Incremental Packing Problems: Algorithms and Polyhedra

Lingyi Zhang

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

2022

Abstract

Incremental Packing Problems: Algorithms and Polyhedra

Lingyi Zhang

In this thesis, we propose and study discrete, multi-period extensions of classical packing problems, a fundamental class of models in combinatorial optimization. Those extensions fall under the general name of incremental packing problems. In such models, we are given an added time component and different capacity constraints for each time. Over time, capacities are weakly increasing as resources increase, allowing more items to be selected. Once an item is selected, it cannot be removed in future times. The goal is to maximize some (possibly also time-dependent) objective function under such packing constraints. Below, we further elaborate upon such problems studied in this thesis.

The *generalized incremental knapsack* problem is a multi-period extension of the classical knapsack problem. In this setting, we are given a set of n items, T time periods with non-decreasing capacities $W_1 \leq \cdots \leq W_T$. Each item i has a non-negative weight w_i and item-time profit p_{it} for each $t \in [T]$. If an item i is inserted into the knapsack at time t, it earns a profit of p_{it} and must remain in the knapsack for all future times. The goal is to maximize the overall profit while satisfying the knapsack capacity constraint at each time.

In the *monotone submodular all-or-nothing incremental knapsack* problem (`IK-AoN`), we are given identical feasibility constraints as the generalized incremental knapsack problem. The difference in this setting is that for every item i, we are given a unique profit $p_i > 0$. For every time t, we are given a time-dependent scaling parameter $\Delta_t \geq 0$. Furthermore, we are given a monotone

submodular profit function γ where for every $S \subseteq [n]$, $\gamma(S)$ gives the profit of S. Having a set S in the knapsack at time t earns profit $\Delta_t \gamma(S)$. Finally, we assume the presence of an item i in a set S either earns the whole profit contribution p_i or 0. That is, $\gamma(S \cup \{i\}) - \gamma(S) \in \{0, p_i\}$.

The *incremental generalized assignment* problem is a multi-period extension of the classical generalized assignment problem. We are given n items to be assigned to m bins, T time periods where the capacity of each and every bin is non-decreasing over time. If an item i is assigned to bin j starting at time t, it takes up capacity $w_{i,j}$ for all times $\tau \geq t$ and earns profit p_{ijt}. The goal is to find an item to bin assignment such that the overall profit is maximized and the capacity for each bin is respected for each time.

In Chapter 1, we give a detailed formulation of each of the problems mentioned above, as well as an overview of the contributions of this thesis and a review of related literature.

In Chapter 2, we present a policy that reduces the generalized incremental knapsack problem to sequentially solving T classical knapsack problems, for which many efficient algorithms are known. We call such an algorithm a *single-time* algorithm. We prove that this algorithm gives a $(0.17 - \epsilon)$-approximation for the generalized incremental knapsack problem. Moreover, we show that the algorithm is very efficient in practice. On randomly generated instances of the generalized incremental knapsack problem, it returns near optimal solutions and runs much faster compared to Gurobi solving the problem using the standard integer programming formulation.

In Chapter 3, we present additional approximation algorithms for the generalized incremental knapsack problem. We first give a polynomial-time $(\frac{1}{2} - \epsilon)$-approximation, improving upon the approximation ratio given in Chapter 2. This result is based on a new reformulation of the generalized incremental knapsack problem as a single-machine sequencing problem, which is addressed by blending dynamic programming techniques and the classical Shmoys-Tardos algorithm for the generalized assignment problem [60]. Using the same sequencing reformulation, combined with further enumeration-based self-reinforcing ideas and new structural properties of nearly-optimal solutions, we give a quasi-polynomial time approximation scheme for the problem, thus ruling out the possibility that the generalized incremental knapsack problem is APX-hard under widely-

believed complexity assumptions.

In Chapter 4, we first turn our attention to IK−AoN. We show that each instance of IK−AoN can be reduced to a linear version of the problem. In particular, using a known PTAS for the linear version [4] as a subroutine, this implies that IK−AoN admits a PTAS. Next, we study special cases of the generalized incremental knapsack problem. Using guessing and LP-rounding techniques, we give a PTAS for the generalized incremental knapsack problem if T is bounded. Finally, adapting classical dynamic programming ideas to the multi-time setting, we give an FPTAS for the generalized incremental knapsack problem when we assume for each $i \in [n]$, $p_{i,t} > 0$ for exactly one time $t \in [T]$.

In Chapter 5, we give a polynomial-time $(\frac{1}{4} - \epsilon)$-approximation in expectation for the incremental generalized assignment problem. To develop this result, similar to the reformulation from Chapter 3, we reformulate the incremental generalized assignment problem as a multi-machine sequencing problem. Following the reformulation, we show that the $(\frac{1}{2} - \epsilon)$-approximation for the generalized incremental knapsack problem, combined with further randomized rounding techniques, can be leveraged to give a constant factor approximation in expectation for the incremental generalized assignment problem.

In Chapter 6, we turn our attention to the incremental knapsack polytope. First, we extend one direction of Balas's characterization of $0/1$-facets of the knapsack polytope [6] to the incremental knapsack polytope. Starting from extended cover inequalities valid for the knapsack polytope, we show how to strengthen them to define facets for the incremental knapsack polytope. In particular, we prove that under the same conditions for which these inequalities define facets for the knapsack polytope, following our strengthening procedure, the resulting inequalities define facets for the incremental knapsack polytope. Then, as there are up to exponentially many such inequalities, we give separation algorithms for this class of inequalities.

Table of Contents

List of Figures

List of Tables

Acknowledgements

First and foremost, I need to thank my advisor Yuri Faenza, who taught me how to be a better researcher and a more careful and thoughtful person. Thank you for your patience and kindness and for being my biggest source of support throughout my entire PhD career. I am so fortunate to have had an advisor as attentive and caring as you for the past five years. You have taught me more than what I can fit in this acknowledgement, but for a subset of these things, see the remainder of the thesis. You have made this journey both challenging and fun. Without you, this thesis would not have been possible.

A big thank you to Donniell Fishkind. Thank you, first of all, for being the first person to teach me anything about combinatorial optimization. Your undergraduate classes piqued my initial interest in operations research. Beyond the classes, you helped me through my many moments of doubts and setbacks in trying to pursuing a graduate degree. In so many ways, my PhD career would never have happened without your unwavering belief, support and guidance. Your advice has never failed me, whether I knew it at the time or not.

Thank you to my thesis committee, Daniel Bienstock, Vineet Goyal, Jay Sethuraman, and Huseyin Topaloglu. I am grateful for your feedback and help that made this thesis possible. Thank you to my coauthor, Danny Segev, for your inspiring ideas and diligence.

To Yeyuan, whom I have annoyed with math problems that are only interesting to me for over a decade. Thank you for putting up with me, for humoring me, and for feigning enthusiasm in all the convoluted, irrelevant topics I refuse to stop talking about. However, if you think I will stop now, I regret to inform you that you are wrong.

Thank you to the IEOR department staff for providing a friendly, collaborative and fun environment. I will deeply miss all of our social events, particularly those pre-pandemic. A special shout out to Lizbeth, Kristen, and Winsor for your help and support. Thank you to all of my friends in the department whom I shared this PhD journey with, Shatian, Oussama, Steven, Tugce, Agathe Xuan, Ruizhe, Jacob, Harsh, and many more... Thank you for all the times we've studied together and helped and supported each other through the last 5 years. Thank you to my first math buddy, Tony. When I started working on optimization, a lot of the joy and interest I had came from working with you. Everything I did subsequently came in part from trying to replicate the experience we had.

Last but not least, to my family. A special thanks to my mom for raising me as a single mother for most of my life and for giving me every advantage I needed to be here. Thank you for always challenging me to do more but supporting me even when I couldn't.

Chapter 1: Incremental packing problems

1.1 Introduction

Packing problems are a fundamental class of combinatorial optimization models. In its most general setting, a packing problem requires a decision-maker to choose a subset of items from a finite ground set. The objective function usually rewards selecting additional items, while feasibility requirements impose the opposite constraint, limiting the amount of items that can be chosen. Due to this adaptable formulation, packing problems are employed in modeling a variety of settings in optimization and operations research. Spanning from fundamental graph-theoretical structures like matching and independent (stable) set, to industrial applications like cutting stock and pallet loading, research on packing problems has helped developing many a fundamental technique, as well as tightening the relationship between theory and applications [23, 49, 54].

In a typical packing problem, all items to be selected are chosen simultaneously. However, classic, statical optimization problems are often too simple to realistically capture real world scenarios. With the improvement in computing power, much research has therefore been devoted to extend fundamental well-studied models to more realistic, yet still algorithmically tractable settings. A very common extension along these lines introduces a time-dependent component. For instance, *maximum flow over time*, originally introduced in the seminal work of Ford and Fulkerson [31], has recently received a great deal of attention [37, 52, 61]. Additional examples for such settings include time-expanded versions of various packing problems [1, 14, 24], network scheduling over time [2, 8], adaptive routing over time [33, 43], and facility location over time [27, 55], just to mention a few. Multi-period extensions naturally capture real-world scenarios where assumptions and constraints may change over time. Theoretically, they add a computationally-challenging layer on top of the inherent complexity of the underlying problem.

Building upon these motivations, in this thesis, we investigate incremental versions of various packing problems. Much recent research has been devoted to problems of this nature, particularly for the incremental knapsack problem, a multi-period extension of the classical knapsack problem (see Section 1.4.1 for references). This thesis expands upon existing research by proposing generalizations of incremental knapsack problems, as well as approximation algorithms and polyhedral results for these new problems.

Before giving a formal definition of incremental knapsack problems, to provide initial intuition for the inner-workings of our models, consider the problem faced by urban planners, who intend to build infrastructural facilities over the course of several years, under budget constraints. Once an infrastructure has been built, its construction cost cannot be recovered. Given each infrastructure's annual contribution to welfare once it is in place, the goal is to maximize the total benefit over the course of the planning horizon (hence, the mayor's chances of being re-elected). In Figure 1.1, we give a pictorial example of a problem in this setting. A host of additional applications, such as planning the incremental growth of highways and networks, community development, and memory allocation can be found within several of the papers mentioned in Section 1.4.1 and the references therein.

As our first set of results, we introduce and study the *generalized incremental knapsack* problem. As we discuss in Section 1.4.1, this problem subsumes as special cases, to the best of our knowledge, all other incremental knapsack problems studied thus far. In Chapters 2 and 3 we present different algorithms for the generalized incremental knapsack problem, and investigate their merits and drawbacks. In Chapter 4, we give polynomial time approximation schemes to special cases of this problem. Finally, in Chapter 6, we study the generalized incremental knapsack problem from a polyhedral perspective and define a class of facet-defining inequalities for the incremental knapsack polytope.

As our second set of results, we present a non-linear extension of the incremental knapsack problem. To give an initial motivation, in the incremental knapsack setting, the objectives considered by the generalized incremental knapsack problem and other previously studied problems

Figure 1.1: In this example, the bridge, school, and park represent three potential infrastructure projects. The different colored bars underneath represent their respective costs. There are $T = 2$ discrete times. We indicate two different possible feasible solutions where the green bar represents any unused budget in each time.

have always been linear, meaning every item earns some fixed profit or utility, independent of other items selected. However, in many scenarios, choosing multiple items that are close substitutes of each other gives much less total utility than the sum of the utility of the individual items. Going back to the urban planning example, building two parks in close proximity to each other would earn less total utility than the sum of the utility of each of them built individually. Submodular functions capture profit functions of this nature, and have a wide range of applications, including in machine learning [47], viral marketing [46], scheduling [53], facility location [32], and many more [50]. We extend a special case of monotone submodular function maximization under a knapsack constraint to a multi-period setting. In this special case, in addition to the profit function being monotone and submodular, we associate each item with a specific profit. Adding an item to a set either earns the full profit, or no profit at all. We call this the *monotone submodular all-or-nothing incremental knapsack* problem, which we will formally define in Section 1.2.2. In Chapter 4, we show that

approximation results for the linear variant of the problem transfer to this non-linear setting.

Finally, as our final set of results, we initiate the study a multi-period extension of the generalized assignment problem, which we call the *incremental generalized assignment* problem. The classical generalized assignment problem is a multi-bin generalization of the knapsack problem where n items are assigned to m capacitated bins. The problem has been extensively studied (see Section 1.4.2) and has a wide range of applications such as scheduling, telecommunication, facility location, transportation and production planning; see, e.g., [57]. One such natural application appears in production planning, where jobs are assigned to different plants for production, subject to capacity constraints of each plant. Over a certain planning horizon, expansions increase the capacity of each plant, allowing more jobs to be assigned to it. The assignment of more jobs naturally translate to the ability to earn more profit over the planning horizon. We give the formal problem description in Section 1.2.3. In Chapter 5, we give a constant-factor approximation for this problem in polynomial time.

1.2 The main models

1.2.1 The generalized incremental knapsack problem

To model multi-period extensions of the classical knapsack problem, we are given a set of n items, each with a strictly positive weight $\{w_i\}_{i\in[n]}$, a collection of T times with non-decreasing capacities $\{W_t\}_{t\in[T]}$, i.e., such that $W_1 \leq \cdots \leq W_T$. We say that a sequence of item sets $\mathcal{S} = (S_1, \ldots, S_T)$ is a *chain* when $S_1 \subseteq \cdots \subseteq S_T \subseteq [n]$; here, S_t represents the subset of items inserted into the knapsack up to and including time t. As such, the chain $\mathcal{S}$ is *feasible* when $w(S_t) \leq W_t$ for every $t \in [T]$. Problems with these feasibility conditions are collectively known as *incremental knapsack problems*. For each item $i \in [n]$ and time $t \in [T]$, we are further given an item-time profit, on which we will further elaborate below.

In the *generalized incremental knapsack* problem, our fundamental assumption is that, for each item $i \in [n]$ and time $t \in [T]$, we are given a non-negative profit parameter $p_{i,t}$. An item i earns profit $p_{i,t}$ if it is inserted at time t (i.e., when $i \in S_t \setminus S_{t-1}$, with the convention that $S_0 = \emptyset$).

Hence, the cumulative profit of any chain $\mathcal{S} = (S_1, \ldots, S_T)$ over all time periods is captured by $\Phi(\mathcal{S}) = \sum_{t\in[T]} \sum_{i\in S_t \setminus S_{t-1}} p_{i,t}$. The goal is to find a feasible chain $\mathcal{S}$ that maximizes $\Phi(\mathcal{S})$. The generalized incremental knapsack problem generalizes the time-invariant incremental knapsack problem (as studied in [7, 25, 40, 59]) and the incremental knapsack problem (as studied in [4, 21, 22, 64]). We will discuss both problems and the relevant results in detail in Section 1.4.1.

