p,q)}{\mathbb{E}_{p',p''\in P}[\mathsf{d}(p',p'')]}, \tag{7.1}$$

where

$$l_i = \log 1/\ \mathbb{E}_{p',p''\sim\mathcal{D}_i}[\mathcal{S}(\varphi_P(p')_i, \varphi_P(p'')_i)], \qquad \mathcal{D}_i(x,y) \triangleq \frac{\|\varphi_P(x)_i\|_1 + \|\varphi_P(y)_i\|_1}{2n \cdot \|\varphi_P(P)_i\|_1}.$$

For any δ with $\log 1/\delta \geq \max_i l_i$, we also refer to the embedding as (c,δ,m)-WASE. We also say $(\mathcal{X},\mathsf{d})$ admits (c,δ,m)-WASE if all subsets $P \subset \mathcal{X}$ of size at least $O(1/\delta)$ admit one.

We allow for randomized embedding and in such cases the guarantee is promised whp (wrt parameter m).

Finally, we say the embedding has aspect ratio $\kappa > 1$ if, for every $i \in [m/s]$, the ratio of all non-zero coordinates in $(\varphi_P)_i$ map is at most κ.

Data Dependent Hashing. We now define the notion of (r,δ)-Data Dependent Hashing:

Definition 117 (Data-Dependent Hashing (DDH))**.** *Fix a metric $(\mathcal{X},\mathsf{d})$, a point-set $P \subseteq \mathcal{X}$, and radius $r > 0$. For $\delta > 2/n$, we say P admits (r,δ)-Data-Dependent Hashing (DDH) if there exists a randomized map $h_P : \mathcal{X} \to [O(1/\delta)]$ of each point to a bucket, such that:*

- *for any $q \in \mathcal{X}$ and $p \in P$ at distance $\mathsf{d}(q,p) \leq r$, we have $\Pr_{h_P}[h_P(q) = h_P(p)] \geq \delta^{\mathsf{d}(q,p)/r}$;*
- *for any $p \in P$, we have $\Pr_{h_P, p' \in P}[h_P(p) = h_P(p')] \leq \delta$.*

7.3 Computational efficiency of embeddings

While the above embedding/hashing definitions are purely existential statements, we also need to specify their computational properties for use inside NNS data structures.

Hence, for and embedding φ_P or a hash function h_P, which are build as a function of a dataset P of size n, we use the following notions:

- Preprocessing time F_p: time to compute the function φ_P/h_P for the given dataset P as well as $\varphi(p)$ for all $p \in P$. We think of the function φ_P/h_P as a data structure that can later answer queries of the form "given $q \in \mathcal{X}$, what is $\varphi_P(q)$ or $h_P(q)$?".
- Space F_s: the space occupied by the above data structure;
- Query time F_q: the time it takes to answer the above query, i.e., compute $\varphi_P(q)/h_P(q)$ for a given new q.

7.4 Aspect ratios

For a metric $(\mathcal{X}, \mathsf{d})$, we say the aspect ratio is $\kappa_{\mathcal{X}} \geq 1$ if the ratio of any two non-zero distance values is $\leq \kappa_{\mathcal{X}}$. Similarly, for an embedding $\phi : \mathcal{X} \to \mathbb{R}^m_+$, its aspect ratio is $\kappa \geq 1$ if the ratio of any two non-zero coordinates is $\leq \kappa$ (though the definition for WASE is a bit more nuanced; see Definition 116).

We note that the only purpose of introducing the aspect ratio is that it is a convenient way to control the computational performance of our embeddings, and in particular the runtimes are polynomial in the aspect ratios.

7.5 Weighted average similarity embedding for ℓ_p norms

In this section, we design a WASE for ℓ_p, proving the following theorem.

Theorem 118 (WASE for ℓ_p)**.** *For $p \geq 2$, the space $(\mathbb{R}^d, \ell_p)$ admits a $(O(\log(p)), \delta, O(d))$-WASE, for $\delta = 2^{-O(p)}$. The computational efficiency on a dataset of size n is as follows: $F_p = \tilde{O}(nd)$ preprocessing time, $F_s = O(nd)$ space, and $F_q = O(d)$ query time. Furthermore, the aspect ratio is polynomial in dimension d.*

Note that the theorem extends to ℓ_∞ by taking $p = O(\log d)$, for which ℓ_p and ℓ_∞ are within a constant factor.

We first show that this task can be reduced to embedding a 1-dimensional dataset with a certain WASE (that depends on p). The main technical lift will then be to design such an embedding for a 1-dimensional dataset, which is captured by the following lemma.

Lemma 119. *For any $x \in \mathbb{R}^n$, with $\|x\|_p^p = n$ and $\sum_i \operatorname{sign}(x_i) \cdot |x_i|^p = 0$, we have an embedding $\phi : \mathbb{R} \to [2^{-p}, \infty)$ such that $\|\phi(x)\|_\infty = O(\|x\|_\infty)^{O(p)}$, and for any $a, b \in \mathbb{R}$, we have:*

$$|\phi(a) + \phi(b)|^{1/p} \cdot \min\left\{1 + 1/l, \tfrac{\log(1/M(\phi(a),\phi(b)))}{l}\right\} = O(\log p) \cdot |a - b|,$$

where

$$l = \log(1/\operatorname{\mathbb{E}}_{i,j\sim\mathcal{D}_x}[M(\phi(x_i), \phi(x_j))]), \qquad \mathcal{D}_x(i,j) \triangleq \frac{|\phi(x_i)| + |\phi(x_j)|}{2n \cdot \|\phi(x_*)\|_1}.$$

The computational efficiency is as follows: $F_p = \tilde{O}(n)$ preprocessing time, $F_s = O(n)$ space, and $F_q = O(1)$ query time.

While we prove the lemma in the rest of the section, we first show how to complete the proof of Theorem 118 assuming the above lemma.

Proof of Theorem 118. Fix a dataset P of size n. For $e \in [d]$ and $i \in [n]$, let $P_e(i)$ refer to the e^{th} coordinate of the i^{th} point in P; then $P_e \in \mathbb{R}^n$ just denotes the e-th

coordinate of the dataset. As is common, we shift the dataset, so that, for every dimension e, we have that $\sum_i \text{sign}(P_e(i)) \cdot |P_e(i)|^p = 0$; see, e.g., [145]. Note this transformation does not change any distances.

Let $\widehat{\mathsf{d}}_e \triangleq \mathbb{E}_{i,j\in[n]}[(P_e(i) - P_e(j))^p]$ be the average distance of $\mathcal{X}$ and let $\widehat{\mathsf{d}} \triangleq \sum_e \widehat{\mathsf{d}}_e = \mathbb{E}_{p',p''\in P}[\|p' - p''\|_p^p]$. Let $L \subset [d]$ be the set of coordinates e where $\widehat{\mathsf{d}}_e < \frac{1}{2d}\widehat{\mathsf{d}}$. We then remove (ignore) all coordinates e (i.e., we only consider vectors $q \in \mathbb{R}^{d-|L|}$ where coordinates in L are deleted). Note that this operation is non-expanding. Also, the average distance $\hat{d}$ decreases by at most a factor 2 (which will be enough to affect the distortion by at most a factor 2). We can now assume $\widehat{\mathsf{d}}_e \geq \frac{1}{2d}\widehat{\mathsf{d}}$.

Fix coordinate e. We define $\widetilde{P_e}(i) = P_e(i)/\alpha_e$ for unique rescaling factor α_e, β_e such that $\left\|\widetilde{P_e}\right\|_p^p = n$. We note that $\alpha_e \geq \frac{1}{2}\widehat{\mathsf{d}}_e^{\,1/p}$ since, for any vector x, we have

$$\alpha_e^p = \frac{\|x\|_p^p}{n} \geq \frac{\mathbb{E}_i\left[\sum_j |x_i - x_j|^p\right]}{2^p n} = 2^{-p}\,\mathbb{E}_{i,j}\left[|x_i - x_j|^p\right] = 2^{-p}\widehat{\mathsf{d}}_e. \tag{7.2}$$

First consider the case when $\max_{i,j\in[n]}\{\|P_*(i) - P_*(j)\|_p\} = O(d^{1/p}\widehat{\mathsf{d}}^{1/p})$ (this is needed to bound the aspect ratio of the final embedding). Then $\max_{i,j\in[n]}\{|P_e(i) - P_e(j)|\} = O(d^{1/p}\widehat{\mathsf{d}}^{1/p})$, and hence, noticing $\left\|\widetilde{P_e}\right\|_\infty = \max_{i,j}\{|\widetilde{P_e}(i) - \widetilde{P_e}(j)|\} = \frac{1}{\alpha_e} \cdot \max_{i,j\in[n]}\{|P_e(i) - P_e(j)|\}$, and using Eqn. (7.2), we have:

$$\left\|\widetilde{P_e}\right\|_\infty \leq O(d^{1/p}\widehat{\mathsf{d}}^{1/p})/\alpha_e = O(d^{1/p}\widehat{\mathsf{d}}^{1/p})/0.5\widehat{\mathsf{d}}_e^{\,1/p} = O(d^{2/p}).$$

Now, use Theorem 119 on vectors $\widetilde{P_e}(i)$ to get $\phi_e : \mathbb{R} \to [2^{-p}, d^{O(1)}]$. Finally, to define $\psi(q)$, we first shift q with the same shift used to center the data P at the beginning, and rescale each coordinate e by α_e as well, to obtain vector $(\widetilde{q_e})_e$. We now define $\psi(q) \triangleq \oplus_e \widehat{\mathsf{d}}_e \cdot \phi_e(\widetilde{q_e})$.

Now, fix $a, b \in \mathbb{R}^d$. For any $a_e, b_e \in \mathbb{R}$, with $\widetilde{a_e}, \widetilde{b_e}$ being the shifting/rescaling, we have by Eqn. (7.2):

$$|\widetilde{a_e} - \widetilde{b_e}|^p \leq \frac{(2|a_e - b_e|)^p}{\widehat{\mathsf{d}}_e}. \tag{7.3}$$

For $e \in [d]$, define $\mathcal{D}_e(i,j) \triangleq \mathcal{D}_{\widetilde{P_e}}(\widetilde{P}_e(i), \widetilde{P}_e(j))$, and $\tau_e \triangleq \min\left\{1 + 1/l_e, \frac{\log(1/M(\psi(a)_e, \psi(b)_e))}{l_e}\right\}$ where $l_e = \log(1/\mathbb{E}_{i,j\sim\mathcal{D}_e}[M(\psi(\widetilde{P}(i))_e, \psi(\widetilde{P}(j))_e)])$. Notice that by Theorem 119:

$$|\psi(a)_e + \psi(b)_e|^{1/p} \cdot \tau_e = O(\log p) \cdot |a_e - b_e|. \tag{7.4}$$

Finally, define

$$\rho \triangleq \sum_{e\in[d]} \frac{|\psi(a)_e + \psi(b)_e|}{\|\psi(a) + \psi(b)\|_1} \cdot \tau_e.$$

We analyze ρ:

$$\begin{aligned}
\rho &= \sum_{e\in[d]} \frac{|\psi(a)_e + \psi(b)_e|}{\|\psi(a) + \psi(b)\|_1} \cdot \tau_e \\
&\overset{\text{Jensen's}}{\leq} \left(\sum_{e\in[d]} \frac{|\psi(a)_e + \psi(b)_e|}{\|\psi(a) + \psi(b)\|_1} \cdot \tau_e^p\right)^{1/p} \\
&\overset{\text{min value of } \psi}{\leq} \left(\sum_{e\in[d]} \frac{|\psi(a)_e + \psi(b)_e|}{\sum_{e'\in[d]} 2\cdot 2^{-p}\cdot \widehat{\mathsf{d}_{e'}}} \cdot \tau_e^p\right)^{1/p} \\
&\overset{Eqn.(7.4)}{=} \left(\frac{2^p}{2\widehat{\mathsf{d}}} \sum_{e\in[d]} O(\log p)^p \cdot |a_e - b_e|^p\right)^{1/p} \\
&= O(\log p) \cdot \left(\frac{\sum_{e\in[d]} |a_e - b_e|^p}{\widehat{\mathsf{d}}}\right)^{1/p} \\
&= O(\log p) \cdot \frac{\|a - b\|_p}{\widehat{\mathsf{d}}^{1/p}}.
\end{aligned}$$

Finally, noting that $\widehat{\mathsf{d}}^{1/p} \geq \mathbb{E}_{p',p''\in P}[\|p' - p''\|_p]$, we conclude that ψ is $(O(\log p), O(d))$-WASE for P as needed.

We also need to consider the case when $\max_{i,j\in[n]}\{\|P_*(i)-P_*(j)\|_p\} \gg d^{1/p}\widehat{\mathsf{d}}^{1/p}$. For large enough constant $C > 1$, take the projection $P'_e(i) = Cd^{1/p}\widehat{\mathsf{d}}_e^{1/p}$ for any point with $\|P_e(i)\| \geq Cd^{1/p}\widehat{\mathsf{d}}_e^{1/p}$ (and otherwise leave unchanged). Then define $\widehat{\mathsf{d}'}^{1/p}$ to be the average distance of the set P'. If $\widehat{\mathsf{d}'}^{1/p} = \Omega(\widehat{\mathsf{d}}^{1/p})$ then we can repeat the argument from above for P', noting that it is non-expanding and contracting by only a constant on average. Otherwise, the pointset P has an $O(1)$-average embedding into ℓ_1 (in the sense of the Definition Definition 8), by setting $\phi(P(i)) = \|P(i)\|_p$, and hence, by [74], into WASE (even UASE in fact).

The computational efficiency guarantees follow immediately. There are just a couple nuances in preprocessing to highlight. The first one is that we need to recenter the each coordinate e: namely find a shift s such that $\sum_{p\in P}\operatorname{sign}(p_e-s)\cdot|p_e-s|^p = 0$. This can be done by binary search; see, e.g., [145]. The second one is that we need to estimate the average distance in a coordinate $\hat{\mathsf{d}}_e = \mathbb{E}_{p,p'\in P}[|p_e-p'_e|^p]$. We can estimate this quantity by sampling, noting that $O(1)^p$ approximation suffices.

Finally, the aspect ratio bound follows from the range of the embedding from Lemma 119. □

7.5.1 Preliminaries

We now introduce some extra notation used in this section. We define the space of intervals $\mathcal{I}$ (over the positive real line). For an interval $I = [a,b] \in \mathcal{I}$ where $0 \leq a \leq b$, we denote $\mu(I) = b-a$ as the measure (length) of the interval, and, for two intervals $I_1 = [a_1,b_1], I_2 = [a_2,b_2]$ say $I_1 < I_2$ if $b_1 < a_2$. We also define projection to an interval:

Definition 120. *For scalar $a \geq 0$, and interval $I \in \mathcal{I}$ we define* $\operatorname{proj}_I(a) \triangleq \mu([0,a]\cap$

I).

In this section whenever a scalar operator is applied on vectors (including proj_I), we mean it is applied coordinate-wise and the output is also a vector.

Fact 121. *Fix $x \in \mathbb{R}^n$ and $1 \le q \le p$. Suppose $\Pr_{i\in[n]}\left[|x_i| = \|x\|_\infty\right] \ge 2^{-q}$, then $\|x\|_q \ge \frac{1}{2} \cdot n^{1/q-1/p} \cdot \|x\|_p$.*

Proof. First, $\|x\|_q^q \ge n \cdot 2^{-q} \cdot \|x\|_\infty^q$ and hence,

$$\|x\|_q \ge \tfrac{n^{1/q}}{2} \cdot \|x\|_\infty \ge \tfrac{n^{1/q}}{2} \cdot n^{-1/p} \cdot \|x\|_p = \tfrac{1}{2} \cdot n^{1/q-1/p} \cdot \|x\|_p,$$

as needed. □

Lemma 122. *Fix $x \in \mathbb{R}^n_+$, $k \ge 1$ and $e_1, e_2, \ldots, e_k$ such that $\sum_j e_j \le \|x\|_p$. There exists disjoint interval partitioning $\{I_1, I_2, \ldots, I_k\} \subset \mathcal{I}$ of $\mathbb{R}_+$ such that $\left\|\text{proj}_{I_j}(x)\right\|_p \ge e_j$ for all $j \in [k]$.*

Proof. Let $\lambda_0 = 0$ and $\lambda_k = \infty$. For $i \in [k-1]$, let $\lambda_i \triangleq \sup\left\{\lambda : \left\|\text{proj}_{[\lambda_{i-1},\lambda]}(x)\right\|_p \le e_i\right\}$. Let $I_i = [\lambda_{i-1}, \lambda_i)$. Note that by ℓ_p continuity, $\left\|\text{proj}_{I_i}(x)\right\|_p \ge e_i$ unless $\left\|\text{proj}_{[\lambda_{i-1},\infty)}(x)\right\|_p < e_i$. By triangle inequality, the latter never happens and we have that $\left\|\text{proj}_{I_i}(x)\right\| \ge e_i$ for all i. □

Lemma 123. *Let $\{I_1, I_2, \ldots, I_k\} \subset \mathcal{I}$ be disjoint interval partitioning of $\mathbb{R}_+$. Then for any $x \in \mathbb{R}^n$ with $\|x\|_0 \le n/2$, and $p \ge 1$, there exists $j \in [k]$ such that, for $q = p^{j/k}$:*

1. $\left\|\text{proj}_{I_j}(x)\right\|_q \ge \frac{1}{2} \cdot n^{1/q-1/p}\left\|\text{proj}_{I_j}(x)\right\|_p$; *and,*
2. $\left\|\text{proj}_{I_j}(x)\right\|_0 \le n \cdot 2^{-q \cdot p^{-1/k}}$.

Proof. Assume wlog $I_1 < I_2 < \ldots < I_k$. This implies $\text{proj}_{I_j}(x_i) < \mu(I_j) \to \text{proj}_{I_{j+1}}(x_i) = 0$. Hence $\left\|\text{proj}_{I_{j+1}}(x)\right\|_0 \leq n \cdot \Pr_{i\in[n]}\left[\text{proj}_{I_j}(x_i) = \left\|\text{proj}_{I_j}(x)\right\|_\infty\right]$. Now for $j \in [k]$, let $q_j \triangleq p^{j/k}$. If $\left\|\text{proj}_{I_1}(x)\right\|_{q_1} \geq \frac{1}{2}\cdot n^{1/q_1-1/p}\left\|\text{proj}_{I_1}(x)\right\|_p$ we are done (both conditions are satisfied). Otherwise, let $j^* \geq 2$ be the minimal j where $\left\|\text{proj}_{I_j}(x)\right\|_{q_j} \geq \frac{1}{2}\cdot n^{1/q_j-1/p}\left\|\text{proj}_{I_j}(x)\right\|_p$ (noting that such j^* always exists since the above is true for $q_k = p$). Then, by Fact 121, $\Pr_{i\in[n]}\left[\text{proj}_{I_{j^*-1}}(x)_i = \left\|\text{proj}_{I_{j^*-1}}(x)\right\|_\infty\right] < 2^{-q_{j^*-1}}$, and hence,

$$\left\|\text{proj}_{I_{j^*}}(x)\right\|_0 \leq n \cdot \Pr_{i\in[n]}\left[\text{proj}_{I_{j^*-1}}(x)_i = \left\|\text{proj}_{I_{j^*-1}}(x)\right\|_\infty\right] \leq n \cdot 2^{-q_{j^*-1}} = n \cdot 2^{-q_{j^*}/p^{1/k}},$$

as needed. □

Claim 124. *Fix $x \in \mathbb{R}^n_+$ and any $p \geq 1$. There exists a mapping $\psi : \mathbb{R}_+ \to [2^{-p}, \infty)$ such that:*

- *For $s \geq 1$, $\psi(s) = (O(s))^p$.*
- *$\|\psi(x)\|_1 \geq \|x\|_p^p$;*
- *$\Pr_{i\in[n]}[\psi(x_i) > 2^{-p}] = \frac{\|x\|_0}{n}$; and*
- *for each $s, t \in \mathbb{R}_+$, $|\psi(s) + \psi(t)|^{1/p} \cdot \min\{p, \log 1/M(\psi(s), \psi(t))\} \leq 4p \cdot |s - t|$.*

Proof. Define $\psi(s) \triangleq (s + 0.5)^p$. Then the first 3 conditions are immediate. For the 4th condition, we assume w.l.o.g $t \leq s$. Now, if $2t + 1 < s$, we have that $\|\psi(s) + \psi(t)\|_1^{1/p} \leq s+t+1 < 3(s-t)$ which implies the claim for this case. Otherwise:

$$\log 1/M(\psi(s), \psi(t)) = \log \frac{\psi(s) + \psi(t)}{2\min\{\psi(s), \psi(t)\}} \leq \log \frac{\psi(s)}{\psi(t)} = \log \frac{(s + 0.5)^p}{(t + 0.5)^p} \leq p \log\left(1 + \frac{4|s - t|}{s + t + 1}\right)$$

Now since $|\psi(s)+\psi(t)|^{1/p} = ((s+0.5)^p + (t+0.5)^p)^{1/p} \leq s+t+1$, hence:

$$|\psi(s)+\psi(t)|^{1/p} \cdot \log 1/M(\psi(s),\psi(t)) \leq (s+t+1)\cdot p \log\left(1+\tfrac{4|s-t|}{s+t+1})\right) \leq 4p\cdot|s-t|,$$

as needed. □

7.5.2 Construction: proof of Lemma 119

Now let $x^{(+)}$ be x after zeroing out all negative coordinates and $x^{(-)}$ be $-x$ after similar zeroing. The construction is merely applying the above Lemmas to one of $\{x^{(+)}, x^{(-)}\}$. In particular, choose $\chi \in \{\pm\}$ such that x^χ is a most $n/2$ sparse; note that $\|x^\chi\|_p^p = \|x\|_p^p/2 = n/2$ by the subsectionons of the lemma. We use Lemma 122 with $k = \log_2 p$ to partition $\mathbb{R}_+$ into intervals with $\left\|\text{proj}_{I_j}\left(x^{(\chi)}\right)\right\|_p \geq n^{1/p}\cdot\gamma_j$, where $\gamma_j = \max\{1/4\log(p), p^{j/k}/4p\}$. Then, we use Lemma 123 to find an interval $I^{(\chi)}$ such that, for $q^{(\chi)} = p^{j/k}$ and $\gamma^{(\chi)} = \gamma_j$:

1. $\left\|\text{proj}_{I^{(\chi)}}\left(x^{(\chi)}\right)\right\|_{q^{(\chi)}} \geq \frac{1}{2}\cdot n^{1/q^{(\chi)}-1/p}\left\|\text{proj}_{I^{(\chi)}}\left(x^{(\chi)}\right)\right\|_p \geq \frac{\gamma^{(\chi)}}{2} n^{1/q^{(\chi)}}$; and,
2. $\left\|\text{proj}_{I^{(\chi)}}(x^{(\chi)})\right\|_0 \leq n\cdot 2^{-q^{(\chi)}\cdot p^{-1/k}} < n\cdot 2^{-q^{(\chi)}/2}$.

Now, let $y^{(\chi)} = 2/\gamma^{(\chi)}\cdot\text{proj}_{I^{(\chi)}}(x^{(\chi)})$. Then, $\left\|y^{(\chi)}\right\|_{q^{(\chi)}}^{q^{(\chi)}} \geq n$. Apply Claim 124 using $x = y^{(\chi)}$ and $p = q^{(\chi)}$ to get a mapping $\psi^{(\chi)} : \mathbb{R}_+ \to \mathbb{R}_+$.