Throughout this thesis, it is often useful to consider an integer programming formulation for the generalized incremental knapsack problem. For each $t \in [T]$ and $i \in [n]$, let $x_{i,t}$ be binary variables such that $x_{i,t} = 1$ if an item i is in the knapsack at time t, $x_{i,t} = 0$ otherwise. Let $x_{i,0} = 0$ for each $i \in [n]$. This leads to the following IP formulation for the generalized incremental knapsack problem:

$$\begin{aligned}
\max \quad & \sum_{i\in[n]} \sum_{t\in[T]} p_{i,t}(x_{i,t} - x_{i,t-1}) & \\
\text{s.t.} \quad & \sum_{i\in[n]} w_i x_{i,t} \le W_t & \forall\, t \in [T] \\
& x_{i,t} \le x_{i,t+1} & \forall\, i \in [n], t \in [T-1] \\
& x_{i,t} \in \{0,1\} & \forall\, i \in [n], t \in [T].
\end{aligned} \tag{GIK-IP}$$

Given any feasible x to (GIK-IP), for every $t \in [T]$, taking $S_t = \{i \in [n] : x_{i,t} = 1\}$ gives a feasible chain to the generalized incremental knapsack problem. Indeed, the constraints $\sum_{i\in[n]} w_i x_{i,t} \le W_t$ guarantee that $w(S_t) \le W_t$ for every $t \in [T]$; while the constraints $x_{i,t} \le x_{i,t+1}$ guarantee that $S_1 \subseteq \cdots \subseteq S_T$. Vice versa, given a feasible chain $\mathcal{S} = (S_1, \ldots, S_T)$, we can obtain a feasible solution to (GIK-IP) by setting $x_{i,t} = 1$ if $i \in S_t$ for any $i \in [n]$ and $t \in [T]$, $x_{i,t} = 0$ otherwise.

1.2.2 Monotone submodular all-or-nothing incremental knapsack problem

In the *monotone submodular all-or-nothing incremental knapsack* problem (`IK-AoN`), we assume the same feasibility setting as other incremental knapsack problems, with the following assumptions on the objective function. We are given a profit parameter $p_i > 0$ for every item $i \in [n]$ and a scaling factor $\Delta_t \ge 0$ for every $t \in [T]$. For every set $S \subseteq [n]$ we let $\Delta_t \gamma(S)$ be the profit

associated to set S at time t. We assume that the function $\gamma : 2^{[n]} \to \mathbb{N}$ satisfies the following properties:

1. (*Monotone Submodularity*) γ is a monotonically non-decreasing submodular function, that is, for $i \in [n]$ and $S \subseteq T \subseteq [n]$, we have $\gamma(S \cup \{i\}) - \gamma(S) \geq \gamma(T \cup \{i\}) - \gamma(T)$.

2. (*All-or-Nothing Contributions*) for each $i \in [n]$ and $S \subseteq [n] \setminus \{i\}$, we have $\gamma(S \cup \{i\}) - \gamma(S) \in \{0, p_i\}$.

Hence, the presence of item i in S at time t either realizes the "full profit" of $\Delta_t p_i$, or no profit at all. The goal is to find a feasible chain $\mathcal{S} = (S_1, \ldots, S_T)$ that maximizes $\Phi(\mathcal{S}) = \sum_{t \in T} \Delta_t \gamma(S_t)$.

It is easy to see that `IK-AoN` captures linear profits by letting $\gamma(S) = \sum_{i \in S} p_i$ for all $S \subseteq [n]$. Additionally, `IK-AoN` allows us to capture problems with nonlinear profit structures. For example, the matroid rank function satisfies the monotone submodular all-or-nothing profit properties. Hence, `IK-AoN` subsumes both linear profits and more combinatorial ones, making it incomparable to the generalized incremental knapsack problem.

1.2.3 Incremental generalized assignment problem

Before defining the incremental generalized assignment problem, we first give the definition of the *generalized assignment* problem, a "static" generalization to the classical knapsack problem. In this setting, we are given n items and m bins. For each $j \in [m]$, bin j has capacity W_j. Assigning an item i to a bin j takes up capacity w_{ij} while generating a profit of p_{ij}. The goal is to compute a feasible item-to-bin assignment whose overall profit is maximized. It is easy to see that when $m = 1$, this problem reduces to the classical knapsack problem. The generalized assignment problem has been well studied, as seen in [17, 29, 30, 56, 60], just to name a few.

We extend the generalized assignment problem to a multi-period setting, along the same lines of the incremental knapsack problems. Specifically, in addition to the parameters of the generalized assignment problem, we are given a collection of T times, where for each bin $j \in [m]$, we have non-decreasing capacities $\{W_{j,t}\}_{t \in [T]}$, i.e., such that $W_{j,1} \leq \cdots \leq W_{j,T}$. We denote a feasible

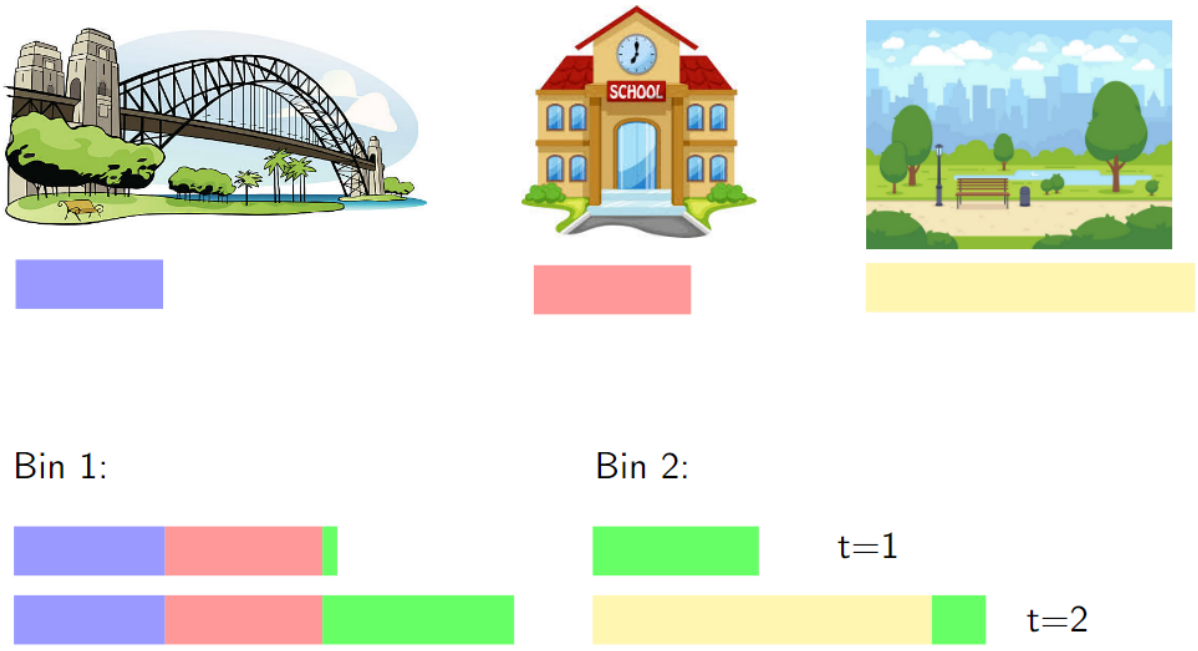

Figure 1.2: continued from Figure 1.1. In this example, we have the same three items as Figure 1.1 but now two different bins to assign these items. We show a possible feasible solution.

solution in the form of m chains $\mathcal{S}_1, \ldots, \mathcal{S}_m$. For each $j \in [m]$, we have $\mathcal{S}_j = (S_{j,1}, \ldots, S_{j,T})$ where $S_{j,1} \subseteq \cdots \subseteq S_{j,T} \subseteq [n]$. Analogous to the incremental knapsack setting, the set $S_{j,t}$ represents the subset of items inserted into the j-th bin up to and including time t. Feasibility of each chain necessitates that for each $j \in [m]$ and $t \in [T]$, we have $w_j(S_{j,t}) \leq W_{j,t}$. Furthermore, we require that each item is assigned to at most one bin, that is, $S_{j,T} \cap S_{j',T} = \emptyset$ for each distinct $j, j' \in [m]$. In Figure 1.2, we give a pictorial example of a problem in this setting.

Finally, we are given non-negative profit parameters $p_{i,j,t}$ for each $i \in [n], j \in [m]$ and $t \in [T]$. Item i earns profit $p_{i,j,t}$ if it is inserted in bin j at time t (i.e., if $i \in S_{j,t} \setminus S_{j,t-1}$ where $S_{j,0} = \emptyset$ for every $j \in [m]$). The profit earned by the chain $\mathcal{S}_j$ in bin j is therefore given by $\Phi_j(\mathcal{S}_j) = \sum_{t \in [T]} \sum_{i \in S_{j,t} \setminus S_{j,t-1}} p_{i,j,t}$. The objective is to find feasible chains $\mathcal{S}_1, \ldots, \mathcal{S}_m$ that maximizes $\sum_{j \in [m]} \Phi_j(\mathcal{S}_j)$, the total profit of chains summed across all bins.

1.3 Main contributions

In this section, we give a brief outline of this thesis and our main contributions in each chapter.

Chapter 2: Single-time policies for the generalized incremental knapsack problem. It is natural to try reducing the generalized incremental knapsack problem to the classical knapsack problem – although the latter is only weakly NP-Hard, and the former is strongly NP-Hard [7]. As our first contribution, we show that indeed some kind of reduction to the classical knapsack problem is possible. More formally, we present what we call a single-time algorithm for the generalized incremental knapsack problem with a constant-approximation bound on its performance guarantee. The main idea behind single-time algorithms (introduced for the first time in this thesis) is to reduce an incremental packing problem to a sequence of "static" packing problems on the same ground set, which can then be solved using known techniques. This approach has two interesting features: first, classical packing problems are more studied, and many techniques to find good solutions are known. Second, the dimension of the problem is reduced dramatically, since the number of variables of each of the resulting classical packing problems do not depend on T. The main result of this chapter is given below.

Theorem 1.3.1. *For any accuracy level $\epsilon > 0$, there exists a single-time algorithm that approximates the generalized incremental knapsack problem within factor $0.17 - \epsilon$ in time $\tilde{O}(Tn + T(\frac{T}{\epsilon})^{\frac{9}{4}})$.*

In addition to the theoretical guarantee, we also test our algorithm in practice. We show in Section 2.5 that on randomly generated instances of the generalized incremental knapsack problem, our algorithm gives near-optimal solutions, far better than the theoretical lower bound given in Theorem 1.3.1. Due to its feature of reducing the incremental packing problem to complexity-wise much simpler static packing problems, it runs much faster compared to Gurobi solving the problem using the standard integer programming formulation. As another added benefit of reducing the dimension of the problem, we show that it is capable of solving much larger instances than Gurobi.

The work in this chapter is based on an ongoing work in collaboration with Yuri Faenza and

Danny Segev.

Chapter 3: Algorithms for the generalized incremental knapsack problem through a sequencing reformulation. In this chapter, we provide additional constant factor approximations for the generalized incremental knapsack problem, including an improvement upon the factor given in Theorem 1.3.1. The first contribution comes in the form of a polynomial-time constant-factor approximation.

Theorem 1.3.2. *For any error parameter $\epsilon \in (0, \frac{1}{2})$, the generalized incremental knapsack problem can be approximated within factor $\frac{1}{2} - \epsilon$. The running time of our algorithm is $O(n^{O(1/\epsilon^2)} \cdot |\mathcal{I}|^{O(1)})$, where $|\mathcal{I}|$ stands for the input size.*

We note that in addition to the approximation factor being better than that given in Theorem 1.3.1, the algorithm also uses completely different approaches and techniques. The starting point of this algorithm is a new reformulation of the generalized incremental knapsack problem as a *single-machine sequencing* problem, where feasible chains are mapped to permutations. Based on this reformulation, we decompose the optimal permutation into heavy and light parts, depending on how item weights compare against the combined weight of all previously-inserted items. This sequencing reformulation is, to our knowledge, very different from how existing related problems were approached up until now. At a high level, to compete against the heavy part, we exploit dynamic programming ideas, whereas for the light part, we propose a further reformulation as a highly-structured generalized assignment instance, which is handled by leveraging the Shmoys-Tardos algorithm [60] and an additional greedy truncation phase. Consequently, we show that both subproblems can be approximated within factor $1 - \epsilon$ of optimal, thus leading to Theorem 1.3.2.

As we will explain in Section 1.4.1, even seemingly-simple special cases of the generalized incremental knapsack problem are known to be strongly NP-hard, admitting a PTAS under specific profit-structure assumptions. A natural question is whether we one can design efficient algorithms with the same degree of accuracy for the generalized incremental knapsack problem, without mitigating assumptions. Towards this goal, our second main contribution of this chapter establishes

the existence of a QPTAS.

Theorem 1.3.3. *The generalized incremental knapsack problem admits a quasi-polynomial time approximation scheme.*

Hence, unless NP $\subseteq$ DTIME($2^{\text{polylog}\,(n)}$) , this result rules out the possibility that the generalized incremental knapsack problem is APX-hard, thus making it substantially different from other knapsack extensions, such as the generalized assignment problem (see a brief discussion in Section 1.4.2). Finally we observe how those algorithms in practice do not terminate in reasonable times, thus making the algorithm from Section 2 more interesting for real-world applications.

The work in this chapter is in collaboration with Yuri Faenza and Danny Segev and appears in [26].

Chapter 4: Some easier, and some not harder, incremental knapsack problems. Our main result in this chapter provides a PTAS for IK−AoN. For a family C of instances of IK−AoN, let us call their *linearization* the family $\overline{C}$ of IK instances containing all and only the instances obtained as follows: starting from an instance $\mathcal{I} \in C$ with profits p, first drop some of the items, assuming (possibly after renaming) that items $[n'] \subseteq [n]$ are kept; then define a new (linear) profit function γ' such that for every $S \subseteq [n']$, $\gamma'(S) = p(S)$ (or equivalently for every $i \in [n']$ and $S \subseteq [n'] \setminus \{i\}$, $\gamma'(S \cup \{i\}) - \gamma'(S) = p_i$).

Our main result shows, that, somehow surprisingly, any family of IK−AoN instances is not harder than its linearization.

Theorem 1.3.4. *Let C be a family of IK−AoN instances and let $\alpha \in [0, 1]$. If there is an α-approximation algorithm for instances in $\overline{C}$, then there is an α-approximation algorithm for instances in C running in time $O(\text{Time}_\alpha(n, T) + nT)$, where $\text{Time}_\alpha(n, T)$ is the running time of the α-approximation algorithm for instances in $\overline{C}$ with n items and T time periods.*

Theorem 1.3.4 implies, for instance, that IK−AoN has a PTAS (using the PTAS for the incremental knapsack problem [4] as a subroutine). Moreover, if one aims at practical algorithms with

good (though suboptimal) theoretical performance guarantee for IK−AoN, then Theorem 1.3.4 can be employed using as a subroutine the single-time algorithm from Chapter 2, which, as previously mentioned, runs much faster than Gurobi – the additional steps added by our algorithm is only in creating the specific linearized instance, which is computationally very little.

Beyond the study of IK−AoN, in this chapter, we also provide approximation schemes for two special cases of the generalized incremental knapsack problem. In the first case, using classical guessing and LP-rounding techniques, we give a PTAS for the generalized incremental knapsack problem when we assume T is a constant.

Theorem 1.3.5. *For any $\epsilon > 0$, the generalized incremental knapsack problem can be approximated within a factor of $(1-\epsilon)$ in time $O((nT)^{O(\frac{T^3}{\epsilon})})$. Therefore, it admits a PTAS when T is a constant.*

In the second special case, we assume that in the generalized incremental knapsack setting, for each $i \in [n]$, there is a unique time t such that $p_{i,t} > 0$. For all other times $\tau \neq t$, we have $p_{i,t} = 0$. We call this problem the *generalized incremental knapsack single profit* problem. Notice that taking $T = 1$, this problem still subsumes the classical knapsack problem as a special case, and hence is NP-hard. We show that, despite the added time dimension, we can modify standard dynamic programming ideas for the classical knapsack problem to obtain an FPTAS.

Theorem 1.3.6. *For any $\epsilon > 0$, the generalized incremental knapsack single profit problem can be approximated within a factor of $1-\epsilon$ in time $O(\frac{n^3}{\epsilon})$.*

The work in Section 4.2 of this chapter is based on an ongoing work in collaboration with Federico D'Onofrio and Yuri Faenza. The work in Sections 4.3 and 4.4 of this chapter is in collaboration with Yuri Faenza.

Chapter 5: Single-machine algorithms for incremental packing problems. Our main contribution in this chapter is to give a polynomial-time constant factor approximation for the incremental generalized assignment problem.

Theorem 1.3.7. *For any $\epsilon > 0$, there exists a polynomial-time algorithm that in expectation gives a $(\frac{1}{4} - \epsilon)$-approximation to the incremental generalized assignment problem.*

To prove this result, we generalize the scheduling reformulation of the generalized incremental knapsack problem from Chapter 3, and show how general packing problems can be reformulated as multi-machine sequencing problems. Following this reformulation, the $(\frac{1}{2} - \epsilon)$-approximated algorithm from Section 3.2 can be leveraged to obtain a polynomial time approximation algorithm for the incremental generalized assignment problem.

In fact, in order to show the above result, we prove a more general version. We show that any single-bin packing problem can be reformulated as a single-machine sequencing problem. Similarly, any multi-bin packing problem can be reformulated as a multi-machine sequencing problem. Furthermore, if there exists a polynomial-time β-approximation algorithm for the single-machine sequencing problem, then for any $\delta > 0$, there exists a polynomial-time $\frac{1}{2}(\beta - \delta)$-approximation algorithm for its multi-machine extension. The techniques used in proving this approximation result extends the ideas developed in [30] for the separable assignment problem. We defer the technical details to Chapter 5, including a discussion of how our results differ from that of [30] in Section 5.5.

The work in this chapter is based on an ongoing work in collaboration with Yuri Faenza and Danny Segev.

Chapter 6: On the facets of the incremental knapsack polytope. In the final chapter, we give polyhedral results for the incremental knapsack polytope. In [6], Balas gave a complete characterization of facets with $0/1$ coefficients of the knapsack polytope using extended cover inequalities from strong covers. For our first contribution, we give a technique that strengthens extended cover inequalities for the incremental knapsack polytope, which we call *lift and push*. We show that lift and push cover inequalities define facets for the incremental knapsack polytope under the same conditions for which extended cover inequalities define facets for the classical knapsack polytope. Notably, this result extends only one direction of Balas's work. For the other direction,

we show that there exists facets with $0/1$ coefficients for the incremental knapsack polytope that cannot be characterized by extended cover inequalities of strong covers.

Then, we show that separation problem for lift and push cover inequalities can be solved in pseudo-polynomial time.

Theorem 1.3.8. *Given a fractional point $\bar{x} \in [0,1]^{nT}$ that satisfies the linear relaxation of* (GIK-IP)*, the lift and push cover inequality separation problem can be solved in time $O(Tn^4||w||_\infty)$.*

Given that the extended cover inequality separation problem for the knapsack polytope is already NP-hard [20], a polynomial time separation algorithm for the lift and push cover inequalities is not possible unless P = NP. As a final contribution, we show an approximate separation algorithm for lift and push cover inequalities in polynomial time.

Theorem 1.3.9. *Given a fractional point $\bar{x} \in [0,1]^{nT}$ that satisfies the constraints of the linear relaxation of* (GIK-IP)*. For any $\epsilon > 0$, there exists an algorithm that, in time $O(T\frac{n^7}{\epsilon})$, gives a lift and push cover inequality that $\bar{x}$ violates, or concludes that $(1-\epsilon)\bar{x}$ satisfies all such inequalities.*

The work in this chapter is in collaboration with Yuri Faenza.

1.4 Related literature

1.4.1 Directly related incremental knapsack settings

The generalized incremental knapsack problem and `IK-AoN` subsume as special cases several previously-studied incremental knapsack problems, all of which have identical feasibility constraints. Probably the simplest incremental knapsack problem studied so far is *time-invariant incremental knapsack* (`IIK`), where each item i is assumed to contribute a profit of p_i to each period starting at its insertion time.

The generalized incremental knapsack problem captures this profit setting by letting $p_{it} = (T+1-t)\cdot p_i$. From an `IK-AoN` standpoint, to capture the time-invariant incremental knapsack

setting, we take $\Delta_t = 1$ for all $t \in [T]$, and set $p_i = \phi_i$ and $\gamma(S) = p(S)$ for all $S \subseteq [n]$. Surprisingly, unlike the classical knapsack problem, Bienstock et al. [7] showed that this extension is strongly NP-hard. On the positive side, Faenza and Malinovic [25] proposed a polynomial-time approximation scheme (PTAS) based on rounding fractional solutions to an appropriate disjunctive relaxation.

The broader *incremental knapsack* problem (IK) is the linear variant to the IK−AoN problem. Specifically, we are given the same input parameters, p_i for every $i \in [n]$ and Δ_t for every $t \in [T]$. Since the profit structure is linear, for all $S \subseteq [n]$, we let $\gamma(S) = p(S) = \sum_{i \in S} p_i$. Thus, inserting item i always realizes the full profit of p_i. From a generalized incremental knapsack problem standpoint, setting $p_{it} = p_i \cdot \sum_{\tau=t}^{T} \Delta_\tau$ captures the incremental knapsack setting. For this problem, Aouad and Segev [4] have recently obtained a PTAS, leveraging on approximate dynamic programming ideas. We refer the reader to a number of additional resources related to incremental knapsack problems [21, 22, 40, 59, 64] for a deeper look into these settings.

In contrast, in the generalized incremental knapsack setting, the flexibility of our item- and time-dependent profit structure allows us to capture a variety of situations. For instance, when an item i gains a profit of $\phi_{i\tau}$ for each period τ, starting at its insertion time, we can set $p_{it} = \sum_{\tau=t}^{T} \phi_{i\tau}$. If, moreover, the per-period profits $\phi_{i\tau}$ are discounted by a factor of $c_{\tau-t}$ after $\tau - t$ time units have elapsed since the insertion of item i, we set $p_{it} = \sum_{\tau=t}^{T} c_{\tau-t}\phi_{i\tau}$.

1.4.2 The generalized assignment problem

In this section, we briefly discuss known results of the generalized assignment problem. From a complexity standpoint, even special cases of the problem is shown to be APX-hard [16]. On the positive side, for the minimization variant of the generalized assignment problem, Shmoys and Tardos [60] proposed an LP-based 2-approximation, which was observed by Chekuri and Khanna [16] to be easily adaptable to obtain a $1/2$-approximation for the maximization variant. Interestingly, while it is unclear whether these algorithmic ideas are translatable to the incremental generalized assignment problem setting, they will be useful within one of the subroutines employed

by our approach to solve the generalized incremental knapsack problem (see Section 3.2.4).

Feige and Vondrák [29] attained a $(1 - 1/e + \delta)$-approximation, for some absolute constant $\delta > 0$, which is currently the best known performance guarantee for maximum generalized assignment. Earlier constant-factor approximations were obtained in [17, 30, 56]. The ideas used in [30] inspire our techniques used in solving the incremental generalized assignment problem in Chapter 5.

1.4.3 Submodular function maximization

Submodular function maximization is a classical well-studied problem. Unconstrained submodular function maximization can be approximated to a tight $\frac{1}{2}$ factor [10]. For maximizing monotone submodular function subject to a knapsack constraint, Sviridenko obtained a tight $(1 - \frac{1}{e})$-approximation through a combination of guessing and combinatorial greedy techniques [62]. There is also extensive literature on submodular function maximization subject to various different types of packing constraints, such as cardinality constraint [11], matroid constraint [13], and multiple knapsack or matroid constraints [51].

To the best of our knowledge, submodular function maximization in an incremental knapsack setting has not previously been studied. In this setting, we already know that commonly-used greedy techniques do not work, even when the objective function is linear. Hence, the quest for tools that can solve the incremental version of the problem is open.

1.4.4 Other related packing problems

In the *unsplittable flow on a path* problem, we are given an edge-capacitated path as well as a collection of tasks. Each task is characterized by its own subpath, profit, and demand. The goal is to select a subset of tasks of maximum total profit, under the constraint that the overall demand of the selected tasks along each edge resides within its capacity. Grandoni et al. recently gave a PTAS for the unsplittable flow on a path problem [35], improving upon many earlier constant-factor guarantees [3, 9, 12, 36]. In Appendix A.1, we describe an unfruitful attempt of reducing the generalized incremental knapsack problem to unsplittable flow on a path, explaining what the

main technical issues are. That said, we further present a reduction to a generalization of the latter problem, with so-called "bag constraints". In the latter setting, the best known polynomial-time algorithm attains an approximation factor of $O(\frac{\log \log n}{\log n})$. This result is incomparable to the constant-factor approximation results of this thesis.

1.4.5 Lifting valid inequalities

There is a wealth of literature on valid inequalities for the classical knapsack polytope. These inequalities have shown to be useful in solving large scale linear and integer programming problems in practice [19, 39]. In this section, we will briefly discuss a few classes of such inequalities and techniques used to obtain them. We refer the reader to [42] for a recent survey that provides a thorough discussion on these topics.

Given an instance of the classical knapsack problem with knapsack capacity W, a set of n items, each item with non-negative weight w_i and non-negative profit p_i, the knapsack polytope is defined as the convex hull of the feasible points satisfying the knapsack constraint:

$$\operatorname{conv}\left\{x \in \{0,1\}^n : \sum_{i\in[n]} w_i x_i \leq W\right\}.$$

Given a set $S \subseteq [n]$ and a valid inequality $\sum_{i\in S} \alpha_i x_i \leq \beta$ for the knapsack polytope, the inequality $\sum_{i\in[n]} \alpha_i x_i \leq \beta$ is called a *lifting* of $\sum_{i\in S} \alpha_i x_i \leq \beta$ if it is also valid. The goal of lifting is to compute large coefficients of α_i for $i \notin S$ as to obtain stronger, or even facet-defining, inequalities.

We call $S \subseteq [n]$ a minimal cover if

- $\sum_{i\in S} w_i > W$, and
- for every $j \in S$, $\sum_{i\in S\setminus\{j\}} w_i \leq W$.

Starting with a minimal cover S, Balas showed how to obtain *extended cover inequalities* and gave necessary and sufficient conditions for when these inequalities define a facet [6]. As it is the main

inspiration of our results in Chapter 6, we give a more in-depth look at this work in Section 6.2.

Zemel [65] showed how to obtain a facet-defining *lifted cover inequality* of the form

$$\sum_{i \in S} x_i + \sum_{i \in [n] \setminus S} \alpha_i x_i \leq |S| - 1,$$

through sequential lifting in $O(n|S|)$ time. Other works in lifting resulting in more general lifted inequalities can be found in [39, 58, 63].

1.5 Shared notations

In this section, we give some shared notations that will be used throughout this thesis.

- We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$ and $[n]_0$ to denote the set $\{0, 1, 2, \ldots, n\}$.
- We use $2^{[n]}$ to denote the set containing all subsets of $[n]$.
- Given a chain $\mathcal{S} = (S_1, \ldots, S_T)$, for any $i \in S_T \subseteq [n]$, we use $t(i)$ to denote the *insertion time* of i. That is, $t(i)$ is the unique time t for which $i \in S_t \setminus S_{t-1}$.
- For any function g defined over a set $[n]$, for any $S \subseteq [n]$, we let $g(S) = \sum_{i \in S} g_i$.
- For any $x \in \mathbb{R}$, we let $[x]^+ = \max\{0, x\}$.

Chapter 2: Single-time policies for the generalized incremental knapsack problem

2.1 Introduction and preliminaries

In this chapter, we present single-time algorithms for the generalized incremental knapsack problem. Somewhat informally, the goal of single-time algorithms is to remove the complexity added by the time component of the problem by sequentially solving classical knapsack problems instead, for which many efficient algorithms are known. Since the generalized incremental knapsack problem consists of T distinct knapsack constraints and items with distinct profits when inserted into the knapsack a certain time, it is natural to consider reducing it to T classical knapsack problems.

In Section 2.2, we first show why certain trivial reductions of the generalized incremental knapsack problem to the classical knapsack problem perform poorly. Although not leading to efficient algorithms, these ideas form the basis for the development of the *c-flexible* single-time algorithm presented in Section 2.3. We show that this algorithm gives a constant factor approximation for the generalized incremental knapsack problem in Sections 2.4, proving Theorem 1.3.1. Finally, we test the algorithm in practice in Section 2.5.

Throughout this chapter, we assume that in an instance of the generalized incremental knapsack problem, for every $i \in [n]$, item i has monotonically non-increasing profit in t. That is, we have $p_{i,t} \geq p_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$. The next lemma shows that this assumption can be made without loss of generality. Its proof is provided in Appendix B.1

Lemma 2.1.1. *Given an instance $\mathcal{I}$ of the generalized incremental knapsack problem, in polynomial time, we can reformulate $\mathcal{I}$ as a new instance $\mathcal{I}'$ of the generalized incremental knapsack problem, where $p_{i,t} \geq p_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$. Specifically, any feasible chain $\mathcal{S}$ of*

$\mathcal{I}$ is feasible for $\mathcal{I}'$ and $\Phi'(\mathcal{S}) \geq \Phi(\mathcal{S})$, where Φ and Φ' denote the profit functions of $\mathcal{I}$ and $\mathcal{I}'$ respectively. Conversely, any feasible chain $\mathcal{S}'$ of $\mathcal{I}'$ can be mapped to a feasible chain $\mathcal{S}$ of $\mathcal{I}$ such that $\Phi(\mathcal{S}) = \Phi'(\mathcal{S}')$.