The final mapping is defined to be as a function of χ: if $\chi = +$,

$$\phi(a) = \psi^+(2/\gamma^{(+)}\cdot\text{proj}_{I^{(+)}}(\max\{0,a\}))$$

otherwise

$$\phi(a) = \psi^-(2/\gamma^{(-)}\cdot\text{proj}_{I^{(-)}}(\max\{0,-a\}).$$

Proof of Lemma 119. First, we argue $\|\phi(x)\|_1 \geq n$. In particular, by Claim 124:

$$\begin{aligned}\left\|\psi^{(\chi)}\left(2/\gamma^{(\chi)}\cdot \mathrm{proj}_{I^{(\chi)}}\left(x^{(\chi)}\right)\right)\right\|_1 &\geq \left\|2/\gamma^{(\chi)}\cdot \mathrm{proj}_{I^{(\chi)}}\left(x^{(\chi)}\right)\right\|_{q^{(\chi)}}^{q^{(\chi)}}\\ &\geq \left\|1/\gamma^{(\chi)}\cdot n^{1/q^{(\chi)}-1/p}\mathrm{proj}_{I^{(\chi)}}\left(x^{(\chi)}\right)\right\|_p^{q^{(\chi)}}\\ &= n\cdot\left\|\tfrac{1}{\gamma^{(\chi)}n^{1/p}}\mathrm{proj}_{I^{(\chi)}}\left(x^{(\chi)}\right)\right\|_p^{q^{(\chi)}}\\ &\geq n.\end{aligned}$$

Next we argue that $\|\phi(x)\|_\infty = \|x\|_\infty^{O(p)}$. Indeed by Claim 124 we have that:

$$\|\phi(x)\|_\infty \leq O(\|x\|_\infty/\gamma^{(\chi)})^{q^{(\chi)}} = O(p/q^{(\chi)}\cdot\|x\|_\infty)^{q^{(\chi)}} = O(\|x\|_\infty)^{O(p)}.$$

Next we argue that $E_{(i,j)\sim\mathcal{D}_x}[M(\psi^\chi(y_i^{(\chi)}),\psi^\chi(y_j^{(\chi)}))]) \leq 2^{-q^{(\chi)}/8}$. Indeed, by Claim 124, we have that $\Pr_{i\in[n]}\left[\psi^\chi(y_i^{(\chi)}) > 2^{-q^{(\chi)}}\right] \leq \frac{\|y^{(\chi)}\|_0}{n} \leq 2^{-q^{(\chi)}/2}$. Now, letting $\psi_i \triangleq \psi^\chi(y_i^{(\chi)})$ we note that $\mathcal{D}_x(i,j) = \frac{\|\psi_i\|_1+\|\psi_j\|_1}{2n\|\psi\|_1}$ with ψ being the vectors of ψ_i's, we further obtain:

$$\begin{aligned}E_{(i,j)\sim\mathcal{D}_x}[M(\psi_i,\psi_j)] &\leq \sum_{i,j}\tfrac{2\min\{\psi_i,\psi_j\}\cdot(\psi_i+\psi_j)}{(\psi_i+\psi_j)\cdot 2n\|\psi\|_1}\\ &= \tfrac{1}{n\|\psi\|_1}\cdot\sum_{i,j}\min\{\psi_i,\psi_j\}\\ &\leq \tfrac{1}{n\|\psi\|_1}\cdot\left(\sum_{i:\psi_i>2^{-q^{(\chi)}}}\sum_j\psi_j + \sum_{i:\psi_i=2^{-q^{(\chi)}}}\sum_j\psi_i\right)\\ &\leq \tfrac{1}{n\|\psi\|_1}\cdot\left(\sum_{i:\psi_i>2^{-q^{(\chi)}}}\|\psi\|_1 + n^2\cdot 2^{-q^{(\chi)}}\right)\\ &\leq \tfrac{1}{n\|\psi\|_1}\cdot\left(2^{-q^{(\chi)}/2}\cdot n\cdot\|\psi\|_1 + n\cdot\|\psi\|_1\cdot 2^{-q^{(\chi)}}\right)\\ &< 2^{-q^{(\chi)}/8},\end{aligned}$$

and hence $l \triangleq \log 1/(E_{(i,j)\sim\mathcal{D}_x}[M(\psi_i,\psi_j)]) \geq q^{(\chi)}/8 \geq 1/4$. On the other hand, we have by the guarantees of Claim 124 that:

- for any $a \in \mathbb{R}$: $\phi(a) \geq 2^{-q^{(\chi)}} \geq 2^{-p}$; and,
- $|\phi(s)+\phi(t)|^{1/q^\chi} \cdot \min\{q^\chi, \log 1/M(\phi(s),\phi(t))\} \leq 4 \cdot q^{(\chi)} \cdot 4/\gamma^{(\chi)} \cdot |s-t| = O(\log p) \cdot q^{(\chi)} \cdot |s-t|$.

Overall, this gives:

$$
\begin{aligned}
&|\phi(s)+\phi(t)|^{1/p} \cdot \min\left\{1+\tfrac{1}{l}, \tfrac{\log 1/M(\phi(s),\phi(t))}{l}\right\} \\
&\qquad \leq |\phi(s)+\phi(t)|^{1/p-1/q^{(\chi)}} \cdot |\phi(s)+\phi(t)|^{1/q^{(\chi)}} \tfrac{\min\left\{1+q^{(\chi)}/8, \log 1/M(\phi(s),\phi(t))\right\}}{q^{(\chi)}/8} \\
&\qquad \leq 2^{-q^{(\chi)}(1/p-1/q^{(\chi)})} \cdot |\phi(s)+\phi(t)|^{1/q^{(\chi)}} \tfrac{\min\left\{5q^{(\chi)}, \log 1/M(\phi(s),\phi(t))\right\}}{q^{(\chi)}/8} \\
&\qquad \leq 2 \cdot O(\log p) \cdot |s-t| \\
&\qquad = O(\log p) \cdot |s-t|.
\end{aligned}
$$

The computational efficiency guarantees are nearly immediate to obtain. The only non-immediate step is that one has to compute the intervals I_j, wnich can be accomplished by binary search. □

7.6 Similarity embedding for matching distances

In this section we design similarity embeddings for the Earth Mover Distance for certain instances. While the natural goal would be to solve the general instances, we only manage to find embeddings for specific distances which we call "sparse instances". In a later section, we use the embedding for such "sparse instances" to design WASE for edit distance (for all instances).

We first define a notion of *distance sparsity* from a point to a dataset.

Definition 125. *Fix a point $p \in \mathcal{X}$ and a point-set $P \subseteq \mathcal{X}$. The distance sparsity of point p, denoted $D^{\mathsf{d}}_{\delta}(p, P)$, is the δ-quantile of distances from p to P, i.e., $D^{\mathsf{d}}_{\delta}(p, P) = t$ if t is the smallest with $\Pr_{p' \in P}[\mathsf{d}(p, p') \leq t] = \delta$. We also define the distance sparsity of a dataset as the average distance sparsity of all its points:*

$$D^{\mathsf{d}}_{\delta}(P) \triangleq \mathbb{E}_{p \in P}\left[D_{\delta}(p, P)\right].$$

When d is clear from context, we sometimes omit it from D_{δ} notation for the sake of cleaner exposition.

The main statement of this section that one can get WASE for $\mathsf{EMD}_{\mathcal{X}}$, if the collections have good enough distance sparsity:

Lemma 126. *Fix a metric $(\mathcal{X}, \mathsf{d})$. Consider a family of multi-sets $P \in (\mathcal{X}^k)^n$ such that the average distance is $\widehat{\mathsf{d}} = \mathbb{E}_{p,p' \in P}[\mathsf{EMD}_{\mathsf{d}}(p', p'')]$ and $D_{\delta}(P_{*,*}) \geq t$ for some $t > 0$; where we also require that $\delta < 1/k^2$. Suppose P admits (c, δ, m)-WASE for some $\delta < 1/\operatorname{poly}(\kappa\kappa_{\mathcal{X}}m)\}$ aspect ratios $\kappa, \kappa_{\mathcal{X}}$, and $\eta > 1/k$.*

Then the family P admits $(O(c \cdot \frac{\widehat{\mathsf{d}}}{kt}), 1/m', m')$-WASE with respect to $(\mathcal{X}^k, \mathsf{EMD}_{\mathsf{d}})$, where $m' = \operatorname{poly}(\kappa\kappa_{\mathcal{X}}m \cdot 1/\delta \cdot \log \mathcal{X})$.

The rest of this section is dedicated for proving Lemma 126.

7.6.1 Auxiliary lemmas

The main challenge for constructing embedding for "sparse EMD" is that the sparsity guarantees we have are "on average only", i.e., we may have some objects that are more dense than other, and we will need to create several discretization to cater to all situations. For that purpose, we first introduce two ancillary constructions. While

the guarantees of those constructions may not make immediate sense apriori, they will provide what we need for the overall solution.

First, we need to show WASE implies UASE, assuming the dataset is sparse (namely, has no dense balls):

Lemma 127. *Fix a metric $(\mathcal{X}, \mathsf{d})$ admitting a (c, δ, m)-WASE embedding for P for some $\delta < 1/\operatorname{poly}(\kappa\kappa_{\mathcal{X}} m)$. Fix a dataset $P \subseteq \mathcal{X}$, and let $\widehat{\mathsf{d}} = \mathbb{E}_{p',p''}[\mathsf{d}(p',p'')]$. Fix $\mathsf{d}' \leq \widehat{\mathsf{d}}$ such that there's no ball of radius $\mathsf{d}'/2$ of size $\delta n/2$.*

Then, there exists a $(O(c\widehat{\mathsf{d}}/\mathsf{d}'), 3\delta, O(\delta^{-2}\log\mathcal{X}))$-UASE for P; furthermore, all output vectors have ℓ_1 norm 1.

Proof. We apply Lemma 147 with $\tau = \delta$ to conclude that there exists a (r, δ^2)-DDH for P where $r = \mathsf{d}'/2c$.

We construct the UASE as follows. Let $L = \frac{O(1)}{\delta}\log\mathcal{X}$, and let $m = O(1/\delta)$ be the range of DDH. Our map is $\psi : \mathcal{X} \to \{0,1\}^{L\times m}$. We draw L iid DDH functions $h_1, \dots h_L : \mathcal{X} \to [m]$, as well as L random functions $r_i : [m] \to [m]$. Then a point $q \in \mathcal{X}$ is mapped to the concatenation of the unitary representation of $r_i(h_i(q))$: $\psi(q)_{i,j} = \mathbb{1}[h_i(q) = j]$. By Chernoff bound, it is immediate that, with high probability, for all $p, q \in \mathcal{X}$:

$$M(\psi(p),\psi(q)) \in \Theta(1)(\Pr_h[h(p) = h(q)] \pm \delta).$$

Hence, we have:

- $\mathbb{E}_{p',p''\in P}[M(\psi(p'),\psi(p''))] \leq O(\Pr_{h,p',p''}[h(p') = h(p'')] + \delta) \leq 3\delta$;
- $\log 1/M(\psi(p),\psi(q)) \leq -\log(\Pr_h[h(p) = h(q)] - \delta) + O(1)$. If $\mathsf{d}(p,q) \leq r$, then $\frac{\log 1/M(\psi(p),\psi(q))}{\log 1/\delta} \leq \frac{-\log(\delta^{\mathsf{d}(p,q)/r} - \delta) + O(1)}{\log 1/\delta} = O(\mathsf{d}(p,q)/r) = O(c\widehat{\mathsf{d}}/\mathsf{d}' \cdot \frac{\mathsf{d}(p,q)}{\widehat{\mathsf{d}}})$. Otherwise, if $\mathsf{d}(p,q) > r$, we have that $\frac{\log 1/M(\psi(p),\psi(q))}{\log 1/\delta} \leq 1 \leq O(c\widehat{\mathsf{d}}/\mathsf{d}' \cdot \frac{\mathsf{d}(p,q)}{\widehat{\mathsf{d}}})$.

Thus we obtained the desired $(O(c\widehat{\mathsf{d}}/\mathsf{d}'), 3\delta, O(\delta^{-2}\log\mathcal{X}))$-UASE. Note that the ℓ_1 norm is the same for any $\psi(p)$, and hence we can rescale it to 1. □

We now strenghen the previous lemma to imply a form of UASE for a fixed sparsity scale Δ.

Lemma 128. *Fix a metric $(\mathcal{X}, \mathsf{d})$ admitting $(c, \sqrt{\delta}, m)$-WASE, parameter $\delta < 1/\operatorname{poly}(m\kappa\kappa_{\mathcal{X}})$, and parameter $\Delta \in \mathbb{R}$. Let $P \subseteq \mathcal{X}$ be a dataset. There exists an embedding $\psi : \mathcal{X} \to \mathbb{R}_+^{m'}$, with $m' = \operatorname{poly}(m\kappa\kappa_{\mathcal{X}}) \cdot O(\log\mathcal{X})$, with the following guarantees:*

1. *For any $p \in \mathcal{X}$ with $D_\delta(p, P) \in [3/4, 3/2] \cdot \Delta$ we have $\|\psi(p)\|_1 \geq 1$.*
2. *For any $p \in \mathcal{X}$ with $D_\delta(p, P) \notin [1/2, 2] \cdot \Delta$ we have $\|\psi(p)\|_1 = O(\delta)$.*
3. *$\mathbb{E}_{p',p''\in P}[\|\min\{\psi(p'), \psi(p'')\}\|_1] = O(\delta)$.*
4. *For any $p, q \in \mathcal{X}$: $\frac{\log(1/M(\psi(p),\psi(q))}{\log(1/\delta)} = O(c) \cdot \frac{\mathsf{d}(p,q)}{\Delta}$.*

Proof. For $a \in \mathcal{X}$, define:

$$\alpha(a) \triangleq \begin{cases} \frac{1}{\Delta}(4D_\delta(a,P) - 2\Delta) & D_\delta(a,P) \in (1/2, 3/4)\Delta \\ 1 & D_\delta(a,P) \in [3/4, 3/2]\Delta \\ \frac{1}{\Delta}(\Delta 4\Delta - 2D_\delta(a,P)) & D_\delta(a,P) \in (3/2, 2)\Delta \\ 0 & \text{otherwise.} \end{cases}$$

There are two constructions, depending on the parameters. Let S be the subset of P with $D_\delta(a,P) > \Delta/2$, $\widehat{\mathsf{d}}$ be the average distance for S, and let $\delta' = \frac{\delta|P|}{|S|}$.

1. Suppose $|S| \geq \sqrt{\delta} \cdot |P|$, equivalently $\delta' \leq \sqrt{\delta}$. Then any ball of radius $\Delta/2$ in S has size at most $\delta n = \delta'|S| \leq \sqrt{\delta} \cdot |S|$. We apply Lemma 127 on S to generate ψ'

as 1-norm $(O(c \cdot \widehat{\mathsf{d}}/\Delta), 3\delta', m')$-UASE for S with m' from the lemma statement. Set $\psi'(a) = \vec{0}$ for $a \notin S$. Then set $\psi(a) = \left(\delta^{1-\alpha(a)} \cdot \psi'(a)\right) \oplus \delta$.

We show conditions (1)-(4) hold. Conditions (1) and (2) holds by construction. In particular, condition (1) holds since for such p, we have $\alpha(p) = 1$, and hence $\delta^{1-\alpha(p)} = 1$. Condition (2) holds since for such p we have $\alpha(p) = 0$ and hence $\delta^{1-\alpha(p)} = \delta$.

For (3), note that for any $p' \in P \setminus S$, we have $\|\psi(p')\|_1 = \delta$ and hence, using UASE guarantees:

$$\begin{aligned}
&\sum_{(p',p'') \in P \times P} \|\min\{\psi(p'), \psi(p'')\}\|_1 \\
&\quad = \sum_{(p',p'') \in S \times S} \|\min\{\psi(p'), \psi(p'')\}\|_1 + \sum_{(p',p'') \in P \times P \setminus S \times S} \|\min\{\psi(p'), \psi(p'')\}\|_1 \\
&\quad = O(\delta' |S|^2) + O(\delta |P|^2) \\
&\quad = O(\delta |P|^2).
\end{aligned}$$

Finally, for (4), suppose $D_\delta(p, P), D_\delta(q, P) > \Delta/2$. We use the following fact:

Fact 129. *Fix $x, y \in \{0,1\}^m$ and positive scalars $a, b > 0$. Then,*

$$\log(1/M(ax, by)) \leq \log(1/M(x, y)) + \log(1/M(a, b)).$$

Proof. Assume w.l.o.g $a \geq b$. Then,

$$\begin{aligned}
\log(\tfrac{1}{M(ax,by)}) &= \log(\tfrac{a\|x\|_1+b\|y\|_1}{2\|\min\{ax,by\}\|_1}) \\
&\leq \log(\tfrac{a(\|x\|_1+\|y\|_1)}{2b\|\min\{x,y\}\|_1}) \\
&= \log(\tfrac{a}{b}) + \log(\tfrac{\|x\|_1+\|y\|_1)}{2\|\min\{x,y\}\|_1}) \\
&\leq \log(\tfrac{1}{M(x,y)}) + \log(\tfrac{1}{M(a,b)}).
\end{aligned}$$

□

Then, using the fact, we have,

$$\begin{aligned}
\log(1/M(\psi(p),\psi(q)) &\leq \log(1/M(\psi'(p),\psi'(q)) + \log(1/M(\delta^{1-\alpha(p)},\delta^{1-\alpha(q)}) \\
&\leq 2c \cdot \tfrac{\mathsf{d}(p,q)}{\Delta/2} \cdot \log(1/\delta) + |\alpha(p)-\alpha(q)|/\Delta \cdot \log(1/\delta) \\
&= O(c) \cdot \tfrac{\mathsf{d}(p,q)}{\Delta} \cdot \log(1/\delta).
\end{aligned}$$

Otherwise, suppose w.l.o.g. $D_\delta(p,P) \leq \Delta/2$, then

$$\begin{aligned}
\log(1/M(\psi(p),\psi(q)) &= \log\left(\tfrac{\|\psi(p)+\psi(q)\|_1}{2\min\{\psi(p),\psi(q)\}}\right) \\
&\leq \log\left(\tfrac{\|\psi(q)\|_1}{\delta}\right) \\
&= \log\left(\tfrac{\delta^{1-\alpha(q)}}{\delta}\right) \\
&= \alpha(q) \cdot \log(1/\delta) \\
&\leq (\tfrac{1}{\Delta} \cdot 4D_\delta(q,P) - 2)\log(1/\delta) \\
&= \tfrac{4}{\Delta} \cdot (D_\delta(q,P) - \Delta/2)\log(1/\delta) \\
&\leq \tfrac{4}{\Delta} \cdot (D_\delta(q,P) - D_\delta(p,P)) \cdot \log(1/\delta) \\
&\leq \tfrac{4}{\Delta} \cdot \mathsf{d}(p,q) \cdot \log(1/\delta).
\end{aligned}$$

2. Now consider the alternative case $|S| < \sqrt{\delta}|P|$, ie., $\delta' > \sqrt{\delta}$. We then set $\psi(a) = (\delta^{1-\alpha(a)}, 0...0)$.

   Again, we verify conditions 1-4. Conditions (1), (2) hold precisely the same. For condition (3), note:

$$\begin{aligned}
&\sum_{(p',p'')\in P\times P} \|\min\{\psi(p'),\psi(p'')\}\|_1 \\
&\quad= \sum_{(p',p'')\in S\times S} \|\min\{\psi(p'),\psi(p'')\}\|_1 + \sum_{(p',p'')\in P\times P\setminus S\times S} \|\min\{\psi(p'),\psi(p'')\}\|_1 \\
&\quad= O(|S|^2) + O(\delta|P|^2) \\
&\quad= O(\delta|P|^2).
\end{aligned}$$

   We now verify condition (4). Indeed, as before:

$$\log 1/M(\psi(p),\psi(q)) = \log 1/M\{\delta^{1-\alpha(p)}, \delta^{1-\alpha(q)}\} \leq O(c)\cdot \mathsf{d}(p,q)/\Delta \cdot \log 1/\delta.$$

□

The third auxiliary lemma is for scalar embedding, and will be used to deal with special pairs of collections that are much denser than the average collection.

Lemma 130. *Fix a non-zero $v \in [0,1]^n$, with $\|v\|_0 \leq n/2$.*

There exists a function $\phi : [0,1] \to (0,1]^{\log(n)}$ with the following guarantees:

1. $\|\phi(1)\|_1 = 1$;

2. *For any p, q:*

$$\sum_{i\in[\log(n)]} \|\phi(p)_i + \phi(q)_i\|_1 \cdot \frac{\log M(\phi(p)_i,\phi(q)_i)}{\log \mathbb{E}_{p',p''\sim\mathcal{D}}[M(\phi(p')_i,\phi(p'')_i)]} = O\left(|p-q|\right),$$

where $\mathcal{D}$ is a distribution over $v \times v$, such that:

$$\mathcal{D}(x,y) \triangleq \frac{\|\phi(x)_i\|_1 + \|\phi(y)_i\|_1}{2n \cdot \|\phi(v)_i\|_1}.$$

Proof. The embedding is similar to the ℓ_p^p one. In particular, for $i \in \{0\} \cup [\log(n)]$ define $\lambda_0 = 0$, $\lambda_{\log(n)} = 1$ and for the rest $\lambda_i \triangleq Q_{1-2^{-i}}(v)$. Define intervals $I_i \triangleq [\lambda_{i-1}, \lambda_i]$. Now, define:

$$\phi_i(x) = 2^{3i\left(\mathrm{proj}_{I_i}(x) - |I_i|\right)/|I_i|} \cdot |I_i|.$$

Notice that $\|\phi(1)\|_1 = \sum_i |I_i| = 1$. Now we bound (2). First of all, note that $\|\phi(v)_i\|_1 \geq |I_i| \cdot n(1 - 2^{-i} + 2^{-i} \cdot 2^{3i}) \geq |I_i| \cdot n \cdot 2^{2i}$. Hence

$$\begin{aligned}
\mathbb{E}_{p',p'' \sim \mathcal{D}_i}[M(\phi(p')_i, \phi(p'')_i)] &= \mathbb{E}_{p',p'' \sim \mathcal{D}_i}\left[\frac{\min\{\phi(p')_i, \phi(p'')_i\}}{n\|\phi(v)_i\|_1/|I_i|}\right] \\
&\leq \frac{\sum_{p:p \leq \lambda_{i-1}} 1 \cdot n + \sum_{p:p > \lambda_{i-1}} \|\phi(v)_i\|_1}{n\|\phi(v)_i\|_1/|I_i|} \\
&\leq 2^{-2i} + \min\{\tfrac{1}{2}, 2^{-i+1}\}
\end{aligned}$$

Overall, we obtain that

$$-\log \mathbb{E}_{p',p'' \sim \mathcal{D}_i}[M(\phi(p')_i, \phi(p'')_i)] \geq i/3.$$

On the other hand $-\log M(\phi(p)_i, \phi(q)_i) = |\mathrm{proj}_{I_i}(p) - \mathrm{proj}_{I_i}(q)| \cdot 3i/|I_i|$. Therefore:

$$\begin{aligned}
&\sum_{i \in [\log(n)]} \|\phi(p)_i + \phi(q)_i\|_1 \cdot \frac{\log M(\phi(p)_i, \phi(q)_i)}{\log \mathbb{E}_{p',p'' \sim \mathcal{D}_i}[M(\phi(p')_i, \phi(p'')_i)]} \\
&\qquad \leq \sum_{i \in [\log(n)]} 2|I_i| \cdot O(|\mathrm{proj}_{I_i}(p) - \mathrm{proj}_{I_i}(q)|/|I_i|) \\
&\qquad = O(1) \cdot \sum_{i \in [\log(n)]} |\mathrm{proj}_{I_i}(p) - \mathrm{proj}_{I_i}(q)| \\
&\qquad = O(|p - q|).
\end{aligned}$$

□

7.6.2 Main construction for Lemma 126

Using the above ancillary constructions, we define the overall construction for Lemma 126:

- For each $\Delta \in \{1, 2, 4, 8, \ldots, 4k\} \cdot \frac{t}{4}$, construct ψ_Δ as embedding for $P_{*,*}$ using Lemma 128 with the corresponding parameter Δ.
- For each $p \in P_*$, let $\gamma(p) \triangleq \mathrm{proj}_{[0,1]}\left(\frac{4}{kt} \cdot (kt/2 - \sum_{j=1}^{k} D_\delta(p_j, P_{*,*}))\right)$.
- Let ϕ be an embedding of the vector $\gamma(P_*)$ using Lemma 130.

Finally, the embedding is, for each $p \in P$, for $E(p) \triangleq \left(\bigoplus_\Delta \Delta/t \cdot \sum_{j \in [k]} \psi_\Delta(p_j)\right)$:

$$\varphi(p) \triangleq k \cdot \phi(p) \oplus E(p).$$

This is viewed as WASE embedding where the coordinate blocks are ϕ_i and E (i.e., $\log n + 1$ total).

7.6.3 Proof of Lemma 126

Proof of Lemma 126. We need to show that φ is a $(O(\frac{c \cdot \widehat{\mathrm{d}}}{kt}), m')$-WASE. Fix $p, q \in \mathcal{X}^k$, for which we need to prove Equation (7.1).

First we claim that $\|\varphi(p)\| = \Omega(k)$, which is the only place where we need both ϕ and ψ. Observe that for any $p \in \mathcal{X}^k$, we either have $\gamma(p) = 1$, in which case $\|\phi(p)\| = 1$ (from Lemma 130 guarantee 1) or otherwise $\sum_j D_\delta(p_j, P_{*,*}) > kt/4$. In the latter case, let $J \subset [k]$ be the set of coordinates where $D_\delta(p_j, P_{*,*}) > 3t/16$. Then we have that $\sum_{j \in J} D_\delta(p_j, P_{*,*}) > kt/16$. Now, for each $j \in J$, there exists a $\Delta \geq 2/3 \cdot D_\delta(p_j, P_{*,*})$ such that $\|\psi_\Delta(p_j)\| \geq 1$ (using guarantee (1) of Lemma 128).

For such Δ, we have the contribution to $\|E(p)\|_1$ is at least $\Delta/t \geq 2/3t \cdot D_\delta(p_j, P_{*,*})$. Hence the total contribution to $\|\varphi(p)\|_1$ is at least $k/24$.

Now fix $p, q \in \mathcal{X}^k$ and σ be a permutation of $[k]$ such that $\mathsf{EMD}_\mathsf{d}(p,q) = \sum_j \mathsf{d}(p_j, q_{\sigma(j)})$. First we show:

$$\sum_{i \in [\log(n)]} (\|\phi(p)_i\|_1 + \|\phi(q)_i\|_1) \cdot \frac{\log M(\phi(p)_i, \phi(q)_i)}{\log \mathbb{E}_{p',p'' \sim \mathcal{D}_i}[M(\phi(p')_i, \phi(p'')_i)]} = O\left(\frac{\mathsf{EMD}_\mathsf{d}(p,q)}{t}\right). \tag{7.5}$$

To obtain that, first observe by triangle inequality, for each j, $|D_\delta(p_j, P_{*,*}) - D_\delta(q_{\sigma(j)}, P_{*,*})| \leq \mathsf{d}(p_j, q_{\sigma(j)})$, and hence $|\sum_j D_\delta(p_j, P_{*,*}) - \sum_j D_\delta(q_j, P_{*,*})| \leq \mathsf{EMD}_\mathsf{d}(p,q)$. Hence $|\gamma(p) - \gamma(q)| = O(\mathsf{EMD}_\mathsf{d}(p,q)/kt)$. By Lemma 130 guarantee 2, we obtain Equation (7.5).

Next, we analyze $E(p) = \left(\bigoplus_\Delta \Delta/t \cdot \sum_{j \in [k]} \psi_\Delta(p_j))\right)$. We show the following:

1. $\mathbb{E}_{p',p'' \sim \mathcal{D}_E}[M(E(p'), E(p''))] = O(\delta k \log k)$.
2. For any $p, q \in \mathcal{X}^k$, we have $(\|E(p)\|_1 + \|E(q)\|_1) \cdot \frac{\log(1/M(E(p),E(q)))}{\log(1/\delta)} = O\left(\frac{\mathsf{EMD}_\mathsf{d}(p,q)}{t}\right)$.

For (1), note that since $D_\delta(P_{*,*}) \geq t$, then by Lemma 128 (1), we have $\|E(*)\|_1 = \Omega(nk)$. On the other hand, by Lemma 128 (3) we have:

$$\begin{aligned}
\sum_{p',p''} \|\min\{E(p'), E(p'')\}\|_1 &= \sum_\Delta \sum_{p',p''} \left\| \tfrac{\Delta}{t} \min\{\psi(p'), \psi(p'')\} \right\|_1 \\
&\leq \sum_\Delta \sum_{p',p''} \sum_{i,j \in [k]} \tfrac{\Delta}{t} \left\| \min\{\psi(p'_i), \psi(p''_j)\} \right\|_1 \\
&\leq \sum_\Delta O(\delta n^2 k^2) \\
&\leq O(\delta n^2 k^2 \log k)
\end{aligned}$$

Overall, we have:

$$\mathbb{E}_{p',p''\sim\mathcal{D}_E}[M(E(p'),E(p''))] = 2^{\frac{\sum_{p',p''}\|\min\{E(p'),E(p'')\}\|_1}{n\cdot\|E(*)\|_1}} = O(\delta k\log k),$$

proving (1).

For (2), we note by Lemma 128 (4) that $\log(1/M(\psi_\Delta(p_j),\psi_\Delta(q_{\sigma(j)})) = O(c)\cdot\frac{\mathsf{d}(p_j,q_{\sigma(j)})}{\Delta}\cdot\log(1/\delta)$. Now we argue the following, using also Lemma 128 (2) and Jensen's inequality:

$$\begin{aligned}
&(\|E(p)\|_1+\|E(q)\|_1)\cdot\tfrac{\log(1/M(E(p),E(q)))}{\log(1/\delta)}\\
&\quad\le -\tfrac{(\|E(p)\|_1+\|E(q)\|_1)}{\log(1/\delta)}\cdot\log\left(\tfrac{\sum_\Delta\Delta/t\cdot\sum_{j\in[k]}\left(\left\|2\min\{\psi_\Delta(p_j),\psi_\Delta(q_{\sigma(j)})\}\right\|_1\right)}{(\|E(p)\|_1+\|E(q)\|_1)}\right)\\
&\quad\le \frac{\sum_\Delta\Delta/t\cdot\sum_{j\in[k]}\left(\|\psi_\Delta(p_j)\|_1+\left\|\psi_\Delta(q_{\sigma(j)})\right\|_1\right)\cdot\log\left(\frac{\left\|2\min\{\psi_\Delta(p_j),\psi_\Delta(q_{\sigma(j)})\}\right\|_1}{\|\psi_\Delta(p_j)\|_1+\left\|\psi_\Delta(q_{\sigma(j)})\right\|_1}\right)}{-\log(1/\delta)}\\
&\quad\le \sum_\Delta\Delta/t\cdot\sum_{j\in[k]}\left(\|\psi_\Delta(p_j)\|_1+\left\|\psi_\Delta(q_{\sigma(j)})\right\|_1\right)\cdot\tfrac{\log(1/M(\psi(p_j),\psi(q_{\sigma(j)})))}{\log(1/\delta)}\\
&\quad\le \sum_\Delta O(c/t)\cdot\sum_{j\in[k]}\left(\|\psi_\Delta(p_j)\|_1+\left\|\psi_\Delta(q_{\sigma(j)})\right\|_1\right)\cdot\mathsf{d}(p_j,q_{\sigma(j)})\\
&\quad= O(c/t)\cdot\sum_{j\in[k]}\left(\sum_\Delta\left(\|\psi_\Delta(p_j)\|_1+\left\|\psi_\Delta(q_{\sigma(j)})\right\|_1\right)\right)\cdot\mathsf{d}(p_j,q_{\sigma(j)})\\
&\quad\le O(c/t)\cdot\sum_{j\in[k]}(6+O(\delta\log k))\,\mathsf{d}(p_j,q_{\sigma(j)})\\
&\quad= O\left(c\right)\cdot\tfrac{\mathsf{EMD}_\mathsf{d}(p,q)}{t}.
\end{aligned}$$

Combining the above with Equation (7.5), letting $\rho \triangleq \sum_{i\in[\log(n)+1]}\frac{\|\varphi(p)_i\|_1+\|\varphi(q)_i\|_1}{\|\varphi(p)\|_1+\|\varphi(q)\|_1}\cdot\frac{\log M(\varphi(p)_i,\varphi(q)_i)}{\log\mathbb{E}_{p',p''\sim\mathcal{D}_i}[M(\varphi(p')_i,\varphi(p'')_i)]}$ and using the fact the overall norm is $\Omega(k)$, average distance is $\widehat{\mathsf{d}}$, and that $\delta < 1/k^2$, we obtain:

$$\begin{aligned}
\rho &\leq \frac{O(1)}{k} \cdot \\
&\left(k \cdot \sum_{i \in [\log(n)]} (\|\phi(p)_i\|_1 + \|\phi(q)_i\|_1) \cdot \frac{\log M(\phi(p)_i, \phi(q)_i)}{\log \mathbb{E}_{p',p'' \sim \mathcal{D}_i}[M(\phi(p')_i, \phi(p'')_i)]} + (\|E(p)\|_1 + \|E(q)\|_1) \cdot \frac{\log(1/M(E(p), E(q)))}{\log(1/\delta)} \right) \\
&\leq O\left(c \cdot \frac{\mathsf{EMD}_{\mathsf{d}}(p,q)}{k \cdot t} \right) \\
&\leq O\left(\frac{c \cdot \widehat{\mathsf{d}}}{k \cdot t} \right) \cdot \frac{\mathsf{EMD}_{\mathsf{d}}(p,q)}{\widehat{\mathsf{d}}}.
\end{aligned}$$

Therefore, φ is a $(O(\frac{c \cdot \widehat{\mathsf{d}}}{kt}), m')$-WASE as needed.

□

7.7 Edit distance embedding

In this section, we design a WASE embedding for the edit distance metric achieving poly-logarithmic approximation, captured by the following main theorem of the section.

Theorem 131. *Fix an n-point dataset P in the edit distance metric (Σ^k, ed), and let $\widehat{\mathsf{ed}} = \mathbb{E}_{p',p'' \in P}[\mathsf{ed}(p',p'')]$ be the average distance inside P. Then (P, ed) admits $(O(\log^{O(1)}(k)), 1/\operatorname{poly}(k), \operatorname{poly}(k))$-WASE. The computational efficiency on a dataset of size n is as follows: $F_p = \tilde{O}(nk^{O(1)})$ preprocessing time, $F_s = O(nk^{O(1)})$ space, and $F_q = k^{O(1)}$ query time. Also the aspect ratio is $\kappa = k^{O(1)}$.*

We note that, combining Theorem 131 together with Theorem 148 immediately implies NNS algorithm for the edit distance with similar guarantees, proving Theorem 12.

The main idea behind the proof of Theorem 131 is to embed "on average" the edit distance metric into $\mathsf{EMD}_{\ell_\infty}$, where the ℓ_∞ portion has certain "sparsity guarantees" that allow us to use the WASE construction for EMD from Lemma 126. We first state the following key lemma. Recall that $D_\delta(P)$ is the average, over $p \in P$, of the δ-th quantile distance from p to $p' \in P$.

Lemma 132. *Fix a set of n strings $P \subset \Sigma^k$, for some alphabet Σ, and let $\widehat{\mathtt{ed}} = \mathbb{E}_{p',p'' \in P}[\mathtt{ed}(p',p'')]$. Then, for any $\delta \le 1/k$, either $D_\delta(P) \le \frac{1}{3} \cdot \widehat{\mathtt{ed}}$, or (ii) there exists an average metric embedding $\varphi : \Sigma^k \to (\mathbb{R}^m)^\gamma$ where $\gamma = O(k^3 \cdot \log n)$ such that the metric $((\mathbb{R}^m)^\gamma, \mathsf{EMD}_{\ell_\infty^m})$, for $m = O(1/\delta \cdot \log^2 k)$, provides the following guarantees:*

1. *For all $p, q \in \Sigma^k$, we have $\mathsf{EMD}_{\ell_\infty^m}(\varphi(p), \varphi(q)) = O\left(\frac{\log^6 k}{k}\right) \cdot \mathtt{ed}(p,q)$; and*
2. *$D_{O(\delta)}(\varphi(P)_*) \ge \Omega\left(\frac{\widehat{\mathtt{ed}}}{k}\right)$, where $\varphi(P)_*$ is the union of all sets $\varphi(p)$, $p \in P$ (and is a considered subset of ℓ_∞^m norm).*

Before proving Lemma 132, we show why it implies Theorem 131.

Proof of Theorem 131 using Lemma 132. Let $\delta = 1/k$. If (i) of Lemma 132 holds, then, for $t = O(1/\delta)$, we embed P into $(\mathbb{N}^{O(t \log t)}, \ell_\infty)$ on average directly (see Section 7.1.3), and then embed ℓ_∞ to WASE (Theorem 118), obtaining $(O(\log\log(k)), 1/O(k), O(k))$-WASE.

If, instead $D_\delta(P) > \frac{1}{3} \cdot \widehat{\mathtt{ed}}$ and hence (ii) holds, we consider the $\mathsf{EMD}_{\ell_\infty}$ embedding φ of P, and invoke Lemma 126 with $(\mathcal{X}, \mathsf{d}) = (\mathbb{R}^m, \ell_\infty)$ and its $(O(\log\log k), O(1/m), O(m))$-WASE given by Theorem 118. In particular, let $\widehat{\mathsf{EMD}}$ be the average distance with respect to $\mathsf{EMD}_{\ell_\infty}$ and use parameter $\delta' = \min\{1/k^2, 1/\operatorname{poly}(\kappa\kappa_{\mathcal{X}} m)\} \ll \delta$. Then Lemma 126, for $\mathtt{ed}$ yields a $(c', 1/m', m')$-WASE for the EMD space, where $c' = O(\log\log k) \cdot \frac{\widehat{\mathsf{EMD}}}{k \cdot \widehat{\mathtt{ed}}/O(k)} = O(\log\log k \cdot \frac{\widehat{\mathsf{EMD}}}{\widehat{\mathtt{ed}}})$, and $m' = k^{O(1)}$.

We now use (1) further to obtain $(O(\log\log(k)\cdot\log^6(k)), 1/m', m')$-WASE for $\texttt{ed}$. The computational efficiency results immediately from the constructions. □

The rest of this section will focus on proving Lemma 132. The main idea behind the construction is to represent edit-distance as $\mathsf{EMD}_{\ell_\infty}$, where the final vectors are constructed using the Fréchet embedding using substrings of smaller length w. The embedding will be randomized and the upper bound guarantees will be on average which will then be amplified, while the lower bound will hold with the sparsity guarantees as needed.

7.7.1 Characterization of ed via substrings

We first establish some structural properties of the edit distance between two strings x, y, which we use to construct our embedding. In particular we consider all substrings of, say, x of appropriate lengths, which we organize in a "string forest" of x, termed $\mathcal{F}_x$, defined below. We then show how a certain distance between $\mathcal{F}_x, \mathcal{F}_y$ relates to $\texttt{ed}(x, y)$.

Below, we assume that k is a power of 2, by padding as well. For a string x, we use $x[i : w]$ to denote a substring of length w starting at i. Whenever such substring leaves the boundaries of string x, by convention we consider those symbols to be \$.

For fixed $x \in \Sigma^k$, we define the *string forest* $\mathcal{F}_x$ as a union of k weighted trees $\{T_i\}_{i\in[k]}$. Each T_i consists of $v_{i,j,w}$ nodes associated with the string $s_v = x[j : w]$, where w ranges over powers of 2 up to k and $j \in i + w\mathbb{N} \cap [n]$ The root is $v_{i,i,k}$. Each $v_{i,j,w}$ has a pair of children: a left child $\text{left}_a(v_{i,j,w}) = v_{i,j,w/2}$ and right child $\text{right}(v_{i,j,w}) = v_{i,j+w/2,w/2}$. One can observe that each tree has $\leq k$ leaves and the

forest has $\leq k^2$ leaves in total.We denote by $\mathcal{L}_x$ the set of leaves in $\mathcal{F}_x$.

Edge Weights. For every edge (u, v) in the forest, we have $W(u, v) \in [0, 1]$ as the weight, such that for each node v: $W(v, \text{left}(v)) + W(v, \text{right}(v)) = 1$. The weights W are (jointly) sampled from a weight distribution $\mathcal{W}$. The actual weight distribution $\mathcal{W}$ is dependent between different forests (which is naturally needed to bound the expected matching distance of the closest pair). We later condition on and then bound such dependency. For now, for the sake of definitions, we focus on a single forest. For any node $v \in \mathcal{F}_x$, we denote by $\mathcal{P}_W(v)$ the vector of weights of the edges from the node to the root. For non-leaves, we also denote by $W(v) \triangleq W(v, \text{left}(v))$.

Now fix x, y, and let $\pi : \{0,1\}^k \to \{0,1\}^k \cup \{\perp\}$ be the optimal alignment between x, y; i.e., on $i \in [n] \setminus \pi^{-1}(\perp)$, π is monotone, $x[i] = y[\pi[i]]$, and $|\pi^{-1}(\perp)| \leq \mathtt{ed}(x, y)$. We now extend π to nodes of the forest. For any node $v_{i,j,w} \in \mathcal{F}_x$, we denote by $\pi(v_{i,j,w})$ the following subset of nodes in $\mathcal{F}_y$:

$$\pi(v_{i,j,w}) \triangleq \left\{u_{i+\pi[z]-z,j+\pi[z]-z,w}\right\}_{z\in[j,j+w)} \cap \mathcal{F}_y.$$

Note that for any $u_{i',j',w} \in \pi(v_{i,j,w})$ we have that $j' - i' = j - i$ (which will be important later).

Distance Measures. Next, we define 2 distance measures. The first will apply for a fixed weight vector $W \in \mathcal{W}$. We define the following distance measure recursively:

$$\mathsf{d}^{(W)}_{\text{tree}}\left(v_{i,j,w}, \pi(v_{i,j,w})\right) \triangleq \max_{u\in\pi(v_{i,j,w})} \mathtt{ed}(v_{i,j,w}, u) + W(v_{i,j,w}, \text{parent}(v_{i,j,w})) \cdot \mathsf{d}^{(W)}_{\text{tree}}(\text{parent}(v_{i,j,w})), \pi(\text{parent}(v_{i,j,w}))),$$

where if the set $\pi(v_{i,j,w})$ is empty, we take the corresponding max to be 0.

We also define the following distance measure over $\mathcal{W}$, for $v \in \mathcal{L}_x$:

$$\mathsf{d}^{(\lambda)}_{\text{match}}(v, \pi(v))$$
$$\triangleq \mathbb{E}_{W\sim\mathcal{W}}\left[\mathsf{d}^{(W)}_{\text{tree}}(v,\pi(v)) + \mathbf{1}[\mathcal{P}_W(v) \neq \mathcal{P}_W(\pi(v))] \cdot \lambda \cdot \left(\prod_{i=1}^{|\mathcal{P}_W(v)|} \mathcal{P}_W(v)_i + \prod_{i=1}^{|\mathcal{P}_W(v)|} \mathcal{P}_W(\pi(v))_i\right)\right].$$

The $\mathsf{d}^{(\lambda)}_{\text{match}}$ distance will be used to upper bound the distance between any p, q string pairs (described in the next subsection).

We now state the following structural lemmas on the forests.

Lemma 133. *Fix $x, y \in \Sigma^k$ and an optimal alignment $\pi \in \Pi$. Then:*

$$\sum_{v\in\mathcal{L}_x} \mathsf{d}^{(W)}_{tree}(v, \pi(v)) \leq O(\log k) \cdot k \cdot \mathtt{ed}(x, y).$$

Proof. We first prove the following claim, which is a variation of the folklore statement that the edit distance between two strings can be decomposed into blocks.

Claim 134. *Fix $x, y \in \Sigma^k$ and an optimal alignment $\pi \in \Pi$. Then, for any $w < k$ and $s \in [-w, w]^k$, we have:*

$$\sum_{i\in[k]} \mathbf{1}\left[\pi[i + s_i] \neq \perp\right] \cdot \mathtt{ed}(x[i : w], y[\pi[i + s_i] - s_i : w]) = w \cdot O(\mathtt{ed}(x, y)).$$

Proof. Let $X = \pi^{-1}(\perp)$ and $Y = [k] \setminus \pi([k])$ be the unmatched positions in x, y respectively. Note that $|X|, |Y| \leq \mathtt{ed}(x, y)$.

For any fixed $i \in X$ with $\pi[i + s_i] \neq \perp$, we have that

$$\mathtt{ed}(x[i : w], y[\pi[i+s_i]-s_i : w]) \leq \mathtt{ed}(x[i : w], y[\pi[i] : w])+2|X\cap[i : w]|+2|Y\cap[\pi[i] : w]|\cdot\mathbf{1}[\pi[i] \neq \perp],$$

where, by convention if $\pi[i] = \perp$, then the distance is defined to be w. Hence we get:

$$\begin{aligned}
&\sum_{i\in[k]} \mathbf{1}\left[\pi[i+s_i] \neq \perp\right] \cdot \mathtt{ed}(x[i:w], y[\pi[i+s_i] - s_i : w]) \\
&\qquad \leq \sum_i \mathtt{ed}(x[i:w], y[\pi[i]:w]) + 2|X \cap [i:w]| + 2|Y \cap [\pi[i]:w]| \cdot \mathbf{1}[\pi[i] \neq \perp] \\
&\qquad \leq 2w \cdot (|Y| + |X|) + w \cdot \mathtt{ed}(x,y) + \sum_{i:\pi(i)\neq\perp} \mathtt{ed}(x[i:w], y[\pi[i]:w]) \\
&\qquad \leq 5w \cdot \mathtt{ed}(x,y) + \sum_{i:\pi(i)\neq\perp} 2|X \cap [i:w]]| + |Y \cap [\pi[i]:w]| \\
&\qquad \leq O(w \cdot \mathtt{ed}(x,y)).
\end{aligned}$$

□

Note that by the weighing constraint, for any $w < k$:

$$\begin{aligned}
\sum_{v\in v_{*,*,w}\cap\mathcal{F}_x} \mathtt{d}^{(W)}_{\text{tree}}(v,\pi(v)) &= \sum_{v\in v_{*,*,w}\cap\mathcal{F}_x} \max_{u\in\pi(v)} \mathtt{ed}(v,u) + \sum_{v\in v_{*,*,2w}\cap\mathcal{F}_x} \mathtt{d}^{(W)}_{\text{tree}}(v,\pi(v)) \\
&\leq k \cdot \beta^{\log_2(k/w)} \cdot O(\mathtt{ed}(x,y)) + \beta \cdot \sum_{v\in v_{*,*,2w}\cap\mathcal{F}_x} \mathtt{d}^{(W)}_{\text{tree}}(v,\pi(v)),
\end{aligned}$$

where the last part uses Claim 134. Hence, overall:

$$\sum_{v\in\mathcal{L}_x} \mathtt{d}^{(W)}_{\text{tree}}(v,\pi(v)) \leq O(\log k) \cdot k \cdot \mathtt{ed}(x,y).$$

□

Finally, we state and prove a Lemma for the $\mathtt{d}^{(\lambda)}_{\text{match}}$ which will be used to bound the EMD distance of the closest pair (the proof of lemma 132 (1) which appears in the next subsection). The Lemma contains several assumptions which will be integral part of the construction (appears later in the section as well). Before stating the Lemma, we introduce new definitions. For a level $\ell \in \{0, 1, \ldots, \log_2 k\}$, define $\mathcal{F}^\ell_x$ the nodes at level ℓ, where level 0 are the roots and level $\log_2 k$ are the leaves (i.e. $\mathcal{L}_x = \mathcal{F}^{\log_2(k)}_x$). For a node v, let $A_\ell(v)$ be v's ancestor in level ℓ. Also define $\mu(v) = \prod_{i=1}^{|\mathcal{P}_W(v)|} \mathcal{P}_W(v)_i$.