2.2 Rigid and fully-flexible single-time algorithms

A rigid algorithm. Our first single-time algorithm starts by solving the knapsack problem in time 1 with capacity W_1, where each item $i \in [n]$ has profit $p_{i,1}$ and weight w_i. Let S_1 be the solution to this classical knapsack problem. We select S_1 in the generalized incremental knapsack solution at time 1. These items are removed from future knapsack problems, and their weights subtracted from future capacities. The algorithm then iterates by solving the residual knapsack problem in time 2 with capacity $W_2 - w(S_1)$, set of items $[n] \setminus S_1$, where each item $i \in [n] \setminus S_1$ has profit $p_{i,2}$ and weight w_i. Let Q_2 be the solution to this knapsack problem, set $S_2 = S_1 \cup Q_2$ and consider the knapsack problem with items in S_2 removed, and so on. It is easy to see that $(S_1, S_1 \cup Q_2, S_1 \cup Q_2 \cup Q_3, \ldots, S_1 \cup (\cup_{t=2}^{T} Q_t))$ is a feasible chain for the original generalized incremental knapsack problem. This algorithm is *rigid* in the sense that it never reconsiders its previous decisions. As such, it may not insert an item that gives a very large profit, but does not fit early capacity constraints. It is easy to build examples containing such items that lead to an arbitrarily bad approximation ratio. We give such an example in Appendix B.2.

A fully flexible algorithm. At the opposite side of the spectrum of single-time algorithms, consider a *fully flexible* algorithm that removes a set of items inserted at previous times to make room for another set of items if the contribution of the latter set to the overall profit is strictly larger than that of the former set. More formally, S_1 is computed as in the rigid case. In the second round, we define a knapsack problem over all items in $[n]$ with capacity W_2. Here, an item $i \in [n] \setminus S_1$ has profit $p_{i,2}$, whereas an item $i \in S_1$ has profit $p_{i,1} \geq p_{i,2}$. So items from S_1 are also part of the decision process in the second knapsack problem, but their profit is modified to take into account of the fact that they are already part of the chain, hence they give a potentially larger contribution

to the objective function (recall that by Lemma 2.1.1, we have $p_{i,1} \geq p_{i,2}$ for all $i \in [n]$). Given a solution S_2 to this knapsack problem, S_1 is modified by setting $S_1 = S_1 \cap S_2$. Hence, we remove from S_1 items that are not selected in S_2. We can now iterate by defining, in round t, a knapsack problem with capacity W_t, ground set of items $[n]$, where an item $i \in [n] \setminus S_{t-1}$ has profit $p_{i,t}$, while every other item $i \in S_\tau \setminus S_{\tau-1}$ for some $\tau \in [t-1]$ has profit $p_{i,\tau}$ (where $S_0 = \emptyset$ by convention). Once a solution S_t to the t-th round knapsack problem has been computed, we set $S_\tau = S_\tau \cap S_t$ for all $\tau \in [t-1]$. It is easy to see that $(S_1, \ldots, S_T)$ is a feasible chain.

The more sophisticated fully flexible algorithm can also be tricked into giving a bad approximation. In fact, it will always remove a previously inserted item with less profit in order to insert an item with more profit, even if the profit difference is negligible and the more profitable item has a much larger weight. Thus, the algorithm might insert an item with a large weight, making many slightly less profitable items with much smaller weights infeasible over time. As such, it can output an $O(\frac{1}{T})$-approximated solution, even if we can solve the single-time classical knapsack problem optimally. A detailed example where this bound is achieved is given in Appendix B.3.

2.3 c-flexible algorithms

Interpolating between the two algorithms. These two elementary examples suggest that, in order to give a constant factor approximation for the generalized incremental knapsack problem, a single-time algorithm should balance between updating an existing solution greedily and never updating an existing solution. Thus, in the following section, we present a class of single-time algorithms called *c-flexible*, where $c \geq 1$ is a parameter. At each time $t \in [T]$, the algorithm solves a knapsack problem where the profit of items inserted in a previous time is multiplied by a factor of c. Intuitively, the parameter c gives a measure of how "conservative" the algorithm will be: the higher the c, the less likely the algorithm is to remove items it has already inserted. Thus, the rigid algorithm corresponds to $c = +\infty$ and the fully flexible algorithm to $c = 1$. We show that for appropriate choices of $c > 1$, this approach gives a constant-factor approximation for the generalized incremental knapsack problem.

The algorithmic framework. Let $c \geq 1$ and $\epsilon \in (0, 1)$ (possibly a function of c and T) be given. The algorithm constructs T chains $\mathrm{ALG}^{(1)}, \ldots, \mathrm{ALG}^{(T)}$, one for each (classical) knapsack problem it solves, where for $t \in [T]$, $\mathrm{ALG}^{(t)} = (\mathrm{ALG}_1^{(t)}, \mathrm{ALG}_2^{(t)}, \ldots, \mathrm{ALG}_T^{(t)})$.

1. For $t = 1, \ldots, T$:

   (a) Let $\Pi^{(t)}$ be the (classical) knapsack problem with capacity W_t over the set of items $[n]$, where item $i \in [n]$ has weight w_i and profit

   $$p'_i = \begin{cases} c \cdot p_{i,\tau} & \text{if } t \geq 2 \text{ and } i \in \mathrm{ALG}_\tau^{(t-1)} \setminus \mathrm{ALG}_{\tau-1}^{(t-1)} \text{ for some } \tau < t \\ p_{i,t} & \text{otherwise} \end{cases}$$

   (b) Using the FPTAS for knapsack, solve $\Pi^{(t)}$ to $(1 - \frac{\epsilon}{1+\epsilon})$-optimality, i.e., as to obtain solution S so that $(1 + \epsilon)p'(S) \geq p'(S^*)$, where S^* is the optimal solution to $\Pi^{(t)}$.

   (c) If $t = 1$, set $\mathrm{ALG}_\tau^{(t)} = S$ for all $\tau \in [T]$.

   (d) If $t \geq 2$,

   - if $p'(S) \geq p'(\mathrm{ALG}_{t-1}^{(t-1)})$, set

   $$\mathrm{ALG}_\tau^{(t)} = \begin{cases} \mathrm{ALG}_\tau^{(t-1)} \cap S & \text{for all } \tau \in [t-1] \\ S & \text{for } \tau = t, \ldots, T, \end{cases}$$

   - else, set $\mathrm{ALG}_\tau^{(t)} = \mathrm{ALG}_\tau^{(t-1)}$ for all $\tau \in [T]$.

2. Output $\mathrm{ALG}^{(T)} = (\mathrm{ALG}_1^{(T)}, \ldots, \mathrm{ALG}_T^{(T)})$.

Feasibility. The following lemma establishes the feasibility of the solution $\mathrm{ALG}^{(t)}$ after every iteration of Step 1. Its proof can be found in Appendix B.4.1.

Lemma 2.3.1. *For $t \in [T]$, $\mathrm{ALG}^{(t)} = (\mathrm{ALG}_1^{(t)}, \ldots, \mathrm{ALG}_T^{(t)})$ is a feasible chain.*

2.4 Technical overview

2.4.1 Main results

In this section, we give an overview of the theoretical results of the c-flexible algorithm. First, we show that for any constant $c > 1$, the algorithm gives a constant factor approximation to the generalized incremental knapsack problem.

Theorem 2.4.1. *For $c > 1$ and $\epsilon > 0$ the c-flexible algorithm outputs a $(\frac{c-1}{c^2+c} - \epsilon')$-approximation for the generalized incremental knapsack problem in time $\tilde{O}(Tn + T(\frac{1}{\epsilon})^{\frac{9}{4}})$ where $\epsilon' = (c + 1 + \frac{c+1}{c-1} + cT)\epsilon$.*

The next result shows that for any $c > 1$, the algorithm cannot achieve an approximation factor better than $\frac{c-1}{c^2}$. Its proof can be found in Appendix B.4.11.

Theorem 2.4.2. *For every $c > 1$ and $\epsilon > 0$, there is an instance of the generalized incremental knapsack problem where the c-flexible algorithm gives a $(\frac{c-1}{c^2} + f(\epsilon))$-approximation, where $f(\epsilon)$ is some function of ϵ only.*

By optimizing the functions of Theorem 2.4.1 and Theorem 2.4.2 over c, we obtain the following result, which implies Theorem 1.3.1.

Corollary 2.4.3. *For each constant $\epsilon' > 0$, the $(1 + \sqrt{2})$-flexible algorithm gives a $(0.17 - \epsilon')$-approximation algorithm for the generalized incremental knapsack problem in time $\tilde{O}(Tn + T(\frac{cT}{\epsilon'})^{\frac{9}{4}})$. Moreover, for any choice of $c > 1$, the c-flexible algorithm cannot give an approximation factor better than $0.25 + \epsilon'$.*

The rest of the section is organized as follows. We start with an overview of the proof of Theorem 2.4.1 in Section 2.4.2, where we highlight the main ideas and steps, with many of the technical details postponed to Appendix B.4. In Section 2.4.3, we tie the steps together to prove Theorem 2.4.1.

2.4.2 Proof overview

Preliminaries. Let $\mathrm{OPT} = (\mathrm{OPT}_1, \mathrm{OPT}_2, \ldots, \mathrm{OPT}_T)$ denote an optimal chain. Throughout this section, we extend any chain by adding a 0-th set equal to the empty set. This set will be denoted by S_0 for a chain $\mathcal{S} = (S_1, \ldots, S_T)$, by $\mathrm{ALG}_0^{(t)}$ for $\mathrm{ALG}^{(t)}$, by OPT_0 for OPT, etc. We assume that the chain OPT is inclusionwise maximal. That is, for any i, t such that $i \in \mathrm{OPT}_t \setminus \mathrm{OPT}_{t-1}$, we have $w(\mathrm{OPT}_{t-1} \cup \{i\}) > W_{t-1}$. Note that this assumption is without loss generality since we assume $p_{i,t} \geq p_{i,t+1}$ for all $t \in [T-1]$.

For $S \subseteq [n]$ and $t \in [T]$, we let $p_t(S) = \sum_{i \in S} p_{i,t}$ denote the profit set S earns if all its elements are first inserted into the knapsack at time t. For all $t \in [T]$ and all sets $S \subseteq [n]$, $p_t(S)$ is non-negative, monotonically non-decreasing in S, and monotonically non-increasing in t. Recall that, for a chain $\mathcal{S} = (S_1, \ldots, S_T)$, we write

$$\Phi(\mathcal{S}) = \sum_{t=1}^{T} p_t(S_t \setminus S_{t-1}) = \sum_{t=1}^{T} \sum_{i \in S_t \setminus S_{t-1}} p_{i,t}$$

to denote its profit with respect to the original objective function of the problem. For any $t \in [T]$, by construction, we have $\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)} = \emptyset$ for all $j > t$. It then follows that

$$\Phi(\mathrm{ALG}^{(t)}) = \sum_{j=1}^{t} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}) + \sum_{j=t+1}^{T} \underbrace{p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)})}_{=0} = \sum_{j=1}^{t} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}). \tag{2.1}$$

We remark that by Lemma 2.3.1, $\mathrm{ALG}^{(t)} = (\mathrm{ALG}_1^{(t)}, \ldots, \mathrm{ALG}_T^{(t)})$ is a feasible chain for all $t \in [T]$. In particular, the chain $\mathrm{ALG} = \mathrm{ALG}^{(T)}$ output by the algorithm is feasible. We next give an inclusion relationship between $\mathrm{ALG}_j^{(t)}$ and $\mathrm{ALG}_j^{(t-1)}$ for all j, t such that $j < t$. Its proof can be found in Appendix B.4.2.

Lemma 2.4.4. *For $t \in [T]$ and $t \geq 2$ and all $j \in [t-1]_0$, we have $\mathrm{ALG}_j^{(t)} \subseteq \mathrm{ALG}_j^{(t-1)}$.*

When $\Pi^{(t)}$ is solved in Step 1, some items that are in the chain $\mathrm{ALG}^{(t-1)}$ may not appear in the newly constructed chain $\mathrm{ALG}^{(t)}$. To keep track of these items, for $j, t \in [T]$, $t \geq 2$, we let $R_j^{(t)}$ be

the items that belong to $\mathrm{ALG}_j^{(t-1)}$ but are not in $\mathrm{ALG}_j^{(t)}$. Formally, let $R_j^{(t)} = \mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_j^{(t)}$ for $j \in [t-1], t \geq 2$. By letting $R_j^{(t)} = R_{t-1}^{(t)}$ for all $j \geq t$, we can define

$$\mathcal{R}^{(t)} = (R_1^{(t)}, \ldots, R_T^{(t)}).$$

The next lemma shows that $\mathcal{R}^{(t)}$ is a chain for each $t \in [T]$, $t \geq 2$. Its proof can be found in Appendix B.4.3.

Lemma 2.4.5. *For every $t \in [T]$, $t \geq 2$, $\mathcal{R}^{(t)}$ is a chain.*

In the remainder of the section, we are going to extensively compare the following quantities related to the choices made by the algorithm in iteration t:

$$a_t = p_t(\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \quad \text{for } t \in [T],$$

$$b_t = \sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}) \quad \text{for } t \in [T],\ t \geq 2,$$

$$d_t = \sum_{j=t+1}^{T} p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})) \quad \text{for } t \in [T].$$

Main ideas. The key steps of the proof are as follows:

1. We create a family of possibly infeasible "hybrid" chains $\mathrm{HB}^{(0)}, \mathrm{HB}^{(1)}, \ldots, \mathrm{HB}^{(T)}$, interpolating between OPT and ALG. This technique is inspired by certain proofs appearing in the literature on submodular functions, such as [10], where however all intermediate solutions that are created are feasible. We set $\mathrm{HB}^{(0)} = \mathrm{OPT}$ and as t increases we smoothly move from OPT to ALG:

$$\mathrm{HB}_j^{(t)} = \begin{cases} \mathrm{ALG}_j^{(t)} & \text{for } j \in [t] \\ (\mathrm{OPT}_j \setminus \mathrm{OPT}_t) \cup \mathrm{ALG}_t^{(t)} & \text{otherwise} \end{cases} \quad \text{for } t \in [T].$$

Note that, in particular, $\mathrm{HB}^{(T)} = \mathrm{ALG}^{(T)} = \mathrm{ALG}$. Clearly, upper bounding $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$ for all $t \in [T]$ and summing those bounds leads to an upper bound on the difference between $\Phi(\mathrm{OPT})$ and $\Phi(\mathrm{ALG})$. Hence, we would like to bound each of $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$ against a function of $\Phi(\mathrm{ALG})$, that is, of the profit of the solution output by the algorithm. This is achieved in two steps.