Lemma 135. *Fix $x, y \in \Sigma^k$ and an optimal alignment $\pi \in \Pi$. Fix $\gamma, \lambda \in \mathbb{R}_+$. Suppose that for any $v \in \mathcal{F}_x^\ell \cup \mathcal{F}_y^\ell$ at some level $\ell < \log_2(k)$:*

1. *For any $u \in \mathcal{F}_x^\ell \cup \mathcal{F}_y^\ell$: $\Pr_{W\sim\mathcal{W}}[W(u) \neq W(v) \mid \mathcal{P}(u) = \mathcal{P}(v)] \leq \gamma \cdot \mathsf{d}_{tree}^{(W)}(u,v)/\lambda \prod_{i=1}^{|\mathcal{P}(u)|} \mathcal{P}(u)_i$;*
2. *$W(v) \in \{0, \epsilon, 2\epsilon, \ldots, 1\}$, where $\epsilon = \Omega(1/\log(k))$; and,*
3. *$\mathcal{W}$ is such that for any $u \in \mathcal{F}_x^{\ell'} \cup \mathcal{F}_y^{\ell'}$ with $\ell' < \ell$, we have $W(u) \perp\!\!\!\perp W(v) \mid W(A_{\ell'}(v))$[2].*

Then:

$$\mathsf{d}_{match}^{(\lambda)}(v, \pi(v)) \leq O(\log^3 k) \cdot k \cdot \gamma \cdot \mathsf{ed}(x, y).$$

Proof. For a level ℓ and $v \in \mathcal{F}_x^{\geq\ell} \cup \mathcal{F}_y^{\geq\ell}$, let $\mathcal{P}^\ell(v) \triangleq W(A_\ell(v))$. Also let $\mathcal{P}^{<\ell}(v) \triangleq \mathcal{P}(A_\ell(v))$. Now fix $v \in \mathcal{L}_x$. We have:

$$\begin{aligned}
&\mathsf{d}_{\text{match}}^{(\lambda)}(v, \pi[v]) \\
&= \mathbb{E}_W \left[\mathsf{d}_{\text{tree}}^{(W)}(v, \pi[v]) + \lambda \left(\mu(v) + \mu(\pi[v])\right) \sum_{\ell=0}^{\log_2 k - 1} \mathbf{1}\left[\mathcal{P}_W^\ell(v) \neq \mathcal{P}_W^\ell(\pi[v])) \wedge \mathcal{P}_W^{<\ell}(v) = \mathcal{P}_W^{<\ell}(\pi[u])\right] \right] \\
&\leq \mathbb{E}_W \left[\mathsf{d}_{\text{tree}}^{(W)}(v, \pi[v]) \right] \\
&\qquad + \lambda \sum_{\ell=0}^{\log_2 k - 1} \mathbb{E}_W \left[\left(\mu(v) + \mu(\pi[v])\right) \mathbf{1}[\mathcal{P}_W^\ell(v) \neq \mathcal{P}_W^\ell(\pi[v]))] \mid \mathcal{P}_W^{<\ell}(v) = \mathcal{P}_W^{<\ell}(\pi[u]) \right].
\end{aligned}$$

We now sum over all leaves in the forest. The first part is easy to bound with Lemma 133, as for any choice of W, we have $\sum_{v\in\mathcal{L}_x} \mathsf{d}_{\text{tree}}^{(W)}(v, \pi(v)) \leq O(\log k) \cdot k \cdot \mathsf{ed}(x, y)$. It remains to bound $\mathsf{d}^* \triangleq \sum_{v\in\mathcal{L}_x} \lambda \sum_{\ell=0}^{\log_2 k-1} \mathbb{E}_W \left[\left(\mu(v) + \mu(\pi[v])\right) \mathbf{1}[\mathcal{P}_W^\ell(v) \neq \mathcal{P}_W^\ell(\pi[v]))] \mid \mathcal{P}_W^{<\ell}(v) = \mathcal{P}_W^{<\ell}(\pi[u]) \right]$. First, define $\mathbf{G}_W^\ell(v) \triangleq [\mathcal{P}_W^\ell(v) = \mathcal{P}_W^\ell(\pi[v]))]$, $\mathbf{B}_W^\ell(v) \triangleq [\mathcal{P}_W^\ell(v) \neq \mathcal{P}_W^\ell(\pi[v]))]$ and $\mathbf{B}_W^\ell(v, h) \triangleq [h = \mathcal{P}_W^\ell(v) \vee h \neq \mathcal{P}_W^\ell(\pi[v]))]$. We derive $\frac{\mathsf{d}^*}{\lambda}$ as follows:

[2]The notation here refers to conditional independence. I.e. $W(u)$ and $W(v)$ are independent for a fixed $W(A_{\ell'}(v))$.

$$
\begin{aligned}
\frac{\mathsf{d}^*}{\lambda} &= \sum_{v\in\mathcal{L}_x}\sum_{\ell=0}^{\log_2 k-1} \mathbb{E}_W\left[(\mu(v)+\mu(\pi[v]))\,\mathbf{1}[\mathbf{B}_W^\ell(v)] \mid \mathbf{G}_W^{<\ell}(v)\right] \\
&\le \sum_{\ell=0}^{\log_2 k-1}\sum_{u\in\mathcal{F}_x^\ell\cup\mathcal{F}_y^\ell}\sum_{\substack{v\in\mathcal{L}_x\cup\mathcal{L}_y:\\ u\in A_\ell(v)}} \mathbb{E}_W\left[\mu(v)\cdot\mathbf{1}[\mathbf{B}_W^\ell(v)] \mid \mathbf{G}_W^{<\ell}(v)\right] \\
&\le \sum_{\ell=0}^{\log_2 k-1}\sum_{u\in\mathcal{F}_x^\ell\cup\mathcal{F}_y^\ell}\sum_{\substack{v\in\mathcal{L}_x\cup\mathcal{L}_y:\\ u\in A_\ell(v)}} \Pr_W\left[\mathbf{B}_W^\ell(v) \mid \mathbf{G}_W^{<\ell}(v)\right]\cdot\mathbb{E}_W\left[\mu(v) \mid \mathbf{B}_W^\ell(v)\cap\mathbf{G}_W^{<\ell}(v)\right] \\
&= \sum_{\ell=0}^{\log_2 k-1}\sum_{u\in\mathcal{F}_x^\ell\cup\mathcal{F}_y^\ell}\sum_{\substack{v\in\mathcal{L}_x\cup\mathcal{L}_y:\\ u\in A_\ell(v)}}\sum_{h\in\{0,\epsilon,2\epsilon,\ldots,1\}} \Pr_W\left[\mathbf{B}_W^\ell(v,h) \mid \mathbf{G}_W^{<\ell}(v)\right]\cdot\mathbb{E}_W\left[\mu(v) \mid \mathbf{B}_W^\ell(v,h)\cap\mathbf{G}_W^{<\ell}(v)\right] \\
&= \sum_{\ell=0}^{\log_2 k-1}\sum_{u\in\mathcal{F}_x^\ell\cup\mathcal{F}_y^\ell}\sum_{\substack{v\in\mathcal{L}_x\cup\mathcal{L}_y:\\ u\in A_\ell(v)}}\sum_{h\in\{0,\epsilon,2\epsilon,\ldots,1\}} \Pr_W\left[\mathbf{B}_W^\ell(v,h) \mid \mathbf{G}_W^{<\ell}(v)\right]\cdot\mathbb{E}_W\left[\mu(v) \mid W(u)=h\cap\mathbf{G}_W^{<\ell}(v)\right] \\
&\le \sum_{\ell=0}^{\log_2 k-1}\sum_{u\in\mathcal{F}_x^\ell\cup\mathcal{F}_y^\ell}\sum_{\substack{v\in\mathcal{L}_x\cup\mathcal{L}_y:\\ u\in A_\ell(v)}}\sum_{h\in\{0,\epsilon,2\epsilon,\ldots,1\}} \Pr_W\left[W(u)\neq W(A_\ell[\pi(v)]) \mid \mathbf{G}_W^{<\ell}(v)\right] \\
&\qquad\qquad\cdot\mathbb{E}_W\left[\mu(v) \mid W(u)=h\cap\mathbf{G}_W^{<\ell}(v)\right] \\
&\le \sum_{\ell=0}^{\log_2 k-1}\sum_{u\in\mathcal{F}_x^\ell\cup\mathcal{F}_y^\ell}\sum_{\substack{v\in\mathcal{L}_x\cup\mathcal{L}_y:\\ u\in A_\ell(v)}}\sum_{h\in\{0,\epsilon,2\epsilon,\ldots,1\}} \frac{\gamma\cdot\mathsf{d}_{\text{tree}}^{(W)}(u,\pi[u])}{\lambda\mu(u)}\cdot\mathbb{E}_W\left[\mu(v) \mid W(u)=h\cap\mathbf{G}_W^{<\ell}(v)\right] \\
&\le \sum_{\ell=0}^{\log_2 k-1}\sum_{u\in\mathcal{F}_x^\ell\cup\mathcal{F}_y^\ell}\sum_{h\in\{0,\epsilon,2\epsilon,\ldots,1\}} \frac{\gamma\cdot\mathsf{d}_{\text{tree}}^{(W)}(u,\pi[u])}{\lambda\mu(u)}\cdot\mu(u) \\
&= \sum_{u\in\mathcal{F}_x\cup\mathcal{F}_y}\sum_{h\in\{0,\epsilon,2\epsilon,\ldots,1\}} \frac{\gamma\cdot\mathsf{d}_{\text{tree}}^{(W)}(u,\pi[u])}{\lambda} \\
&= \sum_{u\in\mathcal{F}_x\cup\mathcal{F}_y}(1/\epsilon+1)\cdot\frac{\gamma\cdot\mathsf{d}_{\text{tree}}^{(W)}(u,\pi[u])}{\lambda} \\
&= \tfrac{1}{\lambda}\cdot O(\log^3 k)\cdot k\cdot\gamma\cdot\mathsf{ed}(x,y),
\end{aligned}
$$

where the 2nd line is simply changing the count order (the inequality is since we now double count mismatches), the 4th line is from guarantee (2), the 5th is from

guarantee (3), the 7th is from guarantee (1) and 8th is since for $W \in \mathcal{W}$, the sum of $\mu(\cdot)$ over all leaf descendants of u is $\mu(u)$ (their conditions of all such leaves match), and the last line is from Lemma 133. This concludes the proof.

□

7.7.2 Proof of Lemma 132: recursive embedding

The high level idea is to reduce ed substrings to "sparse EMD over ℓ_∞", and then apply the construction from Section 7.6. We do so by iterating over w (from large to small), along the trees from the previous section, and obtaining certain sparsity guarantees for each w iteratively. In each iteration $w \to w/2$, the heart of the construction is "redistributing" ℓ_∞ distances obtained so far for w-length substrings to $w/2$-length substrings. When we reach the leafs of the string forest for some $p \in P$, we form a set using these $w = 1$ embeddings into ℓ_∞. These sets will then be interpreted as living in the $\mathsf{EMD}_{\ell_\infty}$ metric.

We will state below our main "per iteration" lemma. Before that, we introduce a new "distance" $\mathrm{a}_\infty(\cdot,\cdot)$ which is related to ℓ_∞. Namely, a^d_∞ is the space of vectors $\mathbb{R}^d$ under the "distance" a_∞:

$$\mathrm{a}_\infty(x,y) = \max_{i:x_i \cdot y_i \le 0} |x_i|.$$

We note that $\|x-y\|_\infty \ge \mathrm{a}_\infty(x,y)$. Since the inequality is one-directional, we can only use it for lower bounding ℓ_∞. The plan is to lower bound the average a_∞ and upper bound a close pair using standard ℓ_∞. We will later state some claims for a_∞ but for now the main benefit of a_∞ to keep in mind is that it is relatively easy to locally redistribute into some (unequal) distributions, as well it provides the necessary conditional independence requirements for Lemma 135. In the below, we

use sign(x) to be a binary function that outputs 1 for positive numbers and -1 for non-positives.

Lemma 136 (Per iteration guarantees)**.** *Fix a metric* $(\mathcal{X}, \mathsf{d})$ *and define the space* $\mathcal{Y} = \mathcal{X}^{\{L,R\}} \times \mathbb{R}^s$*. Fix* n*-points set* $P \subset \mathcal{Y}$*. Fix* $\epsilon \in (0, 0.25]$*. Then for any* $\delta, \eta > 0$*, there exists a randomized embedding* $\psi : \mathcal{Y} \to \mathbb{R}^{O(\log n)/(\eta\delta)}$ *and a (randomized) function* $h : \mathcal{Y} \to \{\epsilon, 2\epsilon, \ldots, 1-2\epsilon, 1-\epsilon\}$*, both depending on* $P_L, P_R, \text{sign}(P_2)$ *only, such that, letting* $\psi_L(p) \triangleq h(p) \cdot (p_2 \oplus \psi(p))$*,* $\psi_R(p) \triangleq (1-h(p)) \cdot (p_2 \oplus \psi(p))$ *and defining distance* $\mathsf{d}_{\mathcal{Y}} \triangleq \max\{\ell_1(\mathsf{d}), \mathrm{a}_\infty\}$[3]*:*

1. *For each* $p, q \in \mathcal{Y}$*, with high probability over* ψ*, we have with probability at least* $1 - \frac{1}{\epsilon^2 D_\delta^{\mathsf{d}_{\mathcal{Y}}}(p,P)} \cdot O(\frac{1}{\epsilon} \cdot (\mathsf{d}(p_L, q_L) + \mathsf{d}(p_R, q_R)) + \|p_2 - q_2\|_\infty)$ *over* h*,* $h(p) = h(q)$ *, and further:*

$$\|\psi_L(p) - \psi_L(q)\|_\infty \leq h(p) \cdot \|p_2 - q_2\|_\infty + O(\tfrac{1}{\epsilon}) \cdot (\mathsf{d}(p_L, q_L) + \mathsf{d}(p_R, q_R))$$

$$\|\psi_R(p) - \psi_R(q)\|_\infty \leq (1-h(p)) \cdot \|p_2 - q_2\|_\infty + O(\tfrac{1}{\epsilon}) \cdot (\mathsf{d}(p_L, q_L) + \mathsf{d}(p_R, q_R))$$

2. *For each* $p \in \mathcal{Y}$*, whp:*

$$D^{\mathsf{d}_{\mathcal{Y}}}_{(1+\eta)\delta}(p_L \oplus \psi_L(p), P_{*,L} \oplus \psi_L(P)) = (h(p) - O(\epsilon)) \cdot D^{\mathsf{d}_{\mathcal{Y}}}_{\delta}(p, P)$$

$$D^{\mathsf{d}_{\mathcal{Y}}}_{(1+\eta)\delta}(p_R \oplus \psi_R(p), P_{*,R} \oplus \psi_R(P)) = (1 - h(p) - O(\epsilon)) \cdot D^{\mathsf{d}_{\mathcal{Y}}}_{\delta}(p, P)$$

In particular, whp,

$$D^{\mathsf{d}_{\mathcal{Y}}}_{(1+\eta)\delta}(P_{*,L} \oplus \psi_L(P)) + D^{\mathsf{d}_{\mathcal{Y}}}_{(1+\eta)\delta}(P_{*,R} \oplus \psi_R(P)) = (1 - O(\epsilon)) \cdot D^{\mathsf{d}_{\mathcal{Y}}}_{\delta}(P).$$

[3]This means that the distance $\mathsf{d}_{\mathcal{Y}}$ is defined as the maximum between the sum of the distances of the first coordinates under d and a_∞ over the second coordinates

We prove Lemma 136 in subsequent section. We complete the main proof, of Lemma 132, assuming it.

Proof of Lemma 132 using Lemma 136. Assume that $D_\delta(P) > \widehat{\mathsf{ed}}/3$ (as we only need to prove the embedding for that case).

We now pad all strings with the same prefix \$ so that all edits are on the second half of the string and all strings are of length $k' = O(k)$ which is power of 2. Now consider the embedding into length k' substrings of every starting position split in half:

$$\mathcal{H}_{k'}(p) \triangleq \{p[i : k'/2] \oplus_\infty p[i + k'/2 : k'/2]\}_{i\in[k']}\,.$$

Note that since the edits are in the second half, we get: $D_\delta(\{\mathcal{H}_{k'}(p)\}_{p\in P}) \geq \widehat{\mathsf{ed}}/6$.

We now analyze for a fixed $p \in \mathcal{Y}$.

Let $\epsilon = 1/C\log(k)$ (for large enough C), $\eta = 1/\log k$. For $w \in \{1, 2, 4, 8, \ldots, k\}$, let $\delta_w = O(\delta \cdot (k/w)^{2\eta})$. Note that $\delta_{w/2} \geq \delta_w(1 + \eta)$. We now build a forest $\mathcal{F}_p$ where for each $v \in \mathcal{F}_p$, we build a corresponding ℓ_∞ vectors $I_p(v)$. proceeding top-to-bottom. We start with the roots v for which we set $I_p(v) = \vec{0}$. Then, for each node $v = v_{i,j,w}$, we invoke Lemma 136 on the associated substring $s_p(v)$ (split into two half-substrings), concatenated with $I_p(v)$. We use $\mathsf{d} = \mathsf{ed}$ and the corresponding δ_w, generating corresponding $I_p(\text{left}(v))$ and $I_p(\text{right}(v))$.

Eventually, when the strings are length $w = 1$, every string is embedded into k'^2 vectors $I_p(v)$ (for each leaf $v \in \mathcal{L}_p$), each of dimension $O(1/\delta\eta\cdot\log k) = O(1/\delta\cdot\log^2 k)$. To obtain high probability embedding, we repeat this process $L = O(k\log n)$ times (noting the aspect ratio of edit distance $\kappa_{\mathsf{ed}} = k$), obtaining forests $\mathcal{F}_{p,a}$ for $a \in$

$[L]$. Now, for any $\Delta \in w \cdot [k'/w]$, define $P_{w,\Delta,a}$ to be the set of points formed as a concatenation of $s_p(v)$ (partitioned to half-strings except when $w = 1$) and $I_p(v)$ (i.e., 3-dimensional product), where v ranges over

$$v \in \{v_{i,i+\Delta,w} \in \mathcal{F}_{x,a} \mid i \in [k], x \in P\}.$$

We also note here that splitting the string to 2 halves is non-contracting and at most a constant factor expanding. We now claim the following, with high probability:

(i) $\sum_{a\in[L]} \sum_{\Delta\in[k]} D_{O(\delta)}(P_{1,\Delta}) = \Omega(L \cdot \widehat{\mathtt{ed}})$. Indeed, by Lemma 136 (2), we get, for any $a \in [L]$

$$\sum_{\Delta} D_{\delta \cdot e^{2\eta \log(k')}}(P_{1,\Delta}) \geq e^{-O(\epsilon \log k)} \cdot D_{\delta}(P_{k'}) = \Omega(\widehat{\mathtt{ed}}).$$

(ii) For any $p, q \in \Sigma^{k'}$, $\mathsf{EMD}_{\ell_\infty}(\{I_p(v)\}_{v\in\mathcal{L}_{p,*}}, \{I_q(v)\}_{v\in\mathcal{L}_{q,*}}) = O\left(\frac{\log^6(k)}{k}\right) \cdot \mathtt{ed}(p, q)$.

Indeed, for any $a \in [L]$, we invoke Lemma 135, using the guarantees of Lemma 136. In particular, we condition on all whp events, define the weight distribution for each node v by the function $h(v)$ and use $\lambda = \widehat{\mathtt{ed}}$ and $\gamma = O(1/\epsilon^3)$. We set $\text{left}(v)_2 = \psi_L(v)$ and $\text{right}(v)_2 = \psi_R(v)$. Indeed Lemma 136 (1) ensures that conditioned on $\mathcal{P}(p) = \mathcal{P}(q)$, we have $\|p_2 - q_2\|_\infty \leq O(1/\epsilon) \cdot \mathtt{d}^{(W)}_{\text{tree}}(p, q)$, hence the upper bound of Lemma 136 (2) satisfies condition (1) of Lemma 135. For condition (3), we notice that multiplying by any positive constant doesn't change the sign of the entires of P_2, and hence the distribution $W(v)$ for each node is fully defined by its ancestors and (conditionally on $\mathcal{P}(v)$) independent of any non-ancestor weights from previous levels. Finally, we also use π for matching to obtain:

$$\mathbb{E}\left[\sum_{v\in\mathcal{L}_{p,a}} \|I_p(v) - I_q(\pi[v])\|_\infty \le O(1/\epsilon^3)\cdot O(\log^3 k)\cdot k\cdot \mathtt{ed}(p,q)\right] = O(\log^6 k)\cdot k\cdot \mathtt{ed}(p,q).$$

and since we sample $L > O(\log n)\cdot(\widehat{\mathtt{ed}}/\mathtt{ed}(p,q))$ trees and the max distance is $O(k)\cdot\widehat{\mathtt{ed}}$, we have the required bound whp (noting that EMD averages over $k'(k'+1)/2 = \Omega(k^2)$ leaves).

We now build the embedding $\varphi(p)$ into a set in ℓ_∞. First, for each point in $\mathcal{P}_{1,\Delta,a}$, which is a tuple of a characters in Σ (under $\mathtt{ed}$), and an ℓ_∞ vector, we replace the character with indicator vector of dimension Σ. Note that this preserves all the distances under a_∞ or ℓ_∞. Furthermore, every vector I in the set is further augmented with extra coordinates of value $(\Delta, a)\cdot k^9$. Call the resulting set $\mathcal{P}'_{1,\Delta,a}$. Then $\varphi(p)$ is the union of $\mathcal{P}'_{1,\Delta,a}$ over all (Δ, a)'s; note that we get a set of size $L\cdot k'(k'+1)/2$.

To obtain the guarantee (2) of Lemma 132, we use (i) from above and note that $\mathrm{a}_\infty(p,q) \le \|p-q\|_\infty$, as well as that the extra coordinate make sure that vectors corresponding to different (Δ, a) never match, and thus

$$D_{O(\delta)}(\varphi(p)) = \tfrac{1}{Lk}\sum_a\sum_\Delta D_{O(\delta)}(\mathcal{P}'_{1,\Delta,a}, \varphi(p)) = \Omega(\widehat{\mathtt{ed}}/k).$$

To obtain guarantee (1) of Lemma 132, we just use (ii) from above. Note that the extra coordinate added (making sure that only substrings corresponding to the same (Δ, a) match in EMD) does not increase the distance since our matching of leaves as above always matches substrings with the same $j-i$ and the same (matching) forest instance. □

7.7.3 Proof of Lemma 136

We first argue argue some facts for a_∞:

Fact 137 ($\ell_\infty \to \mathrm{a}_\infty$)**.** *There exist a mapping $f : [T] \to [-T, T]^T$ such that:*

$$\mathrm{a}_\infty(f(x), f(y)) = \mathrm{a}_\infty(f(y), f(x)) = |x - y|.$$

Proof. Set $f(x)_i = i - x$. Then $\mathrm{a}_\infty(f(x), f(y)) = |f(x)_y| = |f(y)_x| = \mathrm{a}_\infty(f(y), f(x))$. □

Fact 138 (Triangle Inequality for a_∞)**.** *Fix $x, y, z \in \mathbb{R}^n$. Then:*

$$\mathrm{a}_\infty(x, z) - \mathrm{a}_\infty(y, z) \leq \|x - y\|_\infty.$$

Proof. Fix the coordinate i where $\mathrm{a}_\infty(x, z)$ is maximized, and assume that $x_i \cdot z_i \leq 0$ (otherwise it's immediate). It is enough to prove that $|x_i| - \mathbb{1}[y_i z_i \leq 0] y_i \leq |x_i - y_i|$. If $y_i z_i \leq 0$ then it's immediate, so assume that $y_i z_i > 0$. Hence $x_i y_i \leq 0$, and then $|x_i - y_i| = |x_i| + |y_i|$. The inequality follows. □

We next show a "redistribution Lemma" for assigning weights of vectors. Again we will lower bound using a_∞ and upper bound using ℓ_∞:

Lemma 139. *We define mapping $\phi : \mathbb{R}^k \times [0, 1] \times \mathbb{R}_+ \to \mathbb{R}^k$ for any $x, y \in \mathbb{R}^k_+$, $\alpha, \beta \in [0, 1]$ and $T_x, T_y \in \mathbb{R}_+$ by setting $\phi(x, \alpha, T) = \alpha \cdot \mathrm{proj}_{[-T,T]^k}(x)$. Then:*

- $\mathrm{a}_\infty(\phi(x, \alpha, T_x), \phi(y, \beta, T_y)) = \alpha \cdot \min\{T_x, \mathrm{a}_\infty(x, y)\}$;
- $\|\phi(x, \alpha, T_x) - \phi(y, \beta, T_y)\|_\infty \leq \beta \max\{|T_x - T_y|, \|x - y\|_\infty\} + |\alpha - \beta| \cdot T_x$.

Proof. Let $x' = \mathrm{proj}_{[-T,T]^k}(x)$. Then, for (1), we have that

$$\mathrm{a}_\infty(\phi(x, \alpha, T_x), \phi(y, \beta, T_y)) = \max_{i : \alpha x'_i \cdot \beta y'_i \leq 0} |\alpha x'_i| = \alpha \cdot \max_{i : x'_i \cdot y_i \leq 0} |x'_i| = \alpha \cdot \mathrm{a}_\infty(x', y) = \alpha \cdot \min\{T_x, \mathrm{a}_\infty(x, y)\}.$$

For (2), we have that:

$$\begin{aligned}\|\phi(x,\alpha,T_x)-\phi(y,\beta,T_y)\|_\infty &= \max_i |\alpha x'_i - \beta y'_i| \\ &\le \max_i |\beta x'_i - \beta y'_i + (\alpha-\beta)x'_i| \\ &\le \max_i \beta|x'_i - y'_i| + |\alpha-\beta|\cdot|x'_i| \\ &\le \beta\cdot\max\{|T_x-T_y|, \|x-y\|_\infty\} + |\alpha-\beta|\cdot T_x.\end{aligned}$$

□

We can finally ready to prove Lemma 136.

Proof of Lemma 136. The embedding is the following. Fix any $p \in \mathcal{Y}$ denoted as $p = ((p_L, p_R), p_2)$.

First, we use the first pair of coordinates, and apply the standard Fréchet embedding into ℓ_∞ for dense metrics. In particular, sample $O(\log(n))/(\eta\delta))$ random points from P (denoted V) and map p to the following vector of distances: $\left[\frac{2}{\epsilon}\mathsf{d}(p_\chi, p'_\chi)\right]_{(p',\chi)\in V\times\{L,R\}}$. Thereafter, we embed the resulting ℓ_∞ metric into a_∞ using Fact 137, and denote by φ the resulting embedding.

We now use the last coordinate. Let $T_p = D^{\mathsf{d}_\mathcal{Y}}_\delta(p, P)$ and $p^\circ \triangleq p_2 \oplus \varphi(p)$ and $P^\circ \triangleq P_{*,2} \oplus \varphi(P)$.

We next define, for $x \in [0,1]$ $t_p(x)$ that represents the "mass" of ℓ_∞ vector to assign to the left child to keep the sparsity guarantee. Define a threshold function $m_p(x) \triangleq T_p \cdot (1-\epsilon)\cdot(1-\epsilon)^x$ Also define $L_p(t) \triangleq D^{\mathsf{d}_\mathcal{Y}}_{(1+\eta)\delta}(p_L \oplus t\cdot p^\circ, P_{*,L}\oplus P^\circ)$ and $R_p(t) \triangleq D^{\mathsf{d}_\mathcal{Y}}_{(1+\eta)\delta}(p_R \oplus (1-t)\cdot p^\circ, P_{*,R}\oplus P^\circ)$. We define $t_p(x)$, where

$$t_p(x) \triangleq \min_{t\in[0,1]:L_p(t)+R_p(t)\geq m_p(x)}\{t\}$$

We note here that any x will suffice for the sparsity guarantees we need, but for closeness between p and q, we have to be careful as several things can go wrong for a certain x. Therefore, we will pick a single x in random and use it for the embedding (for all points / query):

$$x \sim \text{Uniform}([0,1])$$

Also, to get perfect matches, we implement random rounding for $t_p(x)$ to some multiple of ϵ, using another random variable $s \sim \text{Uniform}([0,\epsilon])$(again, for all points). We use it for function g:

$$g(t) \triangleq \text{proj}_{[\epsilon,1-\epsilon]}(t+s)^4$$

and finally produce overall randomized hash $h : \mathcal{Y} \to [0,1]$, by defining $h(p)$ for $p \in \mathcal{Y}$ to be $g(t_p(x))$ rounded down to the nearest multiple of ϵ. Note that this ensures $\Pr[h(p) \neq h(p')] \leq |t_p(x) - t_{p'}(x)|/\epsilon$, as well as $|h(p) - t_p(x)| \leq \epsilon$, two properties that will be useful for our construction.

Next, we define the ψ embedding using function ϕ generated by invoking Lemma 139:

$$\psi(p) \triangleq \phi\left(p^\circ, 1, T_p\right).$$

We note here that both $h(p)$ and $\psi(p)$ are defined using a_∞ distances to P_2 only, which means they are only dependent on $P_L, P_R, \text{sign}(P_2)$, and independent of the actual entries of P_2. We continue by proving guarantees (1) and (2). First, we

[4]this means that $g(t)$ is set to ϵ or $1-\epsilon$, if $t+s$ is, respectively, below or above the range.

state a claim about some invariants which will be proven later. For $p, q \in \mathcal{Y}$, let $\mathsf{d}_{pq} \triangleq \mathsf{d}(p_L, q_L) + \mathsf{d}(p_R, q_R)$,

Claim 140. *The following invariants hold:*

1. *For any $p, q \in \mathcal{Y}$, $\Pr[h(p) \neq h(q)] = O\left(\frac{1}{\epsilon^2 T_p}\right) \cdot (O(1/\epsilon) \cdot \mathsf{d}_{pq} + \|p_2 - q_2\|_\infty)$;*
2. *For all $p \in \mathcal{Y}$: $t_p(x)$ is well defined for all $x \in [0, 1]$; and,*
3. *For all $p \in \mathcal{Y}$, with high probability:*
   (a) *$L_p(h(p)) \geq (1 - O(\epsilon))h(p) \cdot T_p$; and*
   (b) *$R_p(h(p)) \geq (1 - O(\epsilon))(1 - h(p)) \cdot T_p$.*

We now finish the proof assuming the above claim, and prove the claim afterwards.

For guarantee (1), consider any $q, p \in \mathcal{Y}$, and observe that by Claim 140 (1), we have with sufficient probability $h(p) = h(q)$. From now on we assume that. We invoke Lemma 139 (3) to obtain:

$$\begin{aligned}\|\psi(p)_L - \psi(q)_L\|_\infty &\leq h(p) \cdot \max\{|T_p - T_q|, \|p^\circ - q^\circ\|_\infty\} + |h(p) - h(q)| \cdot T_q \\ &\leq h(p) \cdot \|p_2 - q_2\|_\infty + O(1/\epsilon) \cdot \mathsf{d}_{pq},\end{aligned}$$

where the last part is due to the fact $|T_p - T_q| \leq \|p_2 - q_2\|_\infty$ by Fact 138. Similar argument for R instead of L completes part (1).

For guarantee (2), consider any $p, p' \in P$, and recall that by Lemma 139 (2), $\mathrm{a}_\infty(\psi_L(p), \psi_L(p')) = h(p) \cdot \min\{\mathrm{a}_\infty(p^\circ, p'^\circ), T_p\} = \min\{\mathrm{a}_\infty(h(p)p^\circ, p'^\circ), h(p)T_p\}$. Hence, using Claim 140 (3) for $L_p(h(p))$:

$$D^{\mathsf{d}_\mathcal{Y}}_{(1+\eta)\delta}(p_L \oplus \psi_L(p), P_L \oplus \psi_L(P)) = \min\{T_p h(p), L_p(h(p))\} = h(p) \cdot T_p \cdot (1 - O(\epsilon));$$

and similarly,

$$D^{\mathsf{d}_{\mathcal{Y}}}_{(1+\eta)\delta}(p_R \oplus \psi_R(p), P_R \oplus \psi_R(P)) = \min\{T_p(1-h(p)), R_p(h(p))\} = (1-h(p)) \cdot T_p \cdot (1-O(\epsilon)).$$

This concludes the proof. □

7.7.4 Proof of Claim 140

Before proving the claim, we introduce some ancillary claims:

First claim bounds the distance of cross-monotone bounded sets:

Claim 141. *Let $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ be two non-increasing series of scalars in $[0,1]$ satisfying:*

1. *$a_{i+1} \leq b_i$; and,*
2. *$b_{i+1} \leq a_i$.*

Then $\|a - b\|_1 \leq 2$.

Proof. We have that:

$$\sum_i |a_i - b_i| = \sum_i \max\{a_i - b_i, b_i - a_i\} \leq \sum_{i=2}^{n} \max\{b_{i-1} - b_i, a_{i-1} - a_i\} + |a_1 - b_1| \leq$$

$$\leq \sum_{i=2}^{n} (b_{i-1} - b_i + a_{i-1} - a_i) + |a_1 - b_1| \leq a_1 + b_1 - a_n - b_n + |a_1 - b_1| \leq 2.$$

□

We now extend this to continuous form:

Corollary 142. *Let $a, b \in [0,1] \to [0,1]$ be two non-decreasing functions satisfying $a(x) \in [b(x-\delta), b(x+\delta)]$. Then $\|a - b\|_1 \leq 2\delta$.* [5]

[5] $\|a - b\|_1$ denotes the functional $L_1[0,1]$, ie $\|a - b\|_1 = \int_0^1 |a(x) - b(x)| dx$.

Proof. For $i \in [1/\delta]$, define $a'(x)_i = a(\delta i - x)$ and similar with $b'(x)_i$, and notice for any $x \in [0, \delta]$, $\{a'(x)_i\}, \{b'(x)_i\}$ satisfy Claim 141 conditions and hence for any $x \in [0, \delta]$, we have $\sum_i |a'(x)_i - b'(x)_i| \leq 2$. This further implies:

$$\|a - b\|_1 = \int_0^1 |a(x) - b(x)| dx = \int_0^\delta \sum_i |a'(x)_i - b'(x)_i| dx \leq \delta \sup_{x \in [0,\delta]} \sum_i |a'(x)_i - b'(x)_i| \leq 2\delta,$$

as needed. □

The bounds the sum of child sparsity-parameters for a fixed parameter t between points with preset conditions.

Claim 143. *Fix $p, q \in \mathcal{Y}$. For any $t \in [0, 1]$: $|(L_p(t) + R_p(t)) - (L_q(t) + R_q(t))| \leq \mathsf{d}_{pq} + \|p^\circ - q^\circ\|_\infty$.*

Proof. Note that for any point $p' \in P$, we have, by triangle inequality (and using Fact 138):
$|\max\{\mathsf{d}(p_L, p'_L), a_\infty(tp^\circ, p')\} - \max\{\mathsf{d}(q_L, p'_L), a_\infty(tq^\circ, p')\}| \leq \mathsf{d}_L + t\|p^\circ - q^\circ\|_\infty$. Hence $|L_p(t) - L_q(t)| \leq \mathsf{d}_L + t\|p^\circ - q^\circ\|_\infty$. By a symmetric analysis we get $|R_p(t) - R_q(t)| \leq \mathsf{d}_R + (1-t)\|p^\circ - q^\circ\|_\infty$ and summing both sides obtain the claim. □

The next claim proves two points with small enough tree-distance satisfy the conditions above.

Claim 144. *Fix $\delta < 1$. Suppose $\mathsf{d}_{pq} \cdot 2/\epsilon + \|p_2 - q_2\|_\infty \leq \delta \cdot T_p$, then:*

1. $\frac{T_p}{T_q} \in 1 \pm 2\delta$
2. $\mathsf{d}_{pq} + \|p^\circ - q^\circ\|_\infty \leq 2\delta \cdot T_p$.

Proof. First, note that, for any $p' \in P$, we have by triangle inequality (and using Fact 138) $d_{qp'} + \mathrm{a}_\infty(q_2, p_2') \in d_{pp'} + \mathrm{a}_\infty(p_2, p_2') \pm (d_{pq} + \|p_2 - q_2\|_\infty)$. Hence $T_q \in T_p \pm 2\max\{d_{pq}, \|p_2 - q_2\|_\infty\}$, and thus $T_q \in (1 \pm 2\delta)T_p$, proving the first part.

For the second part, first note that $d_{pq} \leq \epsilon\delta/2 \cdot T_p$. Second, $\|p^\circ - q^\circ\|_\infty \leq \max\{\frac{2}{\epsilon}d_{pq}, \|p_2 - q_2\|_\infty\}$ by construction of the Fréchet embedding, and hence $\|p^\circ - q^\circ\|_\infty \leq \delta T_p$. Summing the two we get the conclusion.

□

Finally, we combine the 2 claims above to show the (unshifted) $t's$ are cross-monotonic so we can apply the first claim:

Corollary 145. *Fix $\delta < 1$. Suppose $\mathsf{d}_{pq} \cdot 2/\epsilon + \|p_2 - q_2\|_\infty \leq \delta \cdot T_p$, then, for all $x \in [0, 1]$:*

$$t_p(x) \in t_q(x \pm 4\delta/\epsilon).$$

Proof. We will show that

$$\min_{t \in [0,1]: L_p(t) + R_p(t) \geq m_p(x)} \{t\} \in \min_{t \in [0,1]: L_q(t) + R_q(t) \geq m_q(x \pm 4\delta/\epsilon)} \{t\},$$

from which the claim follows.

First notice that for all $x \in [0, 1]$: $m_p(x) = T_p(1 - \epsilon)(1 - \epsilon)^x = (1 - O(\epsilon)) \cdot T_p$. Now fix such x and t satisfying $L_p(t) + R_p(t) \geq m_p(x)$. Now we apply Claim 144 to obtain:

1. $\frac{T_p}{T_q} \in 1 \pm 2\delta$
2. $\mathsf{d}_{pq} + \|p^\circ - q^\circ\|_\infty \leq 2\delta \cdot T_p$.

Now using (2) above and Claim 143 we have on one hand:

$$|(L_p(t) + R_p(t)) - (L_q(t) + R_q(t))| \leq 2\delta \cdot T_p$$

On the other hand, using (1) above, and the fact $m_p(x) = (1 - O(\epsilon)) \cdot T_p$, we obtain:

$$m_q(x \pm 4\delta/\epsilon) = m_q(x)(1-\epsilon)^{\pm 4\delta/\epsilon} \supset m_p(x) \pm 2\delta \cdot T_p.$$

That implies the corollary. □

We are now ready to prove Claim 140.

Proof of Claim 140. Fix $\delta = \frac{1}{T_p} \cdot (2/\epsilon \cdot \mathsf{d}_{pq} + \|p_2 - q_2\|_\infty)$. For (1), we invoke Corollary 145 to obtain for all $x \in [0,1]$, $t_p(x) \in t_q(x \pm 4\delta/\epsilon)$. We now use Corollary 142 on $t_p(x), t_q(x)$ to obtain:

$$\|t_p - t_q\|_1 \leq 8\delta/\epsilon.$$

Therefore,

$$\Pr_{x,h}[h(p) \neq h(q)] = \int_0^1 \Pr_h[h(p) \neq h(q) \mid x]\, dx \leq \int_0^1 \tfrac{1}{\epsilon}|t_p(x) - t_q(x)|dx = \tfrac{1}{\epsilon}\|t_p - t_q\|_1 \leq 8\delta/\epsilon^2.$$

That concludes (1). We now derive (2). Let $P^{(\mathsf{d})} \subseteq P$ be a subset of all $p' \in P$ satisfying $a_\infty(p^\circ, p'^\circ) < T_p$, and note there are at most $\delta|P|$ points $p' \in P^{(\mathsf{d})}$ where $\mathsf{d}(p_L, p'_L) + \mathsf{d}(p_R, p'_R) < T_p$. Also let $P^{(\infty)} \triangleq P \setminus P^{(\mathsf{d})}$. Now set $t^* \triangleq \min\{1, D_{(1+\eta)\delta|P|/|P^{(\mathsf{d})}}(p_L, P^{(\mathsf{d})}_{*,L})/T_p\}$.

We now show:

Claim 146. *For any p, with probability at least $1 - n^{-10}$, $L_p(t^*) + R_p(t^*) \geq (1-\epsilon)T_p$.*

Proof. We show:

1. $L_p(t^*) \geq t^* \cdot T_p$; and

2. $R_p(t^*) \geq (1 - t^* - \epsilon) \cdot T_p$.

Indeed, for (1) above, notice that for $p' \in P^{(\infty)}$, $t^* a_\infty(p^\circ, p'^\circ) \geq t^* T_p$ by definition of $P^{(\infty)}$. Hence overall we have $L_p(t^*) = D_{(1+\eta)\delta|P|/|P^{(\mathsf{d})}}(p_L \oplus t^* \cdot p^\circ, P_{*,L} \oplus P^\circ) \geq \min\{t^* T_p, D_{(1+\eta)\delta|P|/|P^{(\mathsf{d})}|}(p_L \oplus t^* \cdot p^\circ, P^{(\mathsf{d})}_{*,L} \oplus (P^{(\mathsf{d})})^\circ)\} = t^* T_p$ by the definition of t^*.

It remains to show (2). We have again $R_p(t^*) = D_{(1+\eta)\delta}(p_R \oplus (1 - t^*) \cdot p^\circ, P_{*,R} \oplus P^\circ) \geq \min\{(1 - t^*)T_p, D_{(1+\eta)\delta|P|/|P^{(\mathsf{d})}|}(p_R \oplus (1 - t^*) \cdot p^\circ, P^{(\mathsf{d})}_{*,R} \oplus (P^{(\mathsf{d})})^\circ)\} \geq \min\{(1 - t^*)T_p, D_{(1+\eta)\delta|P|/|P^{(\mathsf{d})}|}(p_R, P^{(\mathsf{d})}_{*,R})\}$.

For $\chi \in \{L, R\}$, let $\chi'_p = D_{(1+\eta)\delta|P|/|P^{(\mathsf{d})}|}(p_\chi, P^{(\mathsf{d})}_{*,\chi})$ and hence there are at least $(1 + \eta)\delta|P|$ points p' with $\mathsf{d}(p_\chi, p'_\chi) \leq \chi'_p$ and $\mathrm{a}_\infty(p_2, p'_2), \mathrm{a}_\infty(\varphi(p), \varphi(p')) < T_p$. Also, we know that, by the definition of $T_p = D_\delta(p, P)$, there are at most $\delta|P|$ points $p' \in P$ with $\mathsf{d}(p_L, p'_L) + \mathsf{d}(p_R, p'_R) < T_p$ and $\mathrm{a}_\infty(p_2, p'_2) < T_p$.

Together, that implies that, for each of $\chi \in \{L, R\}$, there exist at least $\delta\eta|P|$ points with $\mathsf{d}(p_L, p'_L) + \mathsf{d}(p_R, p'_R) \geq T_p$, $\mathsf{d}(p_\chi, p'_\chi) \leq \chi'_p$, and $\mathrm{a}_\infty(\varphi(p), \varphi(p')) < T_p$. We would like to argue that the Fréchet embedding φ samples suitable l, r from P from these $\delta\eta|P|$-sized sets, but can't do this directly since we've already conditioned on $\varphi(p)$. To address this issue, we think of the set V of points sampled for φ as union of two equal sized sets V_L, V_R, and let φ_{V_χ} correspond to φ on coordinates corresponding only to V_χ. Then, we immediately also have that for each of $\chi \in \{L, R\}$, there exist at least $\delta\eta|P|$ points with $\mathsf{d}(p_L, p'_L) + \mathsf{d}(p_R, p'_R) \geq T_p$, $\mathsf{d}(p_\chi, p'_\chi) \leq \chi'_p$, and $\mathrm{a}_\infty(\varphi_{V_\chi}(p), \varphi_{V_\chi}(p')) < T_p$. Hence, for $\chi = L$, with probability at least $1 - n^{-10}/2$, the sample V_R includes a point $l \in V_R$ satisfying:

- $\mathrm{a}_\infty(\varphi_{V_L}(p), \varphi_{V_L}(l)) < T_p$,

- $\mathsf{d}(p_L, l_L) + \mathsf{d}(p_R, l_R) \geq T_p$,
- $\mathsf{d}(p_L, l_L) \leq L'_p$;

and similarly, with probability at least $1 - n^{-10}/2$, there exists $r \in V_L$ s.t.:

- $\mathrm{a}_\infty(\varphi_{V_R}(p), \varphi_{V_R}(r)) < T_p$,
- $\mathsf{d}(p_L, r_L) + \mathsf{d}(p_R, r_R) \geq T_p$,
- $\mathsf{d}(p_R, r_R) \leq R'_p$.

From the first condition, we have that $\frac{2}{\epsilon}|\mathsf{d}(p_R, r_R) - \mathsf{d}(l_R, r_R)| < T_p$. This further implies that $\mathsf{d}(r_R, l_R) \leq R'_p + \epsilon/2T_p$.

We now argue that $\mathsf{d}(r_R, l_R) \geq (1 - \epsilon/2)T_p - L'_p$. Indeed, similarly to above, $\frac{2}{\epsilon}|\mathsf{d}(p_R, l_R) - \mathsf{d}(r_R, l_R)| < T_p$ and hence

$$\mathsf{d}(r_R, l_R) \geq \mathsf{d}(p_R, l_R) - \epsilon/2T_p \geq T_p - L'_p - \epsilon T_p/2.$$

Overall, we obtain that $R'_p + \epsilon T_p \geq (1 - \epsilon/2)T_p - L'_p$ and hence we conclude that $L'_p + R'_p \geq (1 - \epsilon)T_p$.

Now if $t^* = 1$ we are done since we just need that $R_p(t^*)$ is non-negative. Otherwise, by definition of t^*: $L'_p = t^* T_p$ and hence $L_p(t^*) \geq L'_p$. Also we recall $R_p(t^*) \geq R'_p$, and we obtain overall:

$$R_p(t^*) \geq \min\{(1 - t^*)T_p, R'_p\} \geq \min\{(1 - t^*)T_p, T_p - L'_p - \epsilon T_p\} \geq T_p(1 - t^* - \epsilon).$$

This concludes the proof. □

We have shown such $t^* \in [0, 1]$ exists and is well defined for threshold $m_p(1) = (1 - \epsilon)T_p$ and hence also for $m_p(x)$ for $x < 1$. This proves part (2).

To complete the proof of (3), we use Claim 146 again and notice that all shifts $\{s(x) : x \in S_1\} \subseteq [-\epsilon, \epsilon]$, hence can change the bound by at most ϵT_p.

□

7.8 Building NNS and DDH from WASE

In this section, we show how to use WASE to build a NNS data structure. In fact we show something slighly more nuanced: that we can construct DDH. In particular, we show the following lemma.

Lemma 147 (WASE $\Rightarrow$ DDH)**.** *Fix a metric $(\mathcal{X}, \mathsf{d})$ that admits a (c, δ, m)-WASE. Also let $\kappa, \kappa_{\mathcal{X}} > 1$ be the aspect ratio of WASE and $\mathcal{X}$ respectively. Fix any $\tau < \min\{\delta, 1/(m\kappa\kappa_{\mathcal{X}})^{O(1)}\}$.*

Let $P \subset \mathcal{X}$ be an n-point subset. Then for any $r > 0$ such that P has no ball of size $\geq \tau n/2$ and radius ξcr, P admits a (r, τ^2)-DDH, for a suitable universal constant $\xi \geq 1$.

Furthermore, if the query time/space/preprocessing time of the WASE is $(F_q, n \cdot F_s, n \cdot F_p)$, then the query time/space/preprocessing time of DDH is $(F_q, n \cdot F_s, n \cdot F_p) \cdot \mathrm{poly}(m\kappa\kappa_{\mathcal{X}} \cdot \log n)$.

Before continuing, we show how the above implies a NNS data structure.

Theorem 148 (NNS from WASE)**.** *Fix a metric $(\mathcal{X}, \mathsf{d})$, and an n-point set $P \subset \mathcal{X}$ such that any of its subsets admits a (c, δ, m)-WASE. Let $\kappa, \kappa_{\mathcal{X}} > 1$ be the aspect ratio of WASE and $\mathcal{X}$ respectively. Also, suppose the query time/space/preprocessing time of the WASE is $(F_q, n \cdot F_s, n \cdot F_p)$. Then, for any fixed $\epsilon > 0$, we can build an NNS data structure with:*

- $O(c/\epsilon)$ *approximation;*
- $O(n^{\epsilon} \cdot \text{poly}(m\kappa\kappa_{\mathcal{X}}/\delta \cdot \log n) \cdot F_q)$ *query time;*
- $O(n^{1+\epsilon} \cdot \text{poly}(m\kappa\kappa_{\mathcal{X}}/\delta \cdot \log n) \cdot F_s)$ *space;*
- $O(n^{1+\epsilon} \cdot \text{poly}(m\kappa\kappa_{\mathcal{X}}/\delta \cdot \log n) \cdot F_p)$ *preprocessing time.*

Proof. Suppose we need to build NNS data structure that, for a query q which has a point $p \in P$ within distance $\epsilon r/2$, finds a point $p' \in P$ within distance $O(cr)$.

The algorithm is essentially the one from [133; 24]. In particular we build $L = O(n^{\epsilon})$ random trees drawn as follows. Let $\tau = \min\{\delta, 1/(m\kappa\kappa_{\mathcal{X}})^{O(1)}\}$.

Each tree is build on the dataset P. Every node v corresponds to a subset of P, termed P_v. The root corresponds to the entire P, and for every internal node v, the children's sets are a disjoint partition of P_v. There are two types of internal nodes: ball, and partition.

The construction of the tree proceeds as follows, starting with the root node with set $P_v = P$. First of all, if $|P_v| \leq O(1/\tau)$, then we stop and form a leaf. If, for the current P_v, there exists a ball of radius ξcr with $\tau|P_v|/2$ points (for constant ξ from Lemma 147), the node is a ball node: we extract the ball of radius $2\xi cr$ (forming a leaf child) and another child with the remainder on which we recurse. Otherwise, if the current set P_v contains no ball of radius ξcr containing $\tau|P_v|/2$ points, then we apply Lemma 147 on P_v to obtain (r, τ^2)-DDH and apply DDH to split P_v into a number of subsets. Each of the resulting subsets forms a child of v on which we recurse.

For a query q, we search the tree as follows. For a partition node, we apply the corresponding DDH to q and recurse on the corresponding child. For a ball node, we

check if an (arbitrary) point from the ball is within distance $4\xi cr$ from q, and if not, recurse on the "remainder" child. Once we reach a (non-ball) leaf, we compare the query against all points stored there.

The correctness follows from Lemma 147. In particular, each tree has a probability of at least $n^{-\epsilon}$ of finding a point within distance cr from q (if a r-near point p exists).

For time/space complexity, we note that the depth of the tree is at most $O(1/\tau \cdot \log n)$. □

7.8.1 Construction of the DDH, using a key lemma

We overload the notation for the partition $\mathcal{R}$ to improve readability. Namely, $\mathcal{R}(p)$ refers both to the name of the bucket, as well the set of points that collide with p; the difference will be clear from the context.

Proof of Lemma 147. Our construction is similar to the construction in Lemma 108. In particular, we build a partition tree of P. Every node v corresponds to a subset of P, termed P_v. The root corresponds to the entire P, and for every internal node v, the children's sets are a disjoint partition of P_v.

In particular, as long as a node v has $|P_v| > \tau n$, we partition it further. Otherwise, we leave the node as a leaf, which corresponds to a bucket of the hash function.

For given dataset $P \subseteq \mathcal{X}$, consider some query q and its near neighbor $p \in P$. We use the following key lemma (proven later in this section):

Lemma 149 (LSH partition for WASE)**.** *Fix a metric $(\mathcal{X}, \mathsf{d})$, and let $P \subset \mathcal{X}$ be an n point metric that admits a (c, δ, m)-WASE. Let $\kappa, \kappa_{\mathcal{X}} > 1$ be the aspect ratio of WASE and $\mathcal{X}$ respectively, and suppose $\delta < 1/\kappa^2\kappa_{\mathcal{X}}$.*

There is a randomized partition $\hat{\mathcal{R}}$ *such that, for any* $p \in P, q \in \mathcal{X}$*, if* $\rho \triangleq \zeta \cdot c \cdot \frac{\mathsf{d}(p,q)}{\mathbb{E}_{p',p'' \in P}[\mathsf{d}(p',p'')]} < 1$ *for some fixed constant* $\zeta > 1$*, then:*

$$\Pr_{\hat{\mathcal{R}}}\left[\hat{\mathcal{R}}(p) = \hat{\mathcal{R}}(q)\right] \cdot E_{\hat{\mathcal{R}}}\left[\left(\max\left\{\delta, \frac{|P \cap \hat{\mathcal{R}}(p)|}{|P|}\right\}\right)^{-\rho} \mid \hat{\mathcal{R}}(p) = \hat{\mathcal{R}}(q)\right] \geq 1. \quad (7.6)$$

Furthermore, $E_{\hat{\mathcal{R}}}\left[\frac{|P \cap \hat{\mathcal{R}}^{-1}(\hat{\mathcal{R}}(p))|}{|P|}\right] \leq 1 - \mathrm{poly}(1/m\kappa\kappa_{\mathcal{X}})$.

We now proceed to proving the DDH property for some pair p, q at distance $\mathsf{d}(p,q) \leq r$. Define $\rho = \frac{\mathsf{d}(p,q)}{r}$. Define $\mathcal{R}_P^{tree}$ to be the partition induced by the tree from the algorithm when starting from the set P.

For the 1st DDH property (p_1), we need to prove that $\Pr[\mathcal{R}_P^{tree}(p) = \mathcal{R}_q^{tree}] \geq \tau^\rho$. We use induction, similarly to how it is used in Lemma 108. In particular, we show that for any $P' \subseteq P$ that contains the near neighbor p, we have: if we start the partitioning procedure from P' (containing p), then p, q collide in one of the leafs (buckets) generated with probability at least $\min\{1, |P'|^{-\rho} \cdot \tau^{2\rho} n^\rho\}$.