2. First, we compare $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$ with the *improvement made by the algorithm at time j* – that is, the change of profit between $\mathrm{ALG}^{(j)}$ and $\mathrm{ALG}^{(j-1)}$ – for appropriately chosen times $j \leq t$. We distinguish two cases, according to how the improvement made by the algorithm compares with the improvement made by OPT.

   To define the two cases, we first partition $[T]$ into two sets, according to how the increment of the profit of OPT compares to the increment of the profits of $\mathrm{ALG}^{(t)}$ and $\mathcal{R}^{(t)}$. Formally, for $t \in [T]$, define

   $$\delta_t = a_t - cb_t,$$

   where we let $b_1 = 0$. Let $\mathcal{B} = \{t_1, t_2, \ldots, t_k\}$ denote all the times where for $t_i \in \mathcal{B}$,

   $$p_{t_i}(\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1}) > (1+\epsilon)\delta_{t_i}.$$

   Case 1: $t \in [T] \setminus \mathcal{B}$. Intuitively, when $t \in [T] \setminus \mathcal{B}$, the increment of profit of OPT is not much larger than the increment of profit of $\mathrm{ALG}^{(t)}$. In this case, we can bound $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$ as a function of the operations performed by the algorithm at time t only. In particular, we show the following (see Appendix B.4.5 and Appendix B.4.6 for proofs):

   Lemma 2.4.6 (Step-by-step loss, part 1)**.** $\Phi(\mathrm{HB}^{(0)}) - \Phi(\mathrm{HB}^{(1)}) \leq \epsilon a_1 + d_1$.

   Lemma 2.4.7 (Step-by-step loss, part 2)**.** $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)}) \leq \epsilon a_t - (c-1)b_t + d_t$ *for* $t \in [T] \setminus \mathcal{B}$, $t \geq 2$.

   Case 2: $t \in \mathcal{B}$. When $t \in \mathcal{B}$, to bound $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$ we also need to take into

account decisions made by the algorithm in certain previous steps. This is achieved in the following lemma, where we let $t_0 = 1$ (see Appendix B.4.7 for a proof):

Lemma 2.4.8 (Step-by-step loss, part 3)**.** *For $t_i \in \mathcal{B}$, let $t' = t_{i-1}$. Then:*

$$\Phi(\mathrm{HB}^{(t_i-1)}) - \Phi(\mathrm{HB}^{(t_i)}) \leq \epsilon a_{t_i} - (c-1)b_{t_i} + d_{t_i} + (1+\epsilon)c\sum_{j=t'}^{t_i-1} a_j + c\epsilon\,\Phi(\mathrm{ALG}^{(t_i)}).$$

3. Notice that in Lemma 2.4.7 and Lemma 2.4.8, the bound on $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$ depends also on $\mathcal{R}^{(t)}$ via the term b_t. Keeping in mind our final goal of bounding the improvement made by the algorithm at time t against $\Phi(\mathrm{ALG})$ only, the following lemma eliminates the dependence on b_t. The proof is provided in Appendix B.4.8.

   Lemma 2.4.9 (Removal Lemma)**.** $\sum_{t=2}^{T} b_t \leq \frac{1}{c-1}\Phi(\mathrm{ALG})$.

4. Finally, we put these bounds together to achieve a bound for $\sum_{t=1}^{T} \Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$, which leads to a constant factor bound for the ratio between the profit of the optimal solution and the profit of the solution output by the algorithm. See Theorem 2.4.1. We present the proof in Section 2.4.3.

Running time. Immediately from the running time of the FPTAS for knapsack from [44], that takes time $\tilde{O}(n + (\frac{1}{\epsilon})^{9/4})$.

2.4.3 Proof of Theorem 2.4.1

We can now put things together and prove Theorem 2.4.1. By summing over all $t \in [T]$ the bounds from Lemmas 2.4.7 and 2.4.8 and employing Lemma 2.4.6, in Claim 2.4.10 below we give an initial bound on $\sum_{t=1}^{T}(\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)}))$. Its proof can be found in Appendix B.4.9.

Claim 2.4.10.

$$\sum_{t=1}^{T}(\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})) \leq (c+1+(c+1)\epsilon)\Phi(\mathrm{ALG}) + (2+(c+1)\epsilon)\sum_{t=2}^{T} b_t + c\epsilon\sum_{t=1}^{T}\Phi(\mathrm{ALG}^{(t)}).$$

Let U be the right-hand side of the inequality from Claim 2.4.10. We bound U term-by-term. To bound $c\epsilon \sum_{t=1}^{T} \Phi(\text{ALG}^{(t)})$, note that $\Phi(\text{ALG}^{(t)}) \leq \Phi(\text{ALG})$ for all $t \in [T]$. Thus,

$$\sum_{t=1}^{T} \Phi(\text{ALG}^{(t)}) \leq T\ \Phi(\text{ALG}). \tag{2.2}$$

Finally, using (2.2) and Lemma 2.4.9, we give the following upper bound to the right hand side of the inequality in Claim 2.4.10. Its proof can be found in Appendix B.4.10.

Claim 2.4.11.

$$U \leq (c + 1 + \frac{2}{c-1} + \epsilon')\ \Phi(\text{ALG}).$$

Combining Claims 2.4.10 and 2.4.11, we obtain

$$\sum_{t=1}^{T} (\Phi(\text{HB}^{(t-1)}) - \Phi(\text{HB}^{(t)})) \leq (c + 1 + \frac{2}{c-1} + \epsilon')\ \Phi(\text{ALG}).$$

The left hand side of the inequality is a telescoping sum and reduces to $\Phi(\text{HB}^{(0)}) - \Phi(\text{HB}^{(T)}) = \Phi(\text{OPT}) - \Phi(\text{ALG})$. Thus,

$$\Phi(\text{OPT}) \leq (c + 2 + \frac{2}{c-1} + \epsilon')\ \Phi(\text{ALG}) \quad \Rightarrow \quad (\frac{c-1}{c^2+c} - \epsilon')\ \Phi(\text{OPT}) \leq \Phi(\text{ALG}),$$

concluding the proof.

2.5 Experimental results

In this section, we give some computational results that compare the performance of the c-flexible algorithm presented in Section 2.3 with the integer programming solver Gurobi. In Section 2.5.1, we discuss each of the algorithms tested. In Section 2.5.2, we give detail on the generation of the instances, and on the settings of the experiments. Finally, in Section 2.5.3, we compare the results of the algorithms and discuss their implications.

2.5.1 Algorithms tested

Integer programming solver. For the integer programming solver, we give Gurobi the standard integer programming formulation for the generalized incremental knapsack problem given in (GIK-IP) in Section 1.2.1.

***c*-flexible algorithms.** For the c-flexible algorithms, we implement the algorithm given in Section 2.3, for $c = 1$ (fully flexible) and $c = 2$.

2.5.2 Instance generation and experimental setup

For randomly generated instances of the generalized incremental knapsack problem of different sizes, we test each of the algorithms described in Section 2.5.1. The results are reported in Tables 2.1 and 2.2. For instances of different sizes, Gurobi was given different optimality tolerance thresholds, ranging from 5% to 1%. In Table 2.1, for each of the randomly generated instances, each item has a random weight and random profits that are correlated with the weight of the item; whereas in Table 2.2, each item has random and uncorrelated weight and profits.

We now give more details on the generation of random instances. For Table 2.1, the capacity at time 1 is a random integer between 1 and 50. For each subsequent time, the capacity increases by a random integer between 1 and 50. For each item i, weight w_i is a random integer between 1 and $\frac{10 \cdot W_T}{n}$. Each $p_{i,1}$ is correlated with the weight w_i. In particular, it is a random integer between w_i and $1.2 \cdot w_i$. For $t \in [T]$, $t \geq 2$, profit $p_{i,t} = p_{i,t-1} \cdot (\frac{T-t}{T-t+1} + r \cdot \frac{1}{T-t+1})$, where r is a randomly selected element from the set $\{-1, -0.9, \ldots, 0, 0.1, \ldots, 1\}$. For Table 2.2, the capacity is generated identically to Table 2.1. For each item i and for each time t, weight w_i and profits $p_{i,t}$ are all random integers between 1 and $\frac{10 \cdot W_T}{n}$. Experiments were performed on a Dell XPS 15.

For each value of n and T, we report an average over 10 instances. For all problem sizes except $n = T = 3000$, for each of the algorithm tested, the "Mean difference" columns report the mean difference with respect to the solution that Gurobi outputs. For $n = T = 3000$, Gurobi runs out of memory and are not able to return feasible solutions. Hence, the "Mean difference" column for the

2-flexible algorithm reports the mean difference with respect to the fully-flexible algorithm.

2.5.3 Results and discussion

Problem Size		Gurobi		Fully flexible ($c = 1$)		2-flexible	
n	T	Optimality Gap	Time (Sec)	Mean difference	Time (Sec)	Mean difference	Time (Sec)
50	50	1%	175	2.9%	0.1	12.1%	0.1
100	100	5%	4.3	1.1%	0.8	9.1%	0.8
		3%	210	2.0%		9.9%	
		2%	1534	2.1%		10.0%	
500	500	5%	98	0.0%	6.8	7.8%	8.4
		3%	100	0.2%		7.9%	
		2%	125	0.8%		8.5%	
3000	3000	100%	-	-	191	7.0%	262

Table 2.1: Correlated weights and profits, Gurobi and c-flexible algorithms

Problem Size		Gurobi		Fully flexible ($c = 1$)		2-flexible	
n	T	Optimality Gap	Time (Sec)	Mean difference	Time (Sec)	Mean difference	Time (Sec)
50	50	1%	0.6	6.9%	0.1	3.0%	0.1
100	100	1%	2.4	5.9%	0.3	2.5%	0.3
500	500	1%	25.3	2.9%	7.7	1.4%	6.8
3000	3000	100%	-	-	333	−0.3%	286

Table 2.2: Random and uncorrelated weights and profits, Gurobi and c-flexible algorithms

Unsurprisingly, compared to problems with random and uncorrelated weights and profits, problems with correlated weights and profits take longer to solve across all algorithms implemented. However, these problems drastically increase the running time for Gurobi but only slightly increase the running time for our c-flexible algorithms.

More generally, while for small values of n, T Gurobi converges given a not-too-small gap tolerance, for $n = T = 3000$, it is not even able to find a feasible solution. In contrast, the c-flexible algorithms converge fast even in instances with large n and T, and very fast on instances with smaller values of n and T, outputting solutions of good quality (the best of the solutions output by the algorithms with $c = 1$ and $c = 2$ has profit that is never worse than 3% the solution output

by Gurobi). We also find that when profits and weights are uncorrelated, the c-flexible algorithm perform better when $c = 2$ than when $c = 1$, both in terms of the profit of the solution it outputs as well as in running time.

Chapter 3: Algorithms for the generalized incremental knapsack problem through a sequencing reformulation

3.1 Introduction

In this chapter, we show additional approximability results for the generalized incremental knapsack problem. First, in Section 3.2, we give a polynomial time $(\frac{1}{2} - \epsilon)$-approximation, that improves upon the theoretical results presented in Chapter 2. In Section 3.3, using ideas developed for the $(\frac{1}{2} - \epsilon)$-approximated algorithm, we first give a $(1 - \epsilon)$-approximation with running time exponentially dependent on $\log(n \cdot \frac{w_{\max}}{w_{\min}})$. Thus, this result only gives a QPTAS when the ratio $\frac{w_{\max}}{w_{\min}}$ is polynomial in the problem's input size. With that being said, in Section 3.4, we develop a true QPTAS using a dynamic programming approach, which employs the algorithm from Section 3.3 as a subroutine.

3.2 A polynomial-Time $(\frac{1}{2} - \epsilon)$-approximation

In this section, we show that the generalized incremental problem can be approximated within a factor arbitrarily close to $\frac{1}{2}$. The specifics of this finding, given in Theorem 1.3.2, is reprinted here for convenience.

Theorem 1.3.2. *For any error parameter $\epsilon \in (0, \frac{1}{2})$, the generalized incremental knapsack problem can be approximated within factor $\frac{1}{2} - \epsilon$. The running time of our algorithm is $O(n^{O(1/\epsilon^2)} \cdot |\mathcal{I}|^{O(1)})$, where $|\mathcal{I}|$ stands for the input size.*

Outline. We start off Section 3.2.1 by proposing an equivalent formulation of the generalized incremental knapsack problem as a single-machine sequencing problem. Given this reformulation, we explain in Section 3.2.2 how the profit function can be decomposed into "heavy" and

"light" item contributions. Somewhat informally, with respect to an unknown optimal sequencing solution, the marginal contribution of each item to the overall profit will be classified as being either heavy or light, depending on the item's weight and position on the timeline. Guided by this decomposition, our approach consists of devising two approximation schemes, one competing against the best-possible profit due to heavy contributions (Section 3.2.3) and the other against the analogous quantity due to light contributions (Section 3.2.4). The best of these algorithms will be shown to provide an approximation guarantee of $\frac{1}{2} - \epsilon$, thereby deriving Theorem 1.3.2. It is worth pointing out that the techniques involved in competing against light contributions will be further utilized in Sections 3.3 and 3.4 to obtain an approximation scheme for general instances, albeit in quasi-polynomial time.

3.2.1 An equivalent sequencing formulation

In what follows, we present an equivalent sequencing reformulation for the generalized incremental knapsack problem. As explained in subsequent sections, the interchangeability between these formulations allows us to describe our algorithmic ideas and to analyze their performance guarantees with greater ease. For this purpose, we proceed by arguing that the generalized incremental knapsack problem can be rephrased as a sequencing problem on a single machine as follows:

- Let $\pi : [n] \rightarrow [n]$ be a permutation of the underlying items, where $\pi(i)$ stands for the position of item i.
- By viewing the weight of each item as its processing time, we define the completion time of item i with respect to π as $C_\pi(i) = \sum_{j \in [n]:\pi(j) \leq \pi(i)} w_j$. Accordingly, the profit $\varphi_\pi(i)$ of this item is given by the largest profit we can gain by inserting i at a time period whose capacity occurs is at least $C_\pi(i)$, namely, $\varphi_\pi(i) = \max\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq C_\pi(i)\}$, with the convention that $W_{T+1} = \infty$ and $p_{i,T+1} = 0$ for every item i.
- The overall profit of the permutation π is specified by $\Psi(\pi) = \sum_{i \in [n]} \varphi_\pi(i)$. Our objective is

to compute a permutation whose profit is maximized.