The induction is immediately true if $|P'| \leq \tau n$ (as that node is not partitioned further and becomes a bucket itself). Now let's prove the inductive step for $|P'| \geq \tau n$. Let $\widehat{\mathsf{d}} \triangleq \mathbb{E}_{p',p'' \in P'}[\mathsf{d}(p',p'')]$. Note that by the assumption of no dense ball, we have $\widehat{\mathsf{d}} \geq \xi cr/2$. Hence, $\zeta \cdot c \cdot \frac{\mathsf{d}(p,q)}{\widehat{\mathsf{d}}} \leq \rho < 1$ (for $\xi = 2\zeta$), and hence we can apply Lemma 149 to the node corresponding to P'.

Thus, for $\mathcal{R}$ being the partition employed in node with set P', where $|P'| \geq \tau n$:

$$\begin{aligned}
&\Pr_{\mathcal{R}^{tree}_{P'}}[\mathcal{R}^{tree}_{P'}(q) = \mathcal{R}^{tree}_{P'}(p)] \\
&= \Pr_{\mathcal{R}}[\mathcal{R}(p) = \mathcal{R}(q)] \cdot \Pr_{\mathcal{R}^{tree}_{P'}}[\mathcal{R}^{tree}_{P''}(q) = \mathcal{R}^{tree}_{P''}(p) \mid \ P'' = \mathcal{R}(q) \cap P' \text{ and } p \in P''] \\
&\geq \Pr[\mathcal{R}(p) = \mathcal{R}(q)] \cdot \mathbb{E}_{\mathcal{R}}[\min\{1, |P''|^{-\rho} \cdot \tau^{2\rho} n^{\rho}\} \mid \ P'' = \mathcal{R}(q) \cap P' \text{ and } p \in P''] \\
&= |P'|^{-\rho} \cdot \tau^{2\rho} n^{\rho} \cdot \Pr[\mathcal{R}(p) = \mathcal{R}(q)] \cdot E_{\mathcal{R}}\left[\left(\max\left\{\tau^2 n/|P'|, \frac{|P' \cap \mathcal{R}(p)|}{|P'|}\right\}\right)^{-\rho} \mid \ \mathcal{R}(p) = \mathcal{R}(q)\right] \\
&\geq |P'|^{-\rho} \cdot \tau^{2\rho} n^{\rho},
\end{aligned}$$

where the second step used the inductive hypothesis, and the last one uses Lemma 149 with $\delta = \tau^2 n/|P'| \leq \tau$. For $P' = P$, we get $\Pr[\mathcal{R}^{tree}_{P}(p) = \mathcal{R}^{tree}_{q}] \geq \tau^{2\rho}$.

The 2nd DDH property (p_2) follows immediately from the fact that we partition any node of size more than τn.

We now show that the depth of the tree is at most $O(\text{poly}(m\kappa\kappa_{\mathcal{X}} \cdot \log n)$. Indeed, consider any point $p \in P$. Using the second part of Lemma 149, we obtain that, after $h = \text{poly}(m\kappa\kappa_{\mathcal{X}})$ levels, p's expected bucket size is $\leq n/2$. After $O(h \log n)$ levels, the bucket size is $(1 - \Omega(1))^{O(\log n)} \cdot n < \tau n$ with high-probability. Hence no point p can reach the depth of more than $O(h \log n)$. □

We prove Lemma 149 in the rest of this section.

7.8.2 Triangle inequality via padded vectors

We need to develop a certain triangle inequality under the embedded vectors with respect ot the similarity M. In particular, the equivalent of the following inequality that holds in any metric space (by triangle inequality): for any metric (P, d) and

point p,

$$\mathbb{E}_{p' \in P}\, \mathsf{d}(p, p') \geq \tfrac{1}{2}\, \mathbb{E}_{p', p'' \in P}\, \mathsf{d}(p', p'').$$

We manage to obtain the equivalent of the above for the similarity M once we "pad" the vectors. In particular, define the operator $\mathrm{pad}_T(x) : \mathbb{R}_+^m \to \mathbb{R}_+^{m+1}$ such that $\mathrm{pad}_T(x)_i = x_i$ and $\mathrm{pad}_T(x)_{m+1} = \max\{0, T - \|x\|_1\}$.

We prove the following lemma on padded vectors. The first part captures the effect of padding vectors on the similarity M between two fixed vectors. The second part is the equivalent of the above metric inequality.

Lemma 150. *Fix $\varphi \in (\mathbb{R}_+^m)^n$ with $S = \max_{i \in [n]} \|\varphi_i\|_1$. Then, for any $x, y \in \mathbb{R}_+^m$, there's a large enough constant $\zeta > 1$ such that:*

1. *for any $T \geq \|x\|_1, \|y\|_1$, we have $-\log M(pad_T(x), pad_T(y)) \leq -\log M(x, y) \cdot \zeta^{\frac{\|x\|_1 + \|y\|_1}{T}}$; and,*
2. *there exists $T \in [\|x\|_1, S]$ such that $-\log E_{a \in \varphi}[M(pad_T(x), pad_T(a))] \geq -\log E_{a,b \sim \mathcal{D}}[M(a, b)] \cdot \frac{\|x\|_1}{\zeta T}$.*

where $\mathcal{D}$ is a distribution over φ^2 such that:

$$\mathcal{D}(a, b) \triangleq \frac{\|a\|_1 + \|b\|_1}{2n \cdot \|\varphi\|_1}.$$

To prove it we use the following statements:

Fact 151. *Fix $\varphi \in (\mathbb{R}_+^m)^n$. Then for any $x \in \mathbb{R}_+^m$, we have:*

$$-\log(E_{a \in \{\varphi_1, \ldots \varphi_n\}}[\|\min\{x, a\}\|_1]) \geq -\tfrac{1}{2} \cdot \left(\log \mathbb{E}_{a,b \in \{\varphi_1, \ldots \varphi_n\}^2}[\|\min\{a, b\}\|_1] + \log \|x\|_1\right).$$

Proof. Wlog, we can assume that $a \leq x$ for all $a \in \{\varphi_1, \dots \varphi_n\}$. We need to prove that:

$$E_{a\in\{\varphi_1,\dots\varphi_n\}}[\|a\|_1] \leq \sqrt{\|x\|_1}\sqrt{E_{a,b\in\{\varphi_1,\dots\varphi_n\}^2}[\|\min\{a,b\}\|_1]}$$

For fixed $i \in [m]$, and x_i the i^{th} coordinate of x, we have that:

$$E_{a,b\in\{\varphi_1,\dots\varphi_n\}^2}[\min\{a_i,b_i\}] = x_i \cdot E_{a,b\in\{\varphi_1,\dots\varphi_n\}^2}[\min\{a_i/x_i, b_i/x_i\}] \geq x_i \left(E_{a\in\{\varphi_1,\dots\varphi_n\}}[a_i/x_i]\right)^2.$$

Hence:

$$E_{a\in\{\varphi_1,\dots\varphi_n\}}[\|a\|_1] \leq \sum_i \sqrt{x_i}\sqrt{E_{a,b\in\{\varphi_1,\dots\varphi_n\}^2}[\min\{a_i,b_i\}]},$$

and using Cauchy-Swartz:

$$\leq \sqrt{\sum_i x_i \cdot \sum_i E_{a,b\in\{\varphi_1,\dots\varphi_n\}^2}[\min\{a_i,b_i\}]} = \sqrt{\|x\|_1} \cdot \sqrt{E_{a,b\in\{\varphi_1,\dots\varphi_n\}^2}[\|\min\{a,b\}\|_1]},$$

as needed. □

Fact 152. *For any $x, y \in \mathbb{R}^m_+$ and $\alpha \leq 1$, we have $M(x/\alpha, y) \geq \alpha M(x, y)$.*

Proof.

$$M(x/\alpha, y) = \frac{2\min\{x/\alpha, y\}}{\|x/\alpha\|_1 + \|y\|_1} \geq \frac{2\min\{x,y\}}{\|x\|_1/\alpha + \|y\|_1} \geq \alpha M(x,y).$$

□

Claim 153. *Fix $\varphi \in (\mathbb{R}^m_+)^n$ with $\|\varphi\|_1 = n$. Then for any $x \in \mathbb{R}^m_+$ we have:*

$$-\log(\min\{\|x\|_1, 1\} \cdot E_{a\in\varphi}[M(pad_{\|x\|_1}(x), pad_{\|x\|_1}(a))]) \geq -\tfrac{1}{2} \cdot \log E_{a,b\sim\mathcal{D}}[M(a,b)],$$

where $\mathcal{D}$ is a distribution over φ^2 such that:

$$\mathcal{D}(a,b) \triangleq \frac{\|a\|_1 + \|b\|_1}{2n \cdot \|\varphi\|_1}.$$

Proof. First, by the above claim $-\log(\|x\|_1 \cdot E_{a\in\varphi}[M(x,a)]) \geq -\log(E_{a\in\varphi}[M(x/\|x\|,a)])$ for $\|x\|_1 < 1$. Hence it suffices to show for $\|x\|_1 \geq 1$. Now, notice on one hand:

$$E_{a\in\varphi}[M(\mathrm{pad}_{\|x\|_1}(x),\mathrm{pad}_{\|x\|_1}(a))] = E_{a\in\varphi}[\tfrac{\|\min\{x,a\}\|_1}{\|x\|_1}]$$

and on the other hand:

$$E_{a,b\sim\mathcal{D}}[M(a,b)] = \sum_{a,b} \tfrac{2\|\min\{a,b\}\|_1\cdot(\|a\|_1+\|b\|_1)}{(\|a\|_1+\|b\|_1)\cdot 2n^2} = E_{a,b\in\varphi^2}[\|\min\{a,b\}\|_1].$$

We invoke Fact 151 to conclude:

$$-\tfrac{1}{2}\cdot\log E_{a,b\sim\mathcal{D}}[M(a,b)] \leq \tfrac{1}{2}\log\|x\|_1 - \log E_{a\in\varphi}[\|\min\{x,a\}\|_1] = -\log E_{a\in\varphi}[\tfrac{\|\min\{x,a\}\|_1}{\|x\|_1}],$$

and thus:

$$-\log E_{a\in\varphi}[M(\mathrm{pad}_{\|x\|_1}(x),\mathrm{pad}_{\|x\|_1}(a))] \geq -\tfrac{1}{2}\cdot\log E_{a,b\sim\mathcal{D}}[M(a,b)],$$

as needed. □

Using the above, we prove Lemma 150:

Proof of Lemma 150. Recall that (1) asks to prove:

$$-\log M(\mathrm{pad}_T(x),\mathrm{pad}_T(y)) \leq -\log M(x,y)\cdot O\left(\frac{\|x\|_1+\|y\|_1}{T}\right)$$

For any $T \geq \|x\|_1, \|y\|_1$, we have:

$$\begin{aligned}
-\log M(\mathrm{pad}_T(x),\mathrm{pad}_T(y)) &\leq \log \tfrac{T}{\|\min\{x,y\}\|+T-\max\{\|x\|,\|y\|\}} \\
&\leq \log \tfrac{T}{T-(1-M(x,y))\max\{\|x\|,\|y\|\}} \\
&\leq (1-M(x,y))\cdot\tfrac{\max\{\|x\|,\|y\|\}}{T}.
\end{aligned}$$

If $M(x,y) \leq 1/2$, then:

$$-\log M(\text{pad}_T(x), \text{pad}_T(y)) \leq \log 1/M(x,y) \cdot \tfrac{\max\{\|x\|,\|y\|\}}{T}.$$

If $M(x,y) = 1 - z$ for some $z < 1/2$, then:

$$\begin{aligned} -\log M(\text{pad}_T(x), \text{pad}_T(y)) &\leq z \cdot \tfrac{\max\{\|x\|,\|y\|\}}{T} \\ &\leq O(\log 1/(1-z)) \cdot \tfrac{\max\{\|x\|,\|y\|\}}{T} \\ &\leq -\log M(x,y) \cdot O(\tfrac{\|x\|+\|y\|}{T}). \end{aligned}$$

This completes part (1).

For (2), let $s = E_{a,b\sim\mathcal{D}}[M(a,b)]$. We distinguish between two cases:

High norm case. Suppose $\|x\|_1 \geq s^{1/16}$. Then we set $T = \|x\|_1$. We now invoke Claim 153 to obtain:

$$-\log E_{a\in\varphi}[M(\text{pad}_T(x), \text{pad}_T(a))] \geq -\tfrac{1}{8}\log s + \min\{0, \log(\|x\|_1)\} \geq -\tfrac{1}{16}\log s.$$

Low norm case. Now, suppose $\|x\|_1 < s^{1/16}$. We set $T = S$. This gives us

$$\begin{aligned} E_{a\in\varphi}\left[M(\text{pad}_T(x), \text{pad}_T(a))\right] &\leq E_{a\in\varphi}\left[\tfrac{\min\{\|x\|_1,\|a\|_1\}+T-\max\{\|x\|_1,\|a\|_1\}}{T}\right] \\ &\leq 1 - E_{a\in\varphi}\left[\tfrac{\max\{\|x\|_1,\|a\|_1\}-\min\{\|x\|_1,\|a\|_1\}}{T}\right]. \end{aligned}$$

Since $\max\{\|x\|_1, \|a\|_1\} - \min\{\|x\|_1, \|a\|_1\} \geq \|a\|_1 - \|x\|_1$ and $E_{a\in\varphi}\|a\|_1 = 1$, we obtain:

$$-\log E_{a\in\varphi}[M(\text{pad}_T(x), \text{pad}_T(a))] \geq -\log(1 - \tfrac{1-\|x\|_1}{T}) = \Omega(\tfrac{1-\|x\|_1}{T}).$$

Therefore it's suffice to show for $r = \|x\|_1 \leq s^{1/16}$:

$$\tfrac{1-r}{r} = \Omega(\log(1/s)).$$

Indeed, for any any $0 < r \leq 1$, we have $\frac{1-r}{r} = \Omega(\log(1/r))$ and hence the above inequality verifies.

□

7.8.3 LSH partition for WASE: proof of key Lemma 149

Fact 154. *For any vectors $a, b \in \mathbb{R}^m_+$, we have $M(a,b)^2 \leq J(a,b) \leq M(a,b)$.*

We first argue the following lemma, that argues that adding extra padding to vectors does not hurt the performance of LSH decomposition. In particular, while in general, padding increases the standard calculation of the LSH exponent ρ, it does not affect the inequality from Eqn. (7.6).

Lemma 155. *For $\Delta > 0$, let $f_\Delta : \mathbb{R}^m_+ \to \mathbb{R}^m_+$ be such that $f_\Delta(x)_i = x_i$ whenever $i < m$ and $f_\Delta(x)_m = \max\{0, x_m - \Delta\}$. Now fix $\psi \in (\mathbb{R}^m_+)^n$ and $s, t \in R^m_+$. Fix $\rho, \Delta \geq 0$ such that $\Delta \leq s_m, t_m$. Suppose that for a min-wise hash space partition $\mathcal{R}$, we have for some $\delta \geq 0$:*

$$\Pr_{\mathcal{R}}[\mathcal{R}(f_\Delta(s)) = \mathcal{R}(f_\Delta(t))] \cdot E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|f_\Delta(\varphi) \cap \mathcal{R}(f_\Delta(s))|}{n}\right\}\right)^{-\rho} \mid \mathcal{R}(f_\Delta(s)) = \mathcal{R}(f_\Delta(t))\right] \geq 1.$$

Then,

$$\Pr_{\mathcal{R}}[\mathcal{R}(s) = \mathcal{R}(t)] \cdot E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|\varphi \cap \mathcal{R}(s)|}{n}\right\}\right)^{-\rho} \mid \mathcal{R}(s) = \mathcal{R}(t)\right] \geq 1.$$

Proof. Let $g_\Delta : \mathbb{R}^m_+ \to \mathbb{R}^{m+1}_+$ be such that $g_\Delta(x)_i = f_\Delta(x)_i$ whenever $i \leq m$ and $g_\Delta(x)_{m+1} = \min\{\Delta, x_m\}$; note that $g_\Delta(x)_m + g_\Delta(x)_{m+1} = x_m$. Then notice that for any Δ and any φ, s, t we have the same distribution of points partitions between φ, s, t and $g_\Delta(\varphi, s, t)$, therefore it suffices to show:

$$\Pr_{\mathcal{R}}\left[\mathcal{R}(g_\Delta(s)) = \mathcal{R}(g_\Delta(t))\right] \cdot E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|g_\Delta(\varphi) \cap \mathcal{R}(g_\Delta(s))|}{n}\right\}\right)^{-\rho} \mid \mathcal{R}(g_\Delta(s)) = \mathcal{R}(g_\Delta(t))\right] \geq 1.$$

Now define $M = \max\{g_\Delta(s), g_\Delta(t)\}$, and let $\mathbf{A}$ be the event that $h(M) = m+1$ where $h : \mathbb{R}_+^* \to \mathbb{N}$ is the minwise hash value of $\mathcal{R}$. Now notice $g_\Delta(s)_{m+1} = g_\Delta(t)_{m+1} = \Delta$ and hence $\Pr_{\mathcal{R}}\left[\mathcal{R}(g_\Delta(s)) = \mathcal{R}(g_\Delta(t)) \mid \mathbf{A}\right] \cdot E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|g_\Delta(\varphi)\cap\mathcal{R}(g_\Delta(s))|}{n}\right\}\right)^{-\rho} \mid \mathbf{A} \cap \mathcal{R}(g_\Delta(s)) = \mathcal{R}(g_\Delta(t))\right] \geq 1 \cdot 1 = 1$. Therefore, it suffices to condition on $\neg\mathbf{A}$. Then, by definition of g_Δ, we have

$$\Pr_{\mathcal{R}}\left[\mathcal{R}(g_\Delta(s)) = \mathcal{R}(g_\Delta(t)) \mid \neg\mathbf{A}\right] = \Pr_{\mathcal{R}}\left[\mathcal{R}(f_\Delta(s)) = \mathcal{R}(f_\Delta(t))\right].$$

Similarly, we have that:

$$E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|g_\Delta(\varphi) \cap \mathcal{R}(g_\Delta(s))|}{n}\right\}\right)^{-\rho} \mid \neg\mathbf{A} \cap \mathcal{R}(g_\Delta(s)) = \mathcal{R}(g_\Delta(t))\right]$$

$$= E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|f_\Delta(\varphi) \cap \mathcal{R}(f_\Delta(s))|}{n}\right\}\right)^{-\rho} \mid \mathcal{R}(f_\Delta(s)) = \mathcal{R}(f_\Delta(t))\right].$$

That implies the claim. □

Proof of Lemma 149. We first introduce some notation. Let $\hat{\mathsf{d}} = \mathbb{E}_{p',p''\in\mathcal{D}_i}\mathsf{d}(p',p'')$. Let φ be a (c,m)-WASE for P. We can rescale it so that all coordinates are integers, and hence $\|\varphi(x)\|_1 \leq 2m\kappa$ for all x.

Define $\sigma_i \triangleq \mathbb{E}_{p',p''\sim\mathcal{D}_i}[M(\varphi(p')_i, \varphi(p'')_i)]$, and note that $l_i = \log 1/\sigma_i$. Wlog, we can assume that $\sigma_i \neq 1$ — otherwise $M(\varphi(p)_i, \varphi(q)_i) = 1$ for all p, q, and hence we can drop this outer coordinate i altogether (and this can decrese the distortion of WASE). There exists two points $p', p'' \in P$ with $M(\varphi(p')_i, \varphi(p'')_i) < 1$, and hence

$M(\varphi(p')_i, \varphi(p'')_i) \leq 1 - 1/2\kappa$. Then, by the definition of WASE (applied to points p', p''):

$$\log 1/\sigma_i \geq \frac{1}{2\kappa} \cdot \frac{\hat{\mathsf{d}}}{c\mathsf{d}(p',p'')} \cdot \log(1 - 1/2\kappa) \geq \Omega(\frac{1}{\kappa_{\mathcal{X}}\kappa^2}).$$

Finally, we set $S = O(m\kappa \cdot \kappa^2 \kappa_{\mathcal{X}})$ for large enough implicit constant.

Next, let $\Lambda(p) \triangleq \left(\oplus_i \left(\mathrm{pad}_S \left(\frac{\varphi(p)_i}{\log(1/\sigma_i)}\right)\right)\right)$ be an embedding; note that our selection of S guarantees that the padding is large enough. Finally, let $\hat{\mathcal{R}}$ be a point partition according to minwise hash function over $\Lambda(P)$.

Now we proceed to proving Eqn. (7.6). To show it holds for $\mathcal{R}$ as above, we use Lemma 155, and thus analyze instead random partition of an embedding with a different padding. For $i \in [m/s]$, let $T_i \leq S$ be some quantities that we chose later. Let $\Lambda'(p) \triangleq \left(\oplus_i \left(\mathrm{pad}_{T_i} \left(\frac{\varphi(p)_i}{\log(1/\sigma_i)}\right)\right)\right)$. Now, applying iteratively Lemma 155 for coordinates $(i, |\Lambda_i|)$ with $\Delta = S - T_i$ to argue it suffice to show, for $\mathcal{R}$ the minhash partition over $\Lambda'(P)$:

$$\Pr_{\mathcal{R}}[\mathcal{R}(p) = \mathcal{R}(q)] \cdot E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|P \cap \mathcal{R}(p)|}{|P|}\right\}\right)^{-\rho} \mid \mathcal{R}(p) = \mathcal{R}(q)\right] \geq 1. \tag{7.7}$$

Finally, let $\lambda_m \triangleq \min\{\Lambda'(p), \Lambda'(q)\}$ and $\lambda_M \triangleq \max\{\Lambda'(p), \Lambda'(q)\}$, and let $\mathcal{R}_M$ be the minwise hash for λ_M.

Letting Q be the LHS of Eqn. (7.6), and overloading notation that $\mathcal{R}(p)$ denotes both the subspace colliding with p as well as the "winning coordinate", we have, applying $E[\mathbf{X} \cdot \mathbf{1}[\mathbf{A}]] = \Pr[\mathbf{A}] \cdot E[\mathbf{X} \mid \mathbf{A}]$:

$$
\begin{aligned}
Q &= E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|P\cap\mathcal{R}(p)|}{|P|}\right\}\right)^{-\rho}\cdot\mathbf{1}\left[\mathcal{R}(p)=\mathcal{R}(q)\right]\right] \\
&= \sum_{i\in[m/s]} E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|P\cap\mathcal{R}(p)|}{|P|}\right\}\right)^{-\rho}\cdot\mathbf{1}\left[\mathcal{R}(p)=\mathcal{R}(q)\in(i,*)\right]\right] \\
&= \sum_{i\in[m/s]} \Pr_{\mathcal{R}}\left[\mathcal{R}(p)=\mathcal{R}(q)\in(i,*)\right]\cdot E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|P\cap\mathcal{R}(p)|}{|P|}\right\}\right)^{-\rho} \mid \mathcal{R}(p)=\mathcal{R}(q)\in(i,*)\right] \\
&= \sum_{i\in[m/s]} \Pr_{\mathcal{R}}\left[\mathcal{R}_M\in(i,*)\right]\cdot\Pr_{\mathcal{R}}\left[\mathcal{R}(p)=\mathcal{R}(q)\mid\mathcal{R}_M\in(i,*)\right] \\
&\qquad\cdot E_{\mathcal{R}}\left[\left(\max\left\{\delta, \frac{|P\cap\mathcal{R}(p)|}{|P|}\right\}\right)^{-\rho} \mid \mathcal{R}(p)=\mathcal{R}(q)\in(i,*)\right].
\end{aligned}
$$

We analyze each of those quantities. First, by the minhash construction, we have:

$$\Pr_{\mathcal{R}}[\mathcal{R}_M\in(i,*)] = \|(\lambda_M)_i\|_1/Z,$$

for some fixed quantity $Z \triangleq \|\lambda_M\|_1$.

Now we analyze the other quantities. Define $\rho_i \triangleq \frac{\log M(\varphi(p)_i,\varphi(q)_i)}{\log\sigma_i}$. For fixed i, we use Lemma 150 using $x = \frac{\varphi(p)_i}{\log(1/\sigma_i)}$, $y = \frac{\varphi(q)_i}{\log(1/\sigma_i)}$, and $\varphi = \frac{\varphi(*)_i}{\log(1/\sigma_i)}$. Observe that, by the minhash construction and definition of σ_i, ρ_i, for any $T_i \geq \left\|\frac{\varphi(p)}{\log(1/\sigma_i)}\right\|_1, \left\|\frac{\varphi(p)}{\log(1/\sigma_i)}\right\|_1$:

$$
\begin{aligned}
\log\left(\Pr_{\mathcal{R}}\left[\mathcal{R}(p)=\mathcal{R}(q)\mid\mathcal{R}_M\in(i,*)\right]\right) &= \log J(\mathrm{pad}_{T_i}(x),\mathrm{pad}_{T_i}(y)) \\
&\geq \log J(\varphi(p)_i,\varphi(q)_i)\cdot\zeta\frac{\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i\cdot\log(1/\sigma_i)} \\
&\overset{154}{\geq} \log(\sigma_i)\cdot\rho_i\cdot 2\zeta\frac{\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i\cdot\log 1/\sigma_i} \\
&= -2\zeta\rho_i\cdot\frac{\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i}.
\end{aligned}
$$

For the last quantity, we first argue the following statistical claim:

Claim 156. *Fix scalar $\rho \in (0,1)$. Let $\mathbf{X}$ be a random variable, taking values in $(0,1]$ and $\mathbf{A}$ be an event. Then:*

$$E[\mathbf{X}^{-\rho} \mid \mathbf{A}] \geq \left(\frac{E[\mathbf{X}\cdot\mathbf{1}[\mathbf{A}]]}{\Pr[\mathbf{A}]}\right)^{-\rho}$$

Proof. By Jensen's inequality, we have:

$$E[\mathbf{X}^{-\rho} \mid \mathbf{A}] \geq E[\mathbf{X} \mid \mathbf{A}]^{-\rho} = E\left[\frac{\mathbf{X}\cdot\mathbf{1}[\mathbf{A}]}{\Pr[\mathbf{A}]}\right]^{-\rho}.$$

□

We note that the claim holds even when we condition everything on a fixed event $\mathbf{C}$. Now, define r to be one of p or q depending on which norm is larger: $r = \text{argmax}_{r\in\{p,q\}} \|\varphi(r)_i\|_1$. By Claim 156, setting $\mathbf{X} \triangleq \frac{|P\cap\mathcal{R}(p)|}{|P|}$, $\mathbf{A} \triangleq \{\mathcal{R}(p) = \mathcal{R}(q)\}$, and conditioning on event $\mathbf{C} \triangleq \{\mathcal{R}_M \in (i,*)\}$:

$$\begin{aligned}
&\log\left(E_{\mathcal{R}}\left[\max\left(\delta, \frac{|P\cap\mathcal{R}_m|}{|P|}\right)^{-\rho} \mid \mathcal{R}(p) = \mathcal{R}(q) \in (i,*)\right]\right) \\
&\quad\geq \log\left(\frac{E_{\mathcal{R}}[\max\{\delta, X\}\cdot\mathbf{1}[\mathcal{R}(p)=\mathcal{R}(q)]] \mid \mathcal{R}_M \in (i,*)]}{\Pr[\mathcal{R}(p)=\mathcal{R}(q)] \mid \mathcal{R}_M \in (i,*)]}\right)^{-\rho} \\
&\quad\geq \log\left(\frac{E_{\mathcal{R}}[\delta + X \mid \mathcal{R}(r) \in (i,*)]}{J(\Lambda'(p)_i, \Lambda'(q)_i)}\right)^{-\rho} \\
&\quad\geq \log\left(\frac{J(\Lambda'(p)_i, \Lambda'(q)_i)}{\delta + E_{p'\in P} J(\Lambda'(r)_i, \Lambda'(p')_i)}\right)^{\rho} \\
&\quad\geq \log\left(\frac{M(\Lambda'(p)_i, \Lambda'(q)_i)^2}{\delta + E_{p'\in P} M(\Lambda'(r)_i, \Lambda'(p')_i)}\right)^{\rho}
\end{aligned}$$

where we used Fact 154. Using both parts of Lemma 150, we further have, by the definition of σ_i, Λ', for large enough constant $\zeta > 1$, there is $T_i \geq \left\|\frac{\varphi(r)}{\log(1/\sigma_i)}\right\|_1 = \max\left\{\left\|\frac{\varphi(p)}{\log(1/\sigma_i)}\right\|_1, \left\|\frac{\varphi(p)}{\log(1/\sigma_i)}\right\|_1\right\}$ s.t.:

- $\log M(\Lambda'(p)_i, \Lambda'(q)_i) \geq \log M(\varphi(p)_i, \varphi(q)_i) \cdot \zeta^{\frac{\|\varphi(p)_i\|_1 + \|\varphi(q)_i\|_1}{\log 1/\sigma_i \cdot T_i}} = -\rho_i \cdot \zeta^{\frac{\|\varphi(p)_i\|_1 + \|\varphi(q)_i\|_1}{T_i}}$, where we used the definition of ρ_i, and
- $\log\left(\delta + E_{p' \in P} M(\Lambda'(r)_i, \Lambda'(p')_i)\right) \leq \log\left(\delta + \exp\left(\log \sigma_i \cdot \frac{\|\varphi(r)_i\|}{\zeta \log 1/\sigma_i \cdot T_i}\right)\right) = \log\left(\delta + \exp\left(-\frac{\|\varphi(r)_i\|}{\zeta T_i}\right)\right) \leq -\frac{\|\varphi(r)_i\|}{2\zeta T_i}$, where we used the fact that $\exp\left(-\frac{\|\varphi(r)_i\|}{\zeta T_i}\right) \in [\sigma_i, 1 - \Omega(1/\kappa^2 \kappa_{\mathcal{X}})]$ and $\delta < \min\{\sigma_i, 1/\kappa^2 \kappa_{\mathcal{X}}\}$.

Hence, replacing in the above, we get:

$$\begin{aligned}
&\log\left(E_{\mathcal{R}}\left[\max\left(\delta, \frac{|P \cap \mathcal{R}_m|}{|P|}\right)^{-\rho} \mid \mathcal{R}(p) = \mathcal{R}(q) \in (i, *)\right]\right) \\
&\quad \geq \rho\left(\frac{\|\varphi(p)_i\| + \|\varphi(q)_i\|_1}{4\zeta T_i} - \rho_i \cdot 2\zeta \frac{\|\varphi(p)_i\|_1 + \|\varphi(q)_i\|_1}{T_i}\right) \\
&\quad = \frac{1}{4\zeta} \cdot \rho(1 - 8\zeta^2 \rho_i) \cdot \frac{\|\varphi(p)_i + \varphi(q)_i\|_1}{T_i}.
\end{aligned}$$

Before completing, we define a set $B \subset [m/s]$ of outer coordinates where, in the WASE of φ, the min is achieved by $1 + 1/l_i$, with $l_i = \log 1/\mathbb{E}_{p', p''} M(\varphi(p'), \varphi(p''))$. Let $\pi_B = \sum_{i \in B} (\lambda_M)_i / Z$. Furthermore, for each $i \in B$, we set $T_i = \frac{\|\varphi(p)_i + \varphi(q)_i\|}{\log 1/\sigma_i}$, i.e., the (approximately) minimal T_i.

Finally, combining all the above, we derive, using Jensen's inequality again:

$$\log(Q) = \log$$
$$\left(\sum_{i\in[m/s]} \Pr_{\mathcal{R}}[\mathcal{R}_M \in (i,*)] \Pr_{\mathcal{R}}[\mathcal{R}(p)=\mathcal{R}(q) \,|\, \mathcal{R}_M \in (i,*)] \, \mathbb{E}_{\mathcal{R}} \left[\left(\max\left\{\delta, \tfrac{|P\cap\mathcal{R}_m|}{|P|}\right\}\right)^{-\rho} | \, \mathcal{R}(p)=\mathcal{R}(q) \in (i,*)\right]\right)$$
$$\geq \log\left((1-\pi_B) \sum_{i\in[m/s]\setminus B} \tfrac{\|(\lambda_M)_i\|_1}{(1-\pi_B)Z} \exp\left(-2\zeta\rho_i \cdot \tfrac{\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i} + \tfrac{1}{4\zeta}\cdot\rho(1-8\zeta^2\rho_i)\cdot\tfrac{\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i}\right)\right)$$
$$\geq -\pi_B + \sum_{i\in[m/s]\setminus B} \tfrac{\|(\lambda_M)_i\|_1}{(1-\pi_B)Z}\left(-2\zeta\rho_i \cdot \tfrac{\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i} + \tfrac{1}{4\zeta}\cdot\rho(1-8\zeta^2\rho_i)\cdot\tfrac{\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i}\right)$$
$$= -\pi_B + \tfrac{1}{(1-\pi_B)Z}\cdot \sum_{i\in[m/s]\setminus B} \tfrac{\|(\lambda_M)_i\|_1\cdot\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i}\cdot\left(-2\zeta\rho_i + \tfrac{1}{4\zeta}\cdot\rho(1-8\zeta^2\rho_i)\right)$$
$$= -\pi_B + \tfrac{1}{4(1-\pi_B)\zeta Z}\cdot \sum_{i\in[m/s]\setminus B} \tfrac{\|(\lambda_M)_i\|_1\cdot\|\varphi(p)_i+\varphi(q)_i\|_1}{T_i}\cdot\left(\rho - 16\zeta^2\rho_i(1+\rho)\right).$$

Furthermore, using the fact that $\|(\lambda_M)_i\|_1 \in [T_i, 2T_i]$, we get:

$$\log Q \geq -\pi_B + \tfrac{1}{2\zeta(1-\pi_B)Z}\cdot \sum_{i\in[m/s]\setminus B} \|\varphi(p)_i+\varphi(q)_i\|_1 \cdot \left(\rho - 16\zeta^2\rho_i(1+\rho)\right).$$

We now use the fact φ is WASE embedding for P to obtain $\sum_{i\in[m/s]\setminus B} \frac{\|\varphi(p)_i+\varphi(q)_i\|_1}{\|\varphi(p)+\varphi(q)\|_1}\cdot$ $\rho_i \leq c\cdot\frac{\mathsf{d}(p,q)}{\widehat{\mathsf{d}}}$. Also, by the definition of the set B:

$$\sum_{i\in[m/s]\setminus B} \|\varphi(p)_i+\varphi(q)_i\|_1 \geq \|\varphi(p)+\varphi(q)\| - \sum_{i\in\setminus B} \|\varphi(p)_i+\varphi(q)_i\|_1 \geq \|\varphi(p)+\varphi(q)\|\cdot(1-c\cdot\tfrac{\mathsf{d}(p,q)}{\widehat{\mathsf{d}}}).$$

Therefore, for any $\rho \in \left[40\zeta^2\cdot c\cdot\frac{\mathsf{d}(p,q)}{\widehat{\mathsf{d}}}, 1\right]$, we have

$$\log(Q) \geq -\pi_B + \tfrac{1}{2(1-\pi_B)\zeta Z}\cdot\|\varphi(p)+\varphi(q)\|_1\cdot\left(\rho\tfrac{39}{40} - 16\zeta^2(1+\rho)\cdot\rho/40\zeta^2\right)$$
$$\geq -\pi_B + \tfrac{1}{2\zeta Z}\cdot\|\varphi(p)+\varphi(q)\|_1\cdot\rho/6.$$

To bound π_B, again by the definition of B, we have:

$$Z\cdot\pi_B = \sum_{i\in B}\|(\lambda_M)_i\| \leq \sum_{i\in B} 2\tfrac{\|\varphi(p)_i+\varphi(q)_i\|_1}{\log 1/\sigma_i} \leq \|\varphi(p)+\varphi(q)\|_1\cdot c\cdot\tfrac{\mathsf{d}(p,q)}{\widehat{\mathsf{d}}} \leq \|\varphi(p)+\varphi(q)\|_1\cdot\rho/40\zeta^2.$$

Finally, we conclude

$$\log Q \geq \tfrac{\rho/6}{2\zeta Z} \cdot \|\varphi(p) + \varphi(q)\|_1 - \tfrac{\rho}{40\zeta^2 Z}\|\varphi(p) + \varphi(q)\|_1 > 0$$

as needed. This concludes the proof of the Eqn. (7.7) and hence Eqn. (7.6).

For the last part of the claim, we note that, as before:

$$\begin{aligned}
E_{\mathcal{R}}\left[\frac{|P \cap \mathcal{R}(p)|}{|P|}\right] &= \sum_{i \in [m/s]} \Pr_{\mathcal{R}}\left[\mathcal{R}(p) \in (i,*)\right] \cdot E_{\mathcal{R}}\left[\frac{|P \cap \mathcal{R}(p)|}{|P|} \mid \mathcal{R}(p) \in (i,*)\right] \\
&\leq \sum_{i \in [m/s]} \tfrac{\|\Lambda'(p)_i\|}{\|\Lambda'(p)\|} \cdot \exp\left(-\tfrac{\|\varphi(p)_i\|}{\zeta T_i}\right) \\
&\leq \sum_{i \in [m/s]} \tfrac{\|\Lambda'(p)_i\|}{\|\Lambda'(p)\|} \cdot 1 - \Omega(1/\kappa^2\kappa_{\mathcal{X}}) \\
&\leq 1 - \Omega(1/\kappa^2\kappa_{\mathcal{X}}).
\end{aligned}$$

Furthermore, since $\hat{\mathcal{R}}$ and $\mathcal{R}$ randomized partitioning differ only in the padding, we have that

$$E_{\hat{\mathcal{R}}}\left[\frac{|P \cap \hat{\mathcal{R}}(p)|}{|P|}\right] \leq 1 - \Omega(1/\kappa^2\kappa_{\mathcal{X}}) \cdot 1/S = 1 - \mathrm{poly}(1/m\kappa\kappa_{\mathcal{X}}).$$

□

7.8.4 ANN algorithms for ℓ_p and edit distance

We now prove Theorems 12 and 13 from Chapter 1, on ANNS algorithms for ℓ_p and edit distance, by putting together the elements proven in the prior sections.

We start by proving the ANNS for ℓ_p.

Proof of Theorem 13. We use the WASE embedding from Section 7.5 together with the ANNS algorithm from Theorem 148. We need however to ensure bounded aspect ratios.

To ensure the aspect ratio of the ℓ_p metric, we do the following standard preprocessing to obtain $\kappa_{\mathcal{X}} = O(d^3)$. Fix scale r from the ANNS instance to be solved. First of all, impose a randomly shifted grid, with side-lengths $100\sqrt{d}r$. Then, the following holds for any two points $p, q \in \mathbb{R}^d$. 1) If $\|p-q\|_X \leq r$, then the probability being in the same cell of the grid is ≥ 0.99, and 2) the side-length of any cell is $O(dr)$. Thus we can apply this partition first and solve ANNS for each cell, where each point (dataset and query) is rounded to have all coordinates be integer multiple of r/d. The rounding introduces an additive error of at most $\pm r$, i.e., an extra factor 2 approximation. Now note that, in each cell, after multiplying all the points by d/r, we are considering ANN in the space $(\{0, 1, 2, \ldots O(d^2)]^d, \ell_p)$, which has aspect ratio $O(d^3)$. □

Proof of Theorem 12. We note that the edit distance the aspect ratio is $\kappa_{\mathcal{X}} = k$ from start. As a result, as we build the WASE embedding for edit distance (Theorem 131), the aspect ratios remains polynomial in k throughout. Therefore, applying Theorem 148 implies our Theorem 12. □

Bibliography

[1] A. Abboud, A. Backurs, and V. V. Williams. Tight hardness results for lcs and other sequence similarity measures. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 59–78, 2015. 5

[2] A. Abboud and K. Bringmann. Tighter connections between formula-sat and shaving logs. *arXiv preprint arXiv:1804.08978*, 2018. 5

[3] A. Abboud, T. D. Hansen, V. V. Williams, and R. Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 375–388. ACM, 2016. 1.1, 5

[4] A. Abdullah, A. Andoni, R. Kannan, and R. Krauthgamer. Spectral approaches to nearest neighbor search. In *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2014)*, pages 581–590, 2014. 2.6

[5] N. Ailon, B. Chazelle, S. Comandur, and D. Liu. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31:371–383, 2007. Previously in RANDOM'04. 1.1, 2.5

[6] S. Akmal and V. Vassilevska Williams. Improved Approximation for Longest Common Subsequence over Small Alphabets. In N. Bansal, E. Merelli, and J. Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 2.1

[7] J. Alman, T. M. Chan, and R. Williams. Polynomial representations of threshold functions with applications. In *Proceedings of the 57th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2016)*, 2016. To appear. 1.2.1

[8] A. Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, MIT, 2009. Available at http://www.mit.edu/~andoni/thesis/main.pdf. 1.2.3

[9] A. Andoni. High-dimensional computational geometry. In P. Bühlmann, P. Drineas, M. Kane, and M. J. van der Laan, editors, *Handbook of Big Data.*, pages 105–123. Chapman and Hall/CRC, 2016. 1.2

[10] A. Andoni. Simpler constant-factor approximation to edit distance problems, 2018. Manuscript, available at http://www.cs.columbia.edu/~andoni/papers/edit/. 1.1, 2.1

[11] A. Andoni and D. Cheikhi. From average embeddings to nearest neighbor search. *CoRR*, abs/2105.05761, 2021. 1.2.3

[12] A. Andoni, D. Croitoru, and M. Patrascu. Hardness of nearest neighbor under L-infinity. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2008)*, pages 424–433, 2008. 1.2.3, 7.1.2

[13] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2006)*, pages 459–468, 2006. 1.2

[14] A. Andoni, P. Indyk, and R. Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 343–352, 2008. Previously ECCC Report TR07-048. 2.7, 7.1.4

[15] A. Andoni, P. Indyk, and R. Krauthgamer. Overcoming the ℓ_1 non-embeddability barrier: Algorithms for product metrics. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 865–874, 2009. 1.2.3, 2.5, 2.7

[16] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. In *Proceedings of Advances in Neural Information Processing Systems 28 (NIPS '2015)*, pages 1225–1233, 2015. Available as arXiv:1509.02897. 1.2.1, 1.2.2, 6

[17] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn. Beyond locality-sensitive hashing. In *Proceedings of the 25th ACM-SIAM Symposium on*

Discrete Algorithms (SODA '2014), pages 1018–1028, 2014. Available as arXiv:1306.1547. 1.2, 1.2.1, 1.2.2, 6

[18] A. Andoni, P. Indyk, and I. Razenshteyn. Approximate nearest neighbor search in high dimensions. 2018. This survey accompanied the ICM 2018 talk of Piotr Indyk. arXiv preprint arXiv:1806.09823. 1.2.3

[19] A. Andoni and R. Krauthgamer. The computational hardness of estimating edit distance. *SIAM Journal on Computing*, 39(6):2398–2429, 2010. Previously in FOCS'07. 1.2.3, 2.7

[20] A. Andoni and R. Krauthgamer. The smoothed complexity of edit distance. *ACM Transactions on Algorithms*, 8(4):44, 2012. Previously in ICALP'08. 2.3, 2.5, 6

[21] A. Andoni, R. Krauthgamer, and K. Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2010. Full version at http://arxiv.org/abs/1005.4033. 1.1, 6, 4.2.5, 4.2.5.1

[22] A. Andoni, R. Krauthgamer, and K. Onak. Streaming algorithms from precision sampling. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2011. Full version at http://arxiv.org/abs/1011.1263. 4.2.5, 4.2.5.1

[23] A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten. Lower bounds on time–space trade-offs for approximate near neighbors. Available as arXiv:1605.02701, 2016. 1.2

[24] A. Andoni, A. Naor, A. Nikolov, I. P. Razenshteyn, and E. Waingarten. Data-dependent hashing via nonlinear spectral gaps. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2018. 1.2.3, 1.2.3, 7.8

[25] A. Andoni and H. L. Nguyen. Near-tight bounds for testing Ulam distance. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010. 1.1

[26] A. Andoni, A. Nikolov, I. P. Razenshteyn, and E. Waingarten. Approximate near neighbors for general symmetric norms. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2017. Previously in `http://arxiv.org/abs/1611.06222`. 1.2.3

[27] A. Andoni, A. Nikolov, I. P. Razenshteyn, and E. Waingarten. Approximate nearest neighbors beyond space partitions. In D. Marx, editor, *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1171–1190. SIAM, 2021. 1.2.3, 1.2.3, 7.1.1, 7.1.2

[28] A. Andoni and N. S. Nosatzki. Edit distance in near-linear time: it's a constant factor. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2020. 1.1, 1.1

[29] A. Andoni, N. S. Nosatzki, S. Sinha, and C. Stein. Estimating the longest increasing subsequence in nearly optimal time. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 708–719. IEEE, 2022. 1.1, 2.1, 5, 5.1.1, 2, 94, 95

[30] A. Andoni and K. Onak. Approximating edit distance in near-linear time. *SIAM J. Comput. (SICOMP)*, 41(6):1635–1648, 2012. Previously in STOC'09. 1.1, 2.3, 3.2.2

[31] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 793–801, 2015. Available as arXiv:1501.01062. 1.2, 1.2.1, 1.2.2, 6

[32] A. Andoni and I. Razenshteyn. Tight lower bounds for data-dependent locality-sensitive hashing. In *Proceedings of the 32nd International Symposium on Computational Geometry (SoCG '2016)*, pages 9:1–9:11, 2016. Available as arXiv:1507.04299. 1.2

[33] A. Andoni, I. Razenshteyn, and N. S. Nosatzki. Lsh forest: Practical algorithms made theoretical. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2017. 1.2.2, 7.1.1

[34] A. Apostolico, S. Browne, and C. Guerra. Fast linear-space computations of longest common subsequences. *Theoretical Computer Science*, 92(1):3–17, 1992. 2.2

[35] A. Apostolico and C. Guerra. The longest common subsequence problem revisited. *Algorithmica*, 2(1):315–336, 1987. 5, 5.1.2

[36] A. Apostolico and C. Guerra. Monotonic minimal prefix transversals and applications to string comparison. *Theoretical Computer Science*, 92(1):3–17, 1992. 2.4

[37] A. Backurs and P. Indyk. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2015. 1.1

[38] Y. Bao, A. Baweja, N. Menand, E. Waingarten, N. White, and T. Zhang. Average-distortion sketching. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2025. 1.2.3, 1.2.3, 7, 7.1.2

[39] Z. Bar-Yossef, T. S. Jayram, R. Krauthgamer, and R. Kumar. Approximating edit distance efficiently. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 550–559, 2004. 1.1, 2.3

[40] Y. Bartal and L. Gottlieb. Approximate nearest neighbor search for ℓ_p-spaces ($2 < p < \infty$) via embeddings. *CoRR*, abs/1512.01775, 2015. 1.2.3, 7.1.2

[41] T. Batu, F. Ergün, J. Kilian, A. Magen, S. Raskhodnikova, R. Rubinfeld, and R. Sami. A sublinear algorithm for weakly approximating edit distance. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 316–324, 2003. 1.1, 2.3

[42] T. Batu, F. Ergün, and C. Sahinalp. Oblivious string embeddings and edit distance approximations. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 792–801, 2006. 1.1

[43] M. Bawa, T. Condie, and P. Ganesan. LSH forest: self-tuning indexes for similarity search. In *Proceedings of the 14th international conference on World Wide Web (WWW '2005)*, pages 651–660, 2005. 1.2.2, 6, 6.2

[44] M. Bawa, T. Condie, and P. Ganesan. Lsh forest: self-tuning indexes for similarity search. In *Proceedings of the 14th international conference on World Wide Web*, pages 651–660, 2005. 7.1.1

[45] D. Belazzougui and Q. Zhang. Edit distance: Sketching, streaming, and document exchange. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 51–60, 2016. 2.2

[46] O. Ben-Eliezer, C. Canonne, S. Letzter, and E. Waingarten. Finding monotone patterns in sublinear time. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1469–1494, 2019. 1.1

[47] E. Bernhardsson. ANNOY: Approximate nearest neighbors in C++/Python optimized for memory usage and loading/saving to disk, 2013. Available as `https://github.com/spotify/annoy`. 1.2.2, 6

[48] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning (ICML '2006)*, pages 97–104, 2006. 2.6

[49] H. Black, D. Chakrabarty, and C. Seshadhri. Domain reduction for monotonicity testing: A o(d) tester for boolean functions in d-dimensions. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 19751994, USA, 2020. Society for Industrial and Applied Mathematics. 1.1

[50] M. Boroujeni, S. Ehsani, M. Ghodsi, M. HajiAghayi, and S. Seddighin. Approximating edit distance in truly subquadratic time: Quantum and mapreduce. In

Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, pages 1170–1189. SIAM, 2018. 1.1, 2.1, 2.4, 3.2.1

[51] M. Boroujeni, M. Seddighin, and S. Seddighin. *Improved Algorithms for Edit Distance and LCS: Beyond Worst Case*, pages 1601–1620. 2.3

[52] M. Boroujeni and S. Seddighin. Improved mpc algorithms for edit distance and ulam distance. In *Proceedings of the 31st ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '19, page 3140, New York, NY, USA, 2019. Association for Computing Machinery. 1.1

[53] R. S. Boyer and J. S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20(10):762–772, 1977. 2.5

[54] J. Brakensiek, M. Charikar, and A. Rubinstein. A simple sublinear algorithm for gap edit distance. *arXiv preprint arXiv:2007.14368*, 2020. 2.3

[55] J. Brakensiek and A. Rubinstein. Constant-factor approximation of near-linear edit distance in near-linear time. *arXiv preprint arXiv:1904.05390*, 2019. 2.1, 3, 3.2.1

[56] J. Brakensiek and A. Rubinstein. Constant-factor approximation of near-linear edit distance in near-linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 685–698, New York, NY, USA, 2020. Association for Computing Machinery. 1.1

[57] M. Braverman and Y. K. Ko. Semi-direct sum theorem and nearest neighbor under l_infty. In *Approximation, Randomization, and Combinatorial Op-*

timization. Algorithms and Techniques (APPROX/RANDOM 2018). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018. 1.2.3, 7.1.2