The next lemma captures the equivalence between the item-introducing perspective of the generalized incremental knapsack problem and the sequencing perspective described above.

Lemma 3.2.1. *Any feasible chain $\mathcal{S}$ can be mapped to a permutation $\pi_{\mathcal{S}}$ with $\Psi(\pi_{\mathcal{S}}) \geq \Phi(\mathcal{S})$. Conversely, any permutation π of a subset of the items can be mapped to a feasible chain $\mathcal{S}_\pi$ with $\Phi(\mathcal{S}_\pi) = \Psi(\pi)$.*

Proof. First, given a feasible chain $\mathcal{S}$, we construct the permutation $\pi_{\mathcal{S}}$ as follows:

- For each $t \in [T]$, let π^t be an arbitrary permutation of the items introduced in this period, $S_t \setminus S_{t-1}$. In addition, let π^{T+1} be an arbitrary permutation of the remaining items, i.e., those in $[n] \setminus S_T$.
- The permutation $\pi_{\mathcal{S}}$ is defined as the concatenation of $\pi^1, \ldots, \pi^{T+1}$ in this order. Namely, for $i \in S_t \setminus S_{t-1}$ with $t \in [T]$, we have $\pi_{\mathcal{S}}(i) = \pi^t(i) + |S_{t-1}|$, whereas for $i \in [n] \setminus S_T$, we have $\pi_{\mathcal{S}}(i) = \pi^{T+1}(i) + |S_T|$.

To prove that $\Psi(\pi_{\mathcal{S}}) \geq \Phi(\mathcal{S})$, it suffices to argue that $\varphi_{\pi_{\mathcal{S}}}(i) \geq p_{i,t_i}$ for every item $i \in S_T$, where t_i stands for the insertion time of item i with respect to the chain $\mathcal{S}$. To derive this relation, note that $C_{\pi_{\mathcal{S}}}(i) \leq w(S_{t_i}) \leq W_{t_i}$ for any such item, where the last inequality follows from the feasibility of $\mathcal{S}$. Therefore, $\varphi_{\pi_{\mathcal{S}}}(i) = \max\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq C_{\pi_{\mathcal{S}}}(i)\} \geq p_{i,t_i}$.

Conversely, given a permutation π of any subset of items, we construct a chain $\mathcal{S}_\pi$ that includes all items whose completion time is at most W_T. Specifically, the insertion time t_i of each such item i will be the time period that maximizes p_{i,t_i} over the set $\{t \in [T] : W_t \geq C_\pi(i)\}$. As such, the chain $\mathcal{S}_\pi$ is indeed feasible, since $w(S_t) \leq \sum_{i \in [n]: C_\pi(i) \leq W_t} w_i \leq W_t$ for every $t \in [T]$. To show that $\Phi(\mathcal{S}_\pi) = \Psi(\pi)$, it remains to explain why $p_{i,t_i} = \varphi_\pi(i)$ for inserted items and why $\varphi_\pi(i) = 0$ for non-inserted ones. To this end, note that our choice for the insertion time t_i follows the definition of $\varphi_\pi(i)$ to the letter, meaning that $p_{i,t_i} = \varphi_\pi(i)$. On the other hand, for any item i we do not insert to $\mathcal{S}_\pi$, one has $\varphi_\pi(i) = 0$, since $C_\pi(i) > W_T$. □

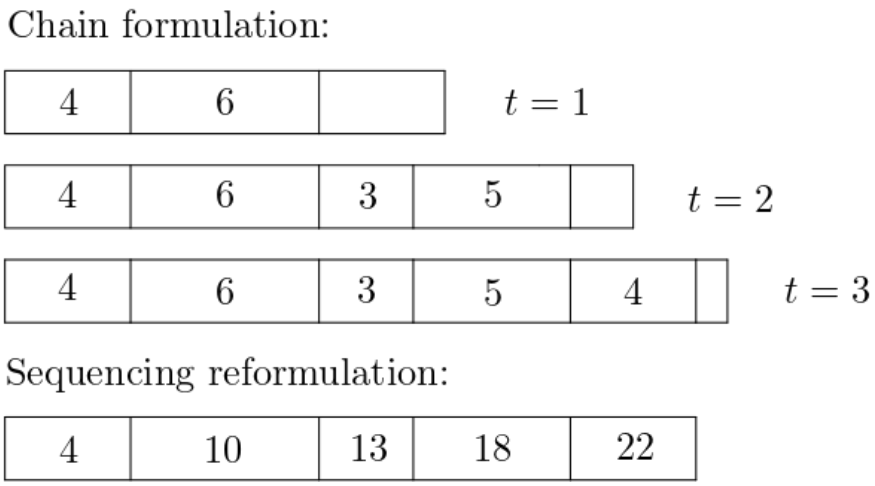

Figure 3.1: In this example, we give a chain representation of 5 items and 3 times and an equivalent sequencing reformulation. In the chain representation, the number within each bar represents the weight of the item. An unlabeled bar represents any unused capacity for that time. The sequencing reformulation gives an equivalent permutation for the chain representation, where the number in each block indicates item i's completion time, given by $C_\pi(i) = \sum_{j \in [n]: \pi(j) \leq \pi(i)} w_j$.

In Figure 3.1, we give a pictorial example of this sequencing reformulation. It is worth noting that, eventually, in Lemma 5.3.2, we will generalize the ideas of Lemma 3.2.1 to give a reformulation of (multi-bin) incremental packing problems as (multi-machine) sequencing problems.

3.2.2 Profit decomposition and high-level overview

In what follows, we focus our attention on the sequencing formulation and present a decomposition of the profit function Ψ into "heavy" and "light" contributions, collected over geometrically-increasing intervals. With the necessary definitions in place, we outline how a decomposition of this nature guides us in proposing two approximation schemes, to separately compete against heavy and light contributions. The main result of this section, as stated in Theorem 1.3.2, will eventually be derived by taking the more profitable of these approaches.

For simplicity of presentation, we assume without loss of generality that $\epsilon \in (0, \frac{1}{2})$, and moreover, that $\frac{1}{\epsilon}$ is an integer. In addition, we assume that $w_{\min} = \min_{i \in [n]} w_i = 3$; the latter property can easily be enforced through scaling all item weights w_i and time period capacities W_t by a factor of $\frac{3}{w_{\min}}$.

Profit decomposition. We begin by geometrically partitioning the interval $[0, \sum_{i\in[n]} w_i]$ by powers of $1+\epsilon$ into a collection of intervals $\mathcal{I}_0, \ldots, \mathcal{I}_K$, where $K = \lceil \log_{1+\epsilon}(\sum_{i\in[n]} w_i) \rceil$. Specifically, $\mathcal{I}_0 = [0, 1]$ and $\mathcal{I}_k = ((1+\epsilon)^{k-1}, (1+\epsilon)^k]$ for $k \in [K]$. With this definition, the profit $\Psi(\pi) = \sum_{i\in[n]} \varphi_\pi(i)$ of any permutation π can be expressed by summing item contributions according to the interval in which their completion times fall, i.e.,

$$\Psi(\pi) = \sum_{k\in[K]_0} \sum_{\substack{i\in[n]:\\ C_\pi(i)\in\mathcal{I}_k}} \varphi_\pi(i) \ .$$

We say that item i is k-heavy when $w_i \geq \epsilon^2 \cdot (1+\epsilon)^k$; otherwise, this item is k-light. We denote the sets of k-heavy and k-light items by H_k and L_k, respectively, noting that $H_0 \supseteq H_1 \supseteq \cdots \supseteq H_K$ and that $L_k = [n] \setminus H_k$ for every k. As a side note, one can easily verify that all items are 0-heavy (i.e., $H_0 = [n]$), by recalling that $w_{\min} = 3$ and $\epsilon < \frac{1}{2}$. Consequently, the profit $\Psi(\pi)$ can be refined by separating k-heavy and k-light items, namely,

$$\Psi(\pi) = \underbrace{\sum_{k\in[K]_0} \sum_{\substack{i\in H_k:\\ C_\pi(i)\in\mathcal{I}_k}} \varphi_\pi(i)}_{\Psi_{\text{heavy}}(\pi)} + \underbrace{\sum_{k\in[K]_0} \sum_{\substack{i\in L_k:\\ C_\pi(i)\in\mathcal{I}_k}} \varphi_\pi(i)}_{\Psi_{\text{light}}(\pi)} \ . \tag{3.1}$$

As shown above, we designate the first and second terms in the above expression by $\Psi_{\text{heavy}}(\pi)$ and $\Psi_{\text{light}}(\pi)$, respectively.

Overview. Let π^* be an optimal permutation, with $\Psi(\pi^*) = \Psi_{\text{heavy}}(\pi^*) + \Psi_{\text{light}}(\pi^*)$. The remainder of this section is dedicated to presenting two approximation schemes that would separately compete against $\Psi_{\text{heavy}}(\pi^*)$ and $\Psi_{\text{light}}(\pi^*)$:

- *Heavy contributions*: Section 3.2.3 explains how dynamic programming ideas allow us to efficiently compute a permutation $\pi_{\text{heavy}} : [n] \to [n]$ satisfying $\Psi(\pi_{\text{heavy}}) \geq (1-\epsilon) \cdot \Psi_{\text{heavy}}(\pi^*)$. The resulting running time will be $O(n^{O(1/\epsilon^2)} \cdot |\mathcal{I}|)$.

- *Light contributions*: Section 3.2.4 argues that the generalized assignment algorithm of

Shmoys and Tardos [60] can be leveraged to compute a permutation $\pi_{\text{light}} : [n] \to [n]$ satisfying $\Psi(\pi_{\text{light}}) \geq (1-\epsilon) \cdot \Psi_{\text{light}}(\pi^*)$. This algorithm can be implemented in $O((\frac{|\mathcal{I}|}{\epsilon})^{O(1)})$ time.

Consequently, to establish the approximation guarantee stated in Theorem 1.3.2, we pick the more profitable permutation out of π_{heavy} and π_{light}, to obtain a profit of

$$\begin{aligned}
\max\left\{\Psi\left(\pi_{\text{heavy}}\right), \Psi\left(\pi_{\text{light}}\right)\right\} &\geq \frac{1}{2} \cdot \left(\Psi\left(\pi_{\text{heavy}}\right) + \Psi\left(\pi_{\text{light}}\right)\right) \\
&\geq \frac{1-\epsilon}{2} \cdot \left(\Psi_{\text{heavy}}\left(\pi^*\right) + \Psi_{\text{light}}\left(\pi^*\right)\right) \\
&= \frac{1-\epsilon}{2} \cdot \Psi\left(\pi^*\right) \ .
\end{aligned}$$

3.2.3 Algorithm for heavy contributions

In what follows, we present a dynamic programming approach for computing a permutation that competes against $\Psi_{\text{heavy}}(\pi^*)$, as formally stated in the next theorem.

Theorem 3.2.2. *For any error parameter $\epsilon \in (0,1)$, there is an $O(n^{O(1/\epsilon^2)} \cdot |\mathcal{I}|)$-time algorithm for constructing a permutation π_{heavy} with a profit of $\Psi(\pi_{\text{heavy}}) \geq (1-\epsilon) \cdot \Psi_{\text{heavy}}(\pi^*)$.*

3.2.3.1 Preliminaries

The intuition behind our algorithm begins with the observation that, in order to compete against $\Psi_{\text{heavy}}(\pi^*)$, we can safely eliminate items that are classified as light with respect to the interval in which their completion time falls. While the remaining items will be shifted back in the residual permutation, potentially being completed in a lower-index interval, each of them will still be heavy. To formalize these notions, for a subset of items $S \subseteq [n]$ and a permutation $\pi : S \to [|S|]$, we say that the pair (S, π) is bulky if, for every $k \in [K]_0$, all items with a completion time in $\mathcal{I}_k$ are k-heavy, i.e., $\{i \in S : C_\pi(i) \in \mathcal{I}_k\} \subseteq H_k$. The next claim shows that bulky pairs can attain a total profit of at least $\Psi_{\text{heavy}}(\pi^*)$.

Lemma 3.2.3. *There exist a subset of items $S \subseteq [n]$ and a permutation $\pi : S \to [|S|]$ such that (S, π) is bulky and $\sum_{i \in S} \varphi_\pi(i) \geq \Psi_{\text{heavy}}(\pi^*)$.*

Proof. With respect to the optimal permutation π^*, we define a new permutation π by eliminating, for every $k \in [K]_0$, all items $i \in L_k$ with $C_{\pi^*}(i) \in \mathcal{I}_k$. The subset S will consist of the remaining items. To see why (S, π) is bulky, note that $C_\pi(i) \leq C_{\pi^*}(i)$ for any $i \in S$, meaning that each such item is still heavy with respect to the interval that contains $C_\pi(i)$, since $H_0 \supseteq \cdots \supseteq H_K$. In terms of profit, the latter observation implies that, for every item $i \in S$,

$$\begin{aligned} \varphi_\pi(i) &= \max\left\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq C_\pi(i)\right\} \\ &\geq \max\left\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq C_{\pi^*}(i)\right\} \\ &= \varphi_{\pi^*}(i) \ . \end{aligned}$$

Summing the above inequality over all items in S, we have $\sum_{i \in S} \varphi_\pi(i) \geq \sum_{i \in S} \varphi_{\pi^*}(i) = \Psi_{\text{heavy}}(\pi^*)$, where the latter equality holds since every eliminated item does not contribute toward $\Psi_{\text{heavy}}(\pi^*)$ but rather toward $\Psi_{\text{light}}(\pi^*)$. □

Additional notation. For a bulky pair (S, π), we define its top index as $\text{top}(S, \pi) = \max\{k \in [K]_0 : \{C_\pi(i) : i \in S\} \cap \mathcal{I}_k \neq \emptyset\}$, that is, the largest index of an interval that contains at least one completion time. In addition, we define $\text{core}(S)$ as the set of $\min\{\frac{1}{\epsilon^2}, |S|\}$ heaviest items in S, breaking ties by adding to $\text{core}(S)$ small-index items before large-index ones. Finally, the makespan of (S, π) corresponds to the maximum completion time of an item in S with respect to the permutation π; in our case, this measure identifies with $w(S)$.

3.2.3.2 The continuous dynamic program

The technical crux in restricting attention to bulky pairs will be exhibited through our dynamic programming formulation. As formally explained below, by focusing on the dual objective of makespan minimization, we prove the existence of a well-hidden optimal substructure within the

sequencing problem.

States. Each state (k, ψ_k, Q_k) of our dynamic program consists of the following parameters, whose precise role will be clarified once their corresponding value function is presented:

- The index of the current interval k, taking values in $[K]_0$.
- The total profit ψ_k collected thus far, due to items whose completion time falls in $\mathcal{I}_0, \ldots \mathcal{I}_k$. For the time being, ψ_k will be treated as a continuous parameter, taking values in $[0, \sum_{i \in [n]} \max_{t \in [T]} p_{i,t}]$.
- The core Q_k of the set of items whose completion time falls in $\mathcal{I}_0, \ldots \mathcal{I}_k$. By definition of core$(\cdot)$, this parameter is restricted to item sets of cardinality at most $\frac{1}{\epsilon^2}$.

It is important to emphasize that, since ψ_k is a continuous parameter, the dynamic programming formulation below is still not algorithmic in nature, and should be viewed as a characterization of optimal solutions. We remark that when the profits $p_{i,t}$ are all integers, we can restrict ψ_k to integer values in $[0, \sum_{i \in [n]} \max_{t \in [T]} p_{i,t}]$, and our dynamic program can be solved in pseudo-polynomial time. In either case, we explain in Section 3.2.3.3 how to discretize the parameter ψ_k to take polynomially-many values while incurring only an ϵ-loss in profit.

Value function. The value function $F(k, \psi_k, Q_k)$ represents the minimum makespan $w(S)$ that can be attained over all bulky pairs (S, π) that satisfy the following conditions:

1. *Top index*: $\text{top}(S, \pi) \leq k$.
2. *Total profit*: $\Psi(\pi) \geq \psi_k$.
3. *Core*: $\text{core}(S) = Q_k$.

For ease of presentation, we denote the collection of bulky pairs that meet conditions 1-3 by $\text{Bulky}(k, \psi_k, Q_k)$. When the latter set is empty, we define $F(k, \psi_k, Q_k) = \infty$. With these definitions, Lemma 3.2.3 proves in retrospect the existence of a bulky pair $(S, \pi) \in$

$\text{Bulky}(K, \Psi_{\text{heavy}}(\pi^*), \text{core}(S))$ with $F(K, \Psi_{\text{heavy}}(\pi^*), \text{core}(S)) < \infty$. Therefore, had we been able to compute the maximal value ψ^* that satisfies $F(K, \psi^*, Q_K) < \infty$ over all possible cores Q_K, its corresponding bulky pair would have guaranteed a profit of at least $\psi^* \geq \Psi_{\text{heavy}}(\pi^*)$.

Optimal substructure. To this end, we proceed by unveiling the optimal substructure that allows us to compute the value function F by means of dynamic programming. In order to gain intuition, suppose that (S, π) is a bulky pair that attains $F(k, \psi_k, Q_k)$. Then, we argue that, by eliminating from S the set of items Q whose completion time falls within the interval $\mathcal{I}_k$, one obtains a bulky pair that attains $F(k-1, \psi_{k-1}, Q_{k-1})$, where the residual profit ψ_{k-1} is obtained by removing from ψ_k the contribution of items in Q and Q_{k-1} is an appropriately chosen core. In this regard, the obvious question is: To attain $F(k-1, \psi_{k-1}, Q_{k-1})$, why would our dynamic program not pick any of the items in Q? The crux of our argument would be that, since the intervals $\mathcal{I}_0, \ldots, \mathcal{I}_k$ are geometrically increasing in length, the k-heaviness of all items in Q forces each such item to reside within the core Q_k, meaning that we will indeed prevent it from being picked when $F(k-1, \psi_{k-1}, Q_{k-1})$ is computed by a suitable choice of the core Q_{k-1} that, in particular, will be disjoint from Q.

Formally, suppose that $\text{Bulky}(k, \psi_k, Q_k) \neq \emptyset$, and let (S, π) be a bulky pair that minimizes $w(S)$ over this set. Let $Q = \{i \in S : C_\pi(i) \in \mathcal{I}_k\}$ be the set of items in S whose completion time with respect to π falls in the interval $\mathcal{I}_k$. Note that since $\text{top}(S, \pi) \leq k$, completion times cannot fall in $\mathcal{I}_{k+1}, \ldots, \mathcal{I}_K$. We first argue that $|Q| \leq \frac{1}{\epsilon}$. To verify this claim, note that since (S, π) is bulky, $Q \subseteq H_k$. As a result, every item in Q has a weight of at least $\epsilon^2 \cdot (1+\epsilon)^k$, while $\mathcal{I}_k = ((1+\epsilon)^{k-1}, (1+\epsilon)^k]$, meaning that we necessarily have $|Q| \leq \frac{(1+\epsilon)^k - (1+\epsilon)^{k-1}}{\epsilon^2 \cdot (1+\epsilon)^k} \leq \frac{1}{\epsilon}$.

Now, let us define the pair $(\hat{S}, \hat{\pi})$, where $\hat{S} = S \setminus Q$ and $\hat{\pi} : \hat{S} \to [|\hat{S}|]$ is the permutation where items in $\hat{S}$ follow their relative order in π, that is, for any pair of items i_1 and i_2, we have $\hat{\pi}(i_1) < \hat{\pi}(i_2)$ if and only if $\pi(i_1) < \pi(i_2)$. In addition, let $\psi_{k-1} = [\psi_k - \sum_{i \in Q} \varphi_\pi(i)]^+$ and $Q_{k-1} = \text{core}(\hat{S})$, where $[x]^+ = \max\{x, 0\}$. These definitions directly ensure that $(\hat{S}, \hat{\pi}) \in \text{Bulky}(k-1, \psi_{k-1}, Q_{k-1})$. Moreover, as we show in Lemma 3.2.4 below, $(\hat{S}, \hat{\pi})$ forms an optimal solution with respect to

the latter state. Intuitively, the key idea for proving this claim shows that, had there been a bulky pair $(\tilde{S}, \tilde{\pi}) \in \text{Bulky}(k-1, \psi_{k-1}, Q_{k-1})$ with $w(\tilde{S}) < w(\hat{S})$, it can be extended to a bulky pair $(\tilde{S}^+, \tilde{\pi}^+) \in \text{Bulky}(k, \psi_k, Q_k)$ by adding the items in Q following their internal order in π, to obtain $w(\tilde{S}^+) < w(S)$, thereby contradicting the optimality of (S, π).

Lemma 3.2.4. $w(\hat{S}) = F(k-1, \psi_{k-1}, Q_{k-1})$.

Proof. Suppose there exists some bulky pair $(\tilde{S}, \tilde{\pi}) \in \text{Bulky}(k-1, \psi_{k-1}, Q_{k-1})$ with $w(\tilde{S}) < w(\hat{S})$. We first claim that $\tilde{S} \cap Q = \emptyset$. To verify this property, had there been an item $i \in \tilde{S} \cap Q$, its weight would satisfy $w_i \geq \epsilon^2 \cdot (1+\epsilon)^k$, since $Q \subseteq H_k$. On the other hand, since $\text{top}(\tilde{S}, \tilde{\pi}) \leq k-1$, the completion times of all items in $\tilde{S}$ with respect to $\tilde{\pi}$ reside within the union of $\mathcal{I}_0, \ldots, \mathcal{I}_{k-1}$, which is the interval $[0, (1+\epsilon)^{k-1}]$, implying that $w(\tilde{S}) \leq (1+\epsilon)^{k-1}$. Therefore, since $\text{core}(\tilde{S})$ is the set of $\min\{\frac{1}{\epsilon^2}, |\tilde{S}|\}$ heaviest items in $\tilde{S}$, regardless of how ties are broken we must have $i \in \text{core}(\tilde{S})$. We have just arrived at a contradiction: Since $\text{core}(\tilde{S}) = Q_{k-1} = \text{core}(\hat{S}) = \text{core}(S \setminus Q)$, it follows that $i \notin Q$.

Knowing that $\tilde{S} \cap Q = \emptyset$, we can extend the permutation $\tilde{\pi} : \tilde{S} \to [|\tilde{S}|]$ to $\tilde{S}^+ = \tilde{S} \cup Q$ by appending the set of items Q in exactly the same order as they appear in π. Letting $\tilde{\pi}^+ : \tilde{S}^+ \to [|\tilde{S}^+|]$ be the resulting permutation, we next argue that $(\tilde{S}^+, \tilde{\pi}^+)$ is in fact a feasible solution to precisely the same subproblem with respect to which (S, π) is optimal. The proof of this structural result is provided in Appendix C.1.1.

Claim 3.2.5. $(\tilde{S}^+, \tilde{\pi}^+) \in \text{Bulky}(k, \psi_k, Q_k)$.

We have just arrived at a contradiction to the fact that (S, π) minimizes $w(S)$ over the set $\text{Bulky}(k, \psi_k, Q_k)$, by observing that $w(\tilde{S}^+) = w(\tilde{S}) + w(Q) < w(\hat{S}) + w(Q) = w(S)$. □

Recursive equations. In light of this structural characterization, to obtain a recursive equation for $F(k, \psi_k, Q_k)$, it suffices to "guess" the collection of items Q, their internal permutation π_Q, the residual profit requirement ψ_{k-1}, and the resulting core Q_{k-1}. Formally, $F(k, \psi_k, Q_k)$ is given by minimizing $F(k-1, \psi_{k-1}, Q_{k-1}) + w(Q)$ over all choices of Q, π_Q, ψ_{k-1}, and Q_{k-1} that simultaneously satisfy the following conditions:

1. Top index: $F(k-1, \psi_{k-1}, \mathcal{Q}_{k-1}) + w(Q) \leq (1+\epsilon)^k$. This constraint ensures that, with the addition of Q, all items can still be packed within $\mathcal{I}_0, \ldots, \mathcal{I}_k$.

2. Total profit: $\psi_{k-1} \geq [\psi_k - \sum_{i \in Q} \varphi^{\rightsquigarrow}_{\pi_Q}(i)]^+$, where the term $\varphi^{\rightsquigarrow}_{\pi_Q}(i)$ denotes the profit of item i with respect to the permutation π_Q, when its completion time is increased by $F(k-1, \psi_{k-1}, \mathcal{Q}_{k-1})$. This constraint guarantees that, by appending π_Q, we obtain a total profit of at least ψ_k.

3. Core: $\mathcal{Q}_{k-1} \cap Q = \emptyset$, $\text{core}(\mathcal{Q}_{k-1} \cup Q) = \mathcal{Q}_k$, $Q \subseteq H_k$, and $|Q| \leq \frac{1}{\epsilon}$. These constraints ensure a correct core update as a result of adding the item set Q, where the latter set consists of at most $\frac{1}{\epsilon}$ items, each restricted to being k-heavy. To better understand the requirement $\text{core}(\mathcal{Q}_{k-1} \cup Q) = \mathcal{Q}_k$, note that the core resulting from the addition of Q can be computed without a complete knowledge of all previously packed items, as all those outside the current core $\mathcal{Q}_{k-1}$ are irrelevant for this purpose (i.e., too light to be one of the $\frac{1}{\epsilon^2}$ heaviest).

3.2.3.3 Discretization and final algorithm

As previously mentioned, due to the continuity of the profit requirement ψ_k, it remains to propose an appropriate discretization of this parameter, so that we obtain a polynomially-sized state space with only negligible loss in profit.

The discrete program $\tilde{F}$. To this end, we alter the underlying state space of our dynamic program, by restricting the continuous parameter ψ_k to a finite set of values, $\mathcal{D}_\psi = \{d \cdot \frac{\epsilon p_{\max}}{n} : d \in [\frac{n^2}{\epsilon}]_0\}$. Here, $p_{\max}$ is the maximum profit attainable by any single item, i.e., $p_{\max} = \max\{p_{it} : i \in [n], t \in [T], \text{ and } w_i \leq W_t\}$. We make use of $\tilde{F}(k, \psi_k, Q_k)$ to designate the value function F restricted to the resulting set of states, and similarly, $\widetilde{\text{Bulky}}(k, \psi_k, \mathcal{Q}_k)$ will stand for the collection of bulky pairs that meet conditions 1-3. As a side note, beyond the additional restriction on ψ_k, both $\tilde{F}$ and $\widetilde{\text{Bulky}}$ are defined identically to F and Bulky.

Analysis. We remind the reader that, in Section 3.2.3.2, the quantity ψ^* was defined as the maximal value satisfying $F(K, \psi^*, Q_K) < \infty$ over all possible cores Q_K, noting that its corresponding bulky pair guarantees a profit of at least $\psi^* \geq \Psi_{\text{heavy}}(\pi^*)$. In order to establish a parallel claim with respect to the discretized program $\tilde{F}$, we prove in Lemma 3.2.6 a lower bound of $(1-\epsilon) \cdot \psi^*$ on the analogous quantity $\tilde{\psi}$ that satisfies $\tilde{F}(K, \tilde{\psi}, Q_K) < \infty$; the proof is provided in Appendix C.1.2. It follows that our dynamic program computes a bulky pair (S, π) in which the permutation π has a profit of $\Psi(\pi) \geq \tilde{\psi} \geq (1-\epsilon) \cdot \psi^* \geq (1-\epsilon) \cdot \Psi_{\text{heavy}}(\pi^*)$.

Lemma 3.2.6. *There exists a value $\tilde{\psi} \in \mathcal{D}_\psi$ such that $\tilde{\psi} \geq (1-\epsilon) \cdot \psi^*$ and such that $\tilde{F}(K, \tilde{\psi}, Q_K) < \infty$ for some core Q_K.*

Running time. We first observe that the function $\tilde{F}(k, \psi_k, Q_k)$ needs to be evaluated over $O(n^{O(1/\epsilon^2)} \cdot |\mathcal{I}|)$ possible states. Indeed, there are $O(K)$ choices for the interval index k, where by definition, $K = \lceil \log_{1+\epsilon}(\sum_{i \in [n]} w_i) \rceil = O(\frac{|\mathcal{I}|}{\epsilon})$. As for the profit parameter ψ_k, following its restriction to the set $\mathcal{D}_\psi$, we ensure that ψ_k takes only $|\mathcal{D}_\psi| = O(\frac{n^2}{\epsilon})$ values. Finally, since the core $Q_k \subseteq [n]$ consists of at most $\frac{1}{\epsilon^2}$ items, there are only $O(n^{O(1/\epsilon^2)})$ subsets to consider.

Now, evaluating each state requires minimizing the restricted function $\tilde{F}(k-1, \psi_{k-1}, Q_{k-1}) + w(Q)$ over all choices of Q, π_Q, ψ_{k-1}, and Q_{k-1} that simultaneously satisfy conditions 1-3 of the recursive equations (see Section 3.2.3.2). In this context, the number of joint configurations for these parameters is $O(n^{O(1/\epsilon^2)})$. Specifically, the profit parameter ψ_{k-1} and the core Q_{k-1} respectively take $O(\frac{n^2}{\epsilon})$ and $O(n^{O(1/\epsilon^2)})$ values as before. In addition, the number of choices for the augmenting set Q is $O(n^{O(1/\epsilon)})$, due to being comprised of at most $\frac{1}{\epsilon}$ items, and there are only $O((\frac{1}{\epsilon})^{O(1/\epsilon)})$ permutations π_Q of these items. To summarize, we incur an overall running time of $O(n^{O(1/\epsilon^2)} \cdot |\mathcal{I}|)$.

3.2.4 Algorithm for light contributions

In this section, we construct a suitably-defined instance of the maximum generalized assignment problem, intended to compete against $\Psi_{\text{light}}(\pi^*)$. We show that, when applied to this highly-

structured instance, the LP-based algorithm of Shmoys and Tardos [60] can be leveraged for computing a permutation that competes against $\Psi_{\text{light}}(\pi^*)$ along the lines of the next theorem.

Theorem 3.2.7. *For any error parameter $\epsilon \in (0,1)$, there is an $O((\frac{|\mathcal{I}|}{\epsilon})^{O(1)})$-time algorithm for constructing a permutation π_{light} with a profit of $\Psi(\pi_{\text{light}}) \geq (1-13\epsilon) \cdot \Psi_{\text{light}}(\pi^*)$.*

3.2.4.1 Instance construction

Intuition. The general intuition behind our construction resides in viewing the intervals $\mathcal{I}_1, \ldots, \mathcal{I}_{K-1}$ as distinct buckets, to which items should be assigned subject to capacity constraints. Clearly, this perspective lacks the extra flexibility of the sequencing formulation, where items may be crossing between multiple successive intervals. In addition, any item-to-bucket assignment has to be associated with a specific profit a-priori, whereas the sequencing-related profits depend on the exact completion time of each item. As explained in the sequel, our approach bypasses the first obstacle by focusing on light items, for which greedy repacking of rounded solutions will be argued to be near-optimal. In regard to the second obstacle, we will allow seemingly unattainable profits, showing that appropriately scaled fractional solutions can be rounded to attain these profits up to negligible loss in optimality.

The construction. Guided by this intuition, we define an instance of the maximum generalized assignment problem as follows:

- *Buckets*: For every $k \in [K-1]$, we set up a bucket $\mathcal{B}_k$. The capacity of this bucket is $\text{capacity}(\mathcal{B}_k) = (1+\epsilon)^k - (1+\epsilon)^{k-1}$, i.e., precisely the length of the interval $\mathcal{I}_k$. It is worth mentioning that there are no buckets corresponding to the intervals $\mathcal{I}_0$ and $\mathcal{I}_K$.

- *Items*: The set of items is still $[n]$, where each item has a weight of w_i.

- *Allowed assignments and profits*: An item i can be assigned to bucket $\mathcal{B}_k$ only when i is $(k+1)$-light. For such an assignment, our profit is $q_{ik} = \max\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq (1+\epsilon)^k\}$.

The goal is to compute a capacity-feasible assignment whose total profit is maximized.

IP formulation. Moving forward, it is instructive to represent this instance through its standard integer programming formulation:

$$\begin{aligned}
\max \quad & \sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik}x_{ik} & & \\
\text{s.t.} \quad & \sum_{k\in[K-1]:i\in L_{k+1}} x_{ik} \le 1 & & \forall\, i \in [n] \\
& \sum_{i\in L_{k+1}} w_i x_{ik} \le \text{capacity}(\mathcal{B}_k) & & \forall\, k \in [K-1] \\
& x_{ik} \in \{0,1\} & & \forall\, k \in [K-1],\, i \in L_{k+1}
\end{aligned} \tag{GAP-IP}$$

In this formulation, each decision variable x_{ik} indicates whether item i is assigned to bucket $\mathcal{B}_k$. The first constraint guarantees that every item is assigned to at most one bucket, and the second constraint ensures that the total weight of the items assigned to each bucket fits within its capacity. The next lemma shows that any feasible assignment can be efficiently mapped to a permutation for our sequencing formulation that collects at least as much profit; the proof is provided in Appendix C.1.3.

Lemma 3.2.8. *Any feasible solution x to* (GAP-IP) *can be translated in $O(nK)$ time to a permutation $\pi_x : [n] \to [n]$ satisfying $\Psi(\pi_x) \ge \sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik}x_{ik}$.*

LP relaxation and lower bound. The linear relaxation of this integer program, (GAP-LP), is obtained by replacing the integrality constraints $x_{ik} \in \{0,1\}$ with non-negativity constraints, $x_{ik} \ge 0$. To have a better intuition for how the fractional optimum of (GAP-LP) is related to $\Psi_{\text{light}}(\pi^*)$, let $C_k^* = \{i \in L_k : C_{\pi^*}(i) \in \mathcal{I}_k\}$ be the subset of k-light items whose completion time with respect to the optimal permutation π^* falls in $\mathcal{I}_k$. Then, within the proof of Lemma 3.2.9 below, we argue that a $1 - O(\epsilon)$ fraction of each such item can be assigned to bucket $\mathcal{B}_{k-1}$. This claim would follow by observing that capacity$(\mathcal{B}_{k-1})$ nearly matches the length of the interval $\mathcal{I}_k$, in which all items in C_k^* are known to fit, potentially except for one item that crosses into $\mathcal{I}_k$. However, the latter item is

k-light, meaning that its weight is very small in comparison to capacity$(\mathcal{B}_{k-1})$, and scaling down all items in C_k^* by a factor of $1 - O(\epsilon)$ clears sufficient capacity for this item as well. The next claim formally shows that this fractional solution is indeed feasible in (GAP-LP) and earns a profit of nearly $\Psi_{\text{light}}(\pi^*)$.

Lemma 3.2.9. OPT(GAP-LP) $\geq (1 - 5\epsilon) \cdot \Psi_{\text{light}}(\pi^*)$.

Proof. In order to derive the desired bound, we prove that (LP) has a feasible fractional solution x with an objective value of at least $(1 - 5\epsilon) \cdot \Psi_{\text{light}}(\pi^*)$. To this end, recalling that $C_k^* = \{i \in L_k : C_{\pi^*}(i) \in \mathcal{I}_k\}$, we have

$$\Psi_{\text{light}}(\pi^*) = \sum_{k \in [K]_0} \sum_{i \in C_k^*} \varphi_{\pi^*}(i) = \sum_{k=2}^{K} \sum_{i \in C_k^*} \varphi_{\pi^*}(i) \ , \tag{3.2}$$

where the second equality follows by observing that completion times cannot fall in either of the intervals $\mathcal{I}_0$ and $\mathcal{I}_1$, since their union is $[0, 2 + \epsilon]$ whereas $w_{\min} = 3$, by our initial assumption in Section 3.2.2.

We define a fractional solution x to (GAP-LP) by setting $x_{i,k-1} = 1 - 5\epsilon$ for every $2 \leq k \leq K$ and $i \in C_k^*$; all other variables take zero values. To verify the feasibility of this solution, note that we clearly have $\sum_{k \in [K-1]: i \in L_{k+1}} x_{ik} \leq 1$ for every item $i \in [n]$. As for the capacity constraints, for every $2 \leq k \leq K$,

$$\begin{aligned}
\sum_{i \in [n]} w_i x_{i,k-1} &= (1 - 5\epsilon) \cdot \sum_{i \in C_k^*} w_i \\
&\leq (1 - 5\epsilon) \cdot \left((1 + \epsilon)^k - (1 + \epsilon)^{k-1} + \epsilon^2 \cdot (1 + \epsilon)^k \right) \\
&\leq (1 - 5\epsilon) \cdot (1 + 5\epsilon) \cdot \left((1 + \epsilon)^{k-1} - (1 + \epsilon)^{k-2} \right) \\
&\leq \text{capacity}\,(\mathcal{B}_{k-1}) \ .
\end{aligned}$$

Here, the first inequality holds since all items in C_k^* have completion times in $\mathcal{I}_k$, implying that their total weight is upper bounded by the length $(1 + \epsilon)^k - (1 + \epsilon)^{k-1}$ of this interval plus the

maximum weight of any item in C_k^*, which is at most $\epsilon^2 \cdot (1+\epsilon)^k$ due to being k-light. The second inequality can easily be verified to hold for every $\epsilon \in (0,1)$.

Consequently, the fractional optimum can be lower-bounded by the objective function of x, to obtain

$$\begin{aligned}
\text{OPT(GAP-LP)} &\geq \sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik} x_{ik} \\
&= (1-5\epsilon) \cdot \sum_{k=2}^{K} \sum_{i\in C_k^*} q_{i,k-1} \\
&\geq (1-5\epsilon) \cdot \sum_{k=2}^{K} \sum_{i\in C_k^*} \varphi_{\pi^*}(i) \\
&= (1-5\epsilon) \cdot \Psi_{\text{light}}(\pi^*) \,,
\end{aligned}$$

where the last equality is precisely (3.2). To understand the second inequality, note that for every item $i \in C_k^*$,

$$\begin{aligned}
\varphi_{\pi^*}(i) &= \max\left\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq C_{\pi^*}(i)\right\} \\
&\leq \max\left\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq (1+\epsilon)^{k-1}\right\} \\
&= q_{i,k-1} \,,
\end{aligned}$$

where the above inequality holds since $C_{\pi^*}(i) \geq (1+\epsilon)^{k-1}$. □

3.2.4.2 Employing the Shmoys-Tardos algorithm

The rounding algorithm. We proceed by utilizing the LP-rounding approach of Shmoys and Tardos [60, Sec. 2], which was originally proposed for the minimum generalized assignment problem. Specifically, given an optimal fractional solution to the linear program (GAP-LP), their algorithm computes an integral vector $\hat{x}$ that satisfies the following properties:

1. *Objective value*: $\hat{x}$ has a super-optimal objective value, i.e.,

$$\sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik}\hat{x}_{ik} \geq \text{OPT(GAP-LP)} \ . \tag{3.3}$$

2. *Item assignment*: $\hat{x}$ assigns each item to at most one bucket, namely, $\sum_{k\in[K-1]:i\in L_{k+1}} \hat{x}_{ik} \leq 1$ for every $i \in [n]$.

3. *Fixable capacity*: For every bucket $\mathcal{B}_k$, if its capacity is violated (i.e., $\sum_{i\in L_{k+1}} w_i\hat{x}_{ik} > \text{capacity}(\mathcal{B}_k)$), there exists a single infeasibility item $i_{\text{inf}(k)}$ with $\hat{x}_{i_{\text{inf}(k)},k} = 1$ whose removal restores the feasibility of that bucket, i.e.,

$$\sum_{i\in L_{k+1}} w_i\hat{x}_{ik} - w_{i_{\text{inf}(k)}} \leq \text{capacity}(\mathcal{B}_k) \ . \tag{3.4}$$

Restoring feasibility with negligible profit loss. Given the above-mentioned properties, a feasible integral solution can obviously be obtained by eliminating the infeasibility item of each bucket with violated capacity. However, this straightforward approach may decrease the objective value by a non-ϵ-bounded factor. Instead, the final step of our algorithm greedily defines an integral solution $\hat{x}^- \leq \hat{x}$ which is feasible for (GAP-IP) and has an objective value of at least $(1-8\epsilon)\cdot\text{OPT(GAP-LP)}$. To this end, for every bucket $\mathcal{B}_k$ whose capacity is not violated by $\hat{x}$, we simply have $\hat{x}^-_{ik} = \hat{x}_{ik}$ for all $i \in L_{k+1}$. In contrast, for every bucket $\mathcal{B}_k$ whose capacity is violated, we proceed as follows:

- Let $i_1, \ldots, i_M$ be an indexing of the set $\{i \in L_{k+1} : \hat{x}_{ik} = 1\}$ such that $\frac{q_{i_1,k}}{w_{i_1}} \geq \cdots \geq \frac{q_{i_M,k}}{w_{i_M}}$.
- Let μ be the maximal index for which $\sum_{m\in[\mu]} w_{i_m} \leq \text{capacity}(\mathcal{B}_k)$.
- Then, our solution sets $\hat{x}^-_{i_1,k} = \cdots = \hat{x}^-_{i_\mu,k} = 1$ and $\hat{x}^-_{ik} = 0$ for any other item. Clearly, $\hat{x}^-_{ik} \leq \hat{x}_{ik}$ for all $i \in L_{k+1}$.

In Lemma 3.2.10, we show that the profit collected by $\hat{x}^-$ nearly matches the fractional optimum. To gain some intuition for the proof of this claim, the main idea is that the capacity of

each bucket $\mathcal{B}_k$ will be shown to be violated in $\hat{x}$ only by an ϵ-related fraction, if at all, due to being assigned only $(k+1)$-light items. For the same reason, our greedy procedure will be shown to nearly exhaust the entire capacity of each violated bucket. In this case, packing items by their profit-to-weight ratio guarantees that we are very close to matching the original profit contribution of such buckets.

Lemma 3.2.10. $\sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik}\hat{x}^{-}_{ik} \geq (1-8\epsilon)\cdot \mathrm{OPT}(\text{GAP-LP})$.

Proof. Recall that the super-optimality property of $\hat{x}$, as stated in (3.3), corresponds to having $\sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik}\hat{x}_{ik} \geq \mathrm{OPT}(\text{GAP-LP})$. Therefore, by changing the order of summation, we can establish the desired claim by proving that $\sum_{i\in L_{k+1}} q_{ik}\hat{x}^{-}_{ik} \geq (1-8\epsilon)\cdot \sum_{i\in L_{k+1}} q_{ik}\hat{x}_{ik}$ for every $k \in [K-1]$. Moreover, since one has $\hat{x}^{-}_{\cdot k} = \hat{x}_{\cdot k}$ with respect to buckets whose capacity is not violated by $\hat{x}$, it remains to focus on violated buckets.

For such buckets, we first observe that, by the maximality of μ,

$$\sum_{m\in[\mu]} w_{i_m} > \text{capacity}\,(\mathcal{B}_k) - w_{i_{\mu+1}} \geq (1-4\epsilon)\cdot \text{capacity}(\mathcal{B}_k)\,, \tag{3.5}$$

where the second inequality holds since $i_{\mu+1} \in L_{k+1}$, and therefore $w_{i_{\mu+1}} \leq \epsilon^2\cdot(1+\epsilon)^{k+1} \leq 4\epsilon\cdot((1+\epsilon)^k - (1+\epsilon)^{k-1}) = 4\epsilon\cdot\text{capacity}(\mathcal{B}_k)$ for $\epsilon \in (0,1)$. On the other hand,

$$\begin{aligned}\sum_{m\in[M]} w_{i_m} &= \sum_{i\in L_{k+1}} w_i\hat{x}_{ik}\\ &\leq \text{capacity}(\mathcal{B}_k) + w_{i_{\inf(k)}}\\ &\leq (1+4\epsilon)\cdot\text{capacity}(\mathcal{B}_k)\,,\end{aligned} \tag{3.6}$$

where the equality above follows from how the indices $i_1,\ldots,i_M$ were defined, the first inequality is precisely the fixable capacity