[58] K. Bringmann. *Sampling from discrete distributions and computing Fréchet distances*. PhD thesis, Saarland University, 2014. 3.9

[59] K. Bringmann, A. Cassis, N. Fischer, and T. Kociumaka. Faster sublinear-time edit distance. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 3274–3301. SIAM, 2024. 2.3

[60] K. Bringmann, A. Cassis, N. Fischer, and V. Nakos. Almost-optimal sublinear-time edit distance in the low distance regime. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2022. 2.3

[61] K. Bringmann, V. Cohen-Addad, and D. Das. A linear-time $n^{0.4}$-approximation for longest common subsequence. *CoRR*, abs/2106.08195, 2021. 1.1, 5

[62] K. Bringmann, V. Cohen-Addad, and D. Das. A linear-time $n^{0.4}$-approximation for longest common subsequence. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, 2021. 2.1

[63] K. Bringmann, A. Driemel, A. Nusser, and I. Psarros. Tight bounds for approximate near neighbor searching for time series under the fréchet distance. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 517–550. SIAM, 2022. 2.7

[64] K. Bringmann, A. Dürr, and A. Polak. Even faster knapsack via rectangular monotone min-plus convolution and balancing. *CoRR*, abs/2404.05681, 2024. 2.3

[65] K. Bringmann and M. Künnemann. Quadratic conditional lower bounds for string problems and dynamic time warping. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 79–97. IEEE, 2015. 5

[66] A. Z. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences (SEQUENCES '1997)*, pages 21–29, 1997. 6

[67] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the web. *Computer Networks and ISDN Systems*, 29(8–13):1157–1166, 1997. 6

[68] A. Chakrabarti. A note on randomized streaming space bounds for the longest increasing subsequence problem. *Inf. Process. Lett.*, 112(7):261263, Mar. 2012. 2.2

[69] D. Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10(17):453–464, 2014. 1.1

[70] D. Chakraborty, D. Das, E. Goldenberg, M. Koucky, and M. Saks. Approximating edit distance within constant factor in truly sub-quadratic time. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 979–990. IEEE, 2018. 1.1, 2.1, 3, 3.2.1, 3.2.2, 3.2.3

[71] D. Chakraborty, D. Das, and M. Koucky. Approximate online pattern matching in sub-linear time. In *FSTTCS*, 2019. 2.1

[72] H.-L. Chan, T.-W. Lam, W.-K. Sung, S.-L. Tam, and S.-S. Wong. A linear size index for approximate pattern matching. In *Combinatorial Pattern Match-*

ing: 17th Annual Symposium, CPM 2006, Barcelona, Spain, July 5-7, 2006. Proceedings 17, pages 49–59. Springer, 2006. 2.7

[73] T. M. Chan, R. Clifford, T. Kociumaka, K. G. Larsen, and T. Starikovskaya. Edit distance in the cell-probe model. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 116–128. SIAM, 2021. 2.4, 2.5

[74] M. Charikar. Similarity estimation techniques from rounding. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 380–388, 2002. 1.2.3, 3.2.3, 7.1.1, 7.5

[75] M. Charikar, M. Kapralov, N. Nouri, and P. Siminelakis. Kernel density estimation through density constrained near neighbor search. In S. Irani, editor, *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 172–183. IEEE, 2020. 1.2

[76] M. Charikar and R. Krauthgamer. Embedding the Ulam metric into ℓ_1. *Theory of Computing*, 2(11):207–224, 2006. 2.7

[77] E. Chávez, G. Navarro, R. A. Baeza-Yates, and J. L. Marroquín. Searching in metric spaces. *ACM Computing Surveys*, 33(3):273–321, 2001. 2.6

[78] S. Chechik. Approximate distance oracles with constant query time. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 654–663, New York, NY, USA, 2014. ACM. 3.2.2

[79] Y. Cherapanamjeri and J. Nelson. Terminal embeddings in sublinear time. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS*

2021, Denver, CO, USA, February 7-10, 2022, pages 1209–1216. IEEE, 2021. 1.2

[80] F. Chierichetti and R. Kumar. Lsh-preserving functions and their applications. *Journal of the ACM (JACM)*, 62(5):33, 2015. 1.2.3

[81] K. L. Clarkson. Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006. 2.6

[82] R. Clifford and T. Starikovskaya. Streaming and sketching algorithms for processing massive data. In *SODA 2020*, pages 2691–2702. SIAM, 2020. 2.5

[83] R. Cole, L. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't caresb. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 91–100, 2004. 2.7

[84] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001. 3.9

[85] N. R. Council. *Frontiers in Massive Data Analysis*. The National Academies Press, Washington, DC, 2013. 1.2.1

[86] M. Crochemore and E. Porat. Fast computation of a longest increasing subsequence and application. *Information and Computation*, 208(9):1054–1059, 2010. 1.1

[87] S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the 40th ACM Symposium on the Theory of Computing (STOC '2008)*, pages 537–546, 2008. 2.6

[88] S. Dasgupta and S. Kpotufe. Nearest-neighbor classification and search. In T. Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press, 2021. 1.2

[89] S. Dasgupta and K. Sinha. Randomized partition trees for exact nearest neighbor search. In *Proceedings of the 26th Annual Conference on Learning Theory (COLT '2013)*, pages 317–337, 2013. 2.6, 6

[90] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the 20th ACM Symposium on Computational Geometry (SoCG '2004)*, pages 253–262, 2004. 1.2

[91] Y. Dodis, O. Goldreich, E. Lehman, S. Raskhodnikova, D. Ron, and A. Samorodnitsky. Improved testing algorithms for monotonicity. *Electron. Colloquium Comput. Complex.*, 6(17), 1999. 1.1

[92] T. Easton and A. Singireddy. A large neighborhood search heuristic for the longest common subsequence problem. *Journal of Heuristics*, 14:271–283, 2008. 2.7

[93] D. Eppstein, Z. Galil, R. Giancarlo, and G. F. Italiano. Sparse dynamic programming i: Linear cost functions. *J. ACM*, 39(3):519545, jul 1992. 5

[94] F. Ergün and H. Jowhari. On distance to monotonicity and longest increasing subsequence of a data stream. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 730–736, 2008. 1.1, 2.2, 2.5

[95] F. Ergün, S. Kannan, R. Kumar, R. Rubinfeld, and M. Viswanathan. Spot-checkers. *J. Comput. Syst. Sci.*, 60(3):717–751, 2000. 1.1

[96] E. Fischer. The art of uninformed decisions: A primer to property testing. *Science*, 75:97–126, 2001. 1.1

[97] E. Fischer. On the strength of comparisons in property testing. *Inf. Comput.*, 189(1):107116, Feb. 2004. 1.1

[98] Y. Fu, C. Wang, J. Zhu, Q. Zhang, Y. Cheung, and Y. Wang. Dynamic-mlcs: Fast searching for dynamic multiple longest common subsequences in sequence stream data. *Knowledge-Based Systems*, 293:111654, 2024. 2.2

[99] A. Gál and P. Gopalan. Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 294–304, 2007. 2.2

[100] P. Gawrychowski and W. Janczewski. Conditional lower bounds for variants of dynamic lis, 2021. 2.4

[101] P. Gawrychowski and W. Janczewski. Fully dynamic approximation of lis in polylogarithmic time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, page 654667, New York, NY, USA, 2021. Association for Computing Machinery. 2.4, 4.2.3, 4.5.1.1, 79

[102] P. Gawrychowski, S. Mozes, and O. Weimann. Fully dynamic longest common subsequence in subquadratic time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1126–1139, 2020. 2.4

[103] E. Goldenberg, T. Kociumaka, R. Krauthgamer, and B. Saha. Gap edit distance via non-adaptive queries: Simple and optimal. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, 2022. 2.3

[104] E. Goldenberg, R. Krauthgamer, and B. Saha. Sublinear algorithms for gap edit distance. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1101–1120. IEEE, 2019. 2.3

[105] E. Goldenberg, A. Rubinstein, and B. Saha. Does preprocessing help in fast sequence comparisons? In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2020. 1.1, 2.1

[106] O. Goldreich. *Introduction to property testing*. Cambridge University Press, 2017. 1.1

[107] P. Gopalan, T. S. Jayram, R. Krauthgamer, and R. Kumar. Estimating the sortedness of a data stream. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 318–327, 2007. 1.1, 2.2, 2.5

[108] D. Gusfield. *Algorithms on strings, trees, and sequences*. Cambridge University Press, Cambridge, 1997. 1.1

[109] M. Hajiaghayi, M. Seddighin, S. Seddighin, and X. Sun. Approximating lcs in linear time: Beating the barrier. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1181–1200. SIAM, 2019. 2.1

[110] M. Hajiaghayi, M. Seddighin, S. Seddighin, and X. Sun. *Approximating LCS in Linear Time: Beating the $\sqrt{n}$ Barrier*, pages 1181–1200. 2019. 1.1

[111] M. Hajiaghayi, S. Seddighin, and X. Sun. Massively parallel approximation algorithms for edit distance and longest common subsequence. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1654–1672. SIAM, 2019. 2.4

[112] M. T. Hajiaghayi, M. Seddighin, S. Seddighin, and X. Sun. *Approximating LCS in Linear Time: Beating the $\sqrt{n}$ Barrier*. 2019. 1.1, 5

[113] J. Hammersley. A few seedlings of research. *Proc. Sixth Berkeley Symp. Math. Statist. and Probability*, pages 345–394, 1972. 1.1

[114] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012. 1.2, 1.2.2, 1.2.2, 6, 105, 106, 107, 7.1.1

[115] X. He and R. Li. Approximating binary longest common subsequence in almost-linear time. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 1145–1158, New York, NY, USA, 2023. Association for Computing Machinery. 2.1

[116] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24(4):664675, oct 1977. 5

[117] J. W. Hunt and T. G. Szymanski. A fast algorithm for computing longest common subsequences. *Commun. ACM*, 20(5):350353, may 1977. 5, 5.1.2

[118] S. Im, B. Moseley, and X. Sun. Efficient massively parallel methods for dynamic programming. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, page 798811, New York, NY, USA, 2017. Association for Computing Machinery. 2.4

[119] P. Indyk. *High-dimensional computational geometry*. Ph.D. Thesis. Department of Computer Science, Stanford University, 2001. 1.2

[120] P. Indyk. On approximate nearest neighbors in ℓ_∞ norm. *J. Comput. Syst. Sci.*, 63(4):627–638, 2001. Preliminary version appeared in FOCS'98. 1.2.3, 7.1.1, 7.1.2

[121] P. Indyk. Approximate nearest neighbor algorithms for fréchet distance via product metrics. In *Proceedings of the eighteenth annual symposium on Computational geometry*, pages 102–106, 2002. 1.2.3, 2.7

[122] P. Indyk. Approximate nearest neighbor under edit distance via product metrics. In *SODA*, volume 4, pages 646–650, 2004. 1.2.3

[123] P. Indyk. Approximate nearest neighbor under edit distance via product metrics. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 646–650, 2004. 2.7

[124] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th ACM Symposium on the Theory of Computing (STOC '1998)*, pages 604–613, 1998. 1.2, 1.2.2, 1.2.2, 6, 105, 106, 107

[125] P. Indyk and A. Naor. Nearest-neighbor-preserving embeddings. *ACM Transactions on Algorithms*, 3(3), 2007. 2.6

[126] P. Indyk and N. Thaper. Fast color image retrieval via embeddings. *Workshop on Statistical and Computational Theories of Vision (at ICCV)*, 2003. 2.7, 7.1.4

[127] R. Jayaram, E. Waingarten, and T. Zhang. Data-dependent lsh for the earth mover's distance. In *Proceedings of the Symposium on Theory of Computing (STOC)*, 2024. 2.7, 7.1.4

[128] C. Jin, J. Nelson, and K. Wu. An Improved Sketching Algorithm for Edit Distance. In M. Bläser and B. Monmege, editors, *38th International Symposium on Theoretical Aspects of Computer Science (STACS 2021)*, volume 187 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 45:1–45:16, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 2.2

[129] M. Kapralov and R. Panigrahy. NNS lower bounds via metric expansion for ℓ_∞ and EMD. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP)*, pages 545–556, 2012. 1.2.3, 7.1.2

[130] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*, pages 741–750, 2002. 2.6

[131] M. Karppa, P. Kaski, and J. Kohonen. A faster subquadratic algorithm for finding outlier correlations. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA '2016)*, pages 1288–1305, 2016. 1.2.1

[132] M. Karppa, P. Kaski, J. Kohonen, and P. Ó Catháin. Explicit correlation amplifiers for finding outlier correlations in deterministic subquadratic time. In *Proceedings of the 24th European Symposium Of Algorithms (ESA '2016)*, 2016. To appear. 1.2.1

[133] A. B. Khesin, A. Nikolov, and D. Paramonov. Preconditioning for the geometric transportation problem. In *35th International Symposium on Computational Geometry (SoCG 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019. 7.8

[134] S. Khot and A. Naor. Nonembeddability theorems via Fourier analysis. *Math. Ann.*, 334(4):821–852, 2006. Preliminary version appeared in FOCS'05. 1.2.3, 1.2.3, 2.7, 7, 7.1.4

[135] D. E. Knuth, J. H. Morris Jr, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977. 2.5

[136] T. Kociumaka, E. Porat, and T. Starikovskaya. Small-space and streaming pattern matching with k edits. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 885–896, 2022. 2.2

[137] T. Kociumaka and B. Saha. Sublinear-time algorithms for computing & embedding gap edit distance. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 1168–1179, 2020. 2.3

[138] T. Kociumaka and S. Seddighin. Improved dynamic algorithms for longest increasing subsequence. In *Proceedings of the 53rd Annual ACM SIGACT Sym-*

posium on Theory of Computing, STOC 2021, page 640653, New York, NY, USA, 2021. Association for Computing Machinery. 2.4, 4.2.3

[139] M. Koucký and M. Saks. Constant factor approximations to edit distance on far input pairs in nearly linear time. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, pages 699–712, New York, NY, USA, 2020. Association for Computing Machinery. 1.1

[140] M. Kouckỳ and M. E. Saks. Constant factor approximations to edit distance on far input pairs in nearly linear time. *arXiv preprint arXiv:1904.05459*, 2019. 2.1, 3, 3.2.1

[141] M. Kouckỳ and M. E. Saks. Almost linear size edit distance sketch. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 956–967, 2024. 2.2

[142] R. Krauthgamer and J. R. Lee. Navigating nets: simple algorithms for proximity search. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA '2004)*, pages 798–807, 2004. 2.6

[143] R. Krauthgamer, N. Petruschka, and S. Sapir. The power of recursive embeddings for ℓ_p metrics, 2025. 1.2.3

[144] R. Krauthgamer and Y. Rabani. Improved lower bounds for embeddings into L_1. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1010–1017, 2006. 1.2.3, 1.2.3, 2.7, 7

[145] D. Kush, A. Nikolov, and H. Tang. Near neighbor search via efficient average distortion embeddings. In K. Buchin and É. C. de Verdière, editors, *Proceedings*

of the ACM Symposium on Computational Geometry (SoCG), volume 189 of *LIPIcs*, pages 50:1–50:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 1.2.3, 1.2.3, 1.2.3, 7.1.2, 7.5, 7.5

[146] W. Kuszmaul. Efficiently approximating edit distance between pseudorandom strings. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1165–1180. SIAM, 2019. 2.3

[147] G. M. Landau, E. W. Myers, and J. P. Schmidt. Incremental string comparison. *SIAM J. Comput.*, 27(2):557–582, 1998. 1.1

[148] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals (in russian). *Doklady Akademii Nauk SSSR*, 4(163):845–848, 1965. Appeared in English as: V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady* 10(8), 707–710, 1966. 1.2.3

[149] C. L. Mallows. Patience sorting. *Bull. Inst. Math. Appl.*, 9:216–224, 1973. 1.1

[150] W. J. Masek and M. Paterson. A faster algorithm computing string edit distances. *J. Comput. Syst. Sci.*, 20(1):18–31, 1980. 1.1, 5

[151] J. Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93(1):333–344, Dec 1996. 3.2.2, 3.3, 21

[152] J. Matoušek. On the distortion required for embedding finite metric spaces into normed spaces. *Israel Journal of Mathematics*, 93:333–344, 1996. 7.1.3, 114

[153] J. Matoušek and A. Naor. Open problems on embeddings of finite metric spaces. August 2011. Available at http://kam.mff.cuni.cz/~matousek/metrop.ps. 1.2.3, 1.2.3, 2.7, 7

[154] S. McCauley. Approximate similarity search under edit distance using locality-sensitive hashing. *arXiv preprint arXiv:1907.01600*, 2019. 1.2.3

[155] S. McCauley. Approximate similarity search under edit distance using locality-sensitive hashing. In K. Yi and Z. Wei, editors, *24th International Conference on Database Theory, ICDT 2021, March 23-26, 2021, Nicosia, Cyprus*, volume 186 of *LIPIcs*, pages 21:1–21:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. 2.7

[156] J. McNames. A fast nearest-neighbor algorithm based on a principal axis search tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):964–976, 2001. 2.6

[157] M. Mendel and A. Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007. Extended abstract appeared in FOCS 2006. 3.2.4

[158] M. Mendel and A. Naor. Nonlinear spectral calculus and super-expanders. *Publications mathématiques de l'IHÉS*, 119(1):1–95, 2014. 1.2.3

[159] M. Mirzanezhad. On approximate near-neighbors search under the (continuous) fréchet distance in higher dimensions. *Information Processing Letters*, 183:106405, 2024. 2.7

[160] M. Mitzenmacher and S. Seddighin. Dynamic algorithms for lis and distance to monotonicity. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, page 671684, New York, NY, USA, 2020. Association for Computing Machinery. 2.4, 4.2.3

[161] M. Mitzenmacher and S. Seddighin. Improved sublinear time algorithm for longest increasing subsequence. In *Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '21, page 19341947, USA, 2021. Society for Industrial and Applied Mathematics. 1.1, 4.2

[162] R. Motwani, A. Naor, and R. Panigrahy. Lower bounds on locality sensitive hashing. *SIAM Journal on Discrete Mathematics*, 21(4):930–935, 2007. 1.2

[163] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *Proceedings of the 4th International Conference on Computer Vision Theory and Applications (VISAPP '2009)*, pages 331–340, 2009. 2.6

[164] E. W. Myers. An $O(ND)$ difference algorithm and its variations. *Algorithmica*, 1(2):251–266, 1986. 1.1, 3.2.1

[165] A. Naor. Comparison of metric spectral gaps. *Analysis and Geometry in Metric Spaces*, 2(1), 2014. 1.2.3, 1.2.3, 1.2.3, 7.1.2, 7.1.4

[166] A. Naor. An average john theorem. *Geometry & Topology*, 25(4):1631–1717, 2021. 1.2.3, 8, 1.2.3

[167] A. Naor and Y. Rabani. On approximate nearest neighbor search in ℓ_p, $p > 2$. Manuscript, 2006. 1.2.3, 7.1.2

[168] A. Naor and G. Schechtman. Planar earthmover is not in L_1. *SIAM Journal on Computing*, 37(3):804–826, 2007. An extended abstract appeared in FOCS'06. 2.7, 7.1.4

[169] T. Naumovitz and M. Saks. A polylogarithmic space deterministic streaming algorithm for approximating distance to monotonicity. In *Proceedings of the 2015 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1252–1262, 2015. 2.5

[170] T. Naumovitz, M. Saks, and C. Seshadhri. *Accurate and Nearly Optimal Sublinear Approximations to Ulam Distance*, pages 2012–2031. 2017. 2.5

[171] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001. 1.1

[172] I. Newman and N. Varma. New Sublinear Algorithms and Lower Bounds for LIS Estimation. In N. Bansal, E. Merelli, and J. Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 100:1–100:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. 1.1, 4.2

[173] N. S. Nosatzki. Approximating the longest common subsequence problem within a sub-polynomial factor in linear time. *arXiv preprint arXiv:2112.08454*, 2021. 1.1, 1.1, 2.1

[174] R. O'Donnell, Y. Wu, and Y. Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Transactions on Computation Theory*, 6(1):5, 2014. 1.2, 1.2.2, 6.3.0.2

[175] R. Ostrovsky and Y. Rabani. Low distortion embedding for edit distance. *J. ACM*, 54(5), 2007. Preliminary version appeared in STOC'05. 1.1, 1.2.3, 1.2.3, 2.7, 6, 7

[176] R. Ostrovsky and Y. Rabani. Low distortion embeddings for edit distance. *Journal of the ACM (JACM)*, 54(5):23–es, 2007. 1.2.3

[177] List of open problems in sublinear algorithms: Problem 44. http://sublinear.info/44. 2.2

[178] R. K. S. Pallavoor, S. Raskhodnikova, and E. Waingarten. *Approximating the Distance to Monotonicity of Boolean Functions*, pages 1995–2009. 2020. 1.1

[179] R. Panigrahy, K. Talwar, and U. Wieder. Lower bounds on near neighbor search via metric expansion. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '2010)*, pages 805–814, 2010. 1.2.3, 7.1.2

[180] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012 – 1042, 2006. 1.1

[181] E. Porat and B. Porat. Exact and approximate pattern matching in the streaming model. In *FOCS 2009*, pages 315–323. IEEE, 2009. 2.5

[182] Y. Rabinovich. On average distortion of embedding metrics into the line and into L_1. In *35th Annual ACM Symposium on Theory of Computing*. ACM, 2003. 1.2.3

[183] C. Rashtchian, K. Makarychev, M. Rácz, S. Ang, D. Jevdjic, S. Yekhanin, L. Ceze, and K. Strauss. Clustering billions of reads for dna data storage. In *Advances in Neural Information Processing Systems*, pages 3360–3371, 2017. 1.2.3

[184] A. Rubinstein. Hardness of approximate nearest neighbor search. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1260–1268. ACM, 2018. 1.2

[185] A. Rubinstein, S. Seddighin, Z. Song, and X. Sun. Approximation algorithms for lcs and lis with truly improved running times. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1121–1145. IEEE, 2019. 1.1, 1.1, 2.1, 4.2, 4.2, 4.2.1, 4.2.3, 4.2.4, 4.6.2.1, 5

[186] A. Rubinstein and Z. Song. Reducing approximate longest common subsequence to approximate edit distance. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1591–1600. SIAM, 2020. 2.1, 5

[187] S. C. Sahinalp. Edit distance under block operations. In M.-Y. Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008. 1.1

[188] M. Saks and C. Seshadhri. Space efficient streaming algorithms for the distance to monotonicity and asymmetric edit distance. In *Proceedings of the*

Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '13, page 16981709, USA, 2013. Society for Industrial and Applied Mathematics. 2.5

[189] M. E. Saks and C. Seshadhri. Estimating the longest increasing sequence in polylogarithmic time. *SIAM J. Comput.*, 46(2):774–823, 2017. 1.1, 2.5

[190] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009. 2.6

[191] G. Shakhnarovich, T. Darrell, and P. Indyk, editors. *Nearest Neighbor Methods in Learning and Vision*. Neural Processing Information Series, MIT Press, 2006. 1.2

[192] C. Silpa-Anan and R. Hartley. Optimised KD-trees for fast image descriptor matching. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '2008)*, pages 1–8, 2008. 2.6

[193] R. F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6(1–6):579–589, 1991. 2.6

[194] X. Sun and D. P. Woodruff. The communication and streaming complexity of computing the longest common and increasing subsequences. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, page 336345, USA, 2007. Society for Industrial and Applied Mathematics. 1.1, 5

[195] M. Thorup and U. Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, Jan. 2005. 3.2.2

[196] A. Tiskin. Semi-local string comparison: Algorithmic techniques and applications. *Mathematics in Computer Science*, 1(4):571–603, 2008. 3.1

[197] C. D. Toth, J. O'Rourke, and J. E. Goodman, editors. *Handbook of discrete and computational geometry*. Chapman and Hall/CRC, third edition, 2017. 1.2

[198] E. Ukkonen. Algorithms for approximate string matching. *Information and control*, 64(1-3):100–118, 1985. 1.1, 3.2.1

[199] J. Ullman, A. Aho, and D. Hirschberg. Bounds on the complexity of the longest common subsequence problem. *Journal of the ACM (JACM)*, 23(1):1–12, 1976. 5

[200] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13, 2015. 1.2.1

[201] N. Verma, S. Kpotufe, and Sanjoy. Which spatial partition trees are adaptive to intrinsic dimension? In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence (UAI '2009)*, 2009. 2.6

[202] R. A. Wagner and M. J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168 – 173, 1974. 5

[203] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data - a survey. Available as arXiv:1509.05472, 2015. 1.2.1, 2.6

[204] J. Wang, H. T. Shen, K. Song, and J. Ji. Hashing for similarity search: a survey. Available as arXiv:1408.2927, 2014. 1.2.1, 2.6

[205] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proceedings of Advances in Neural Information Processing Systems 21 (NIPS '2008)*, pages 1753–1760, 2008. 2.6

[206] J. Yagnik, D. Strelow, D. Ross, and R.-S. Lin. The power of comparative reasoning. In *Proceedings of the 13th International Conference on Computer Vision (ICCV '2011)*, 2011. 2.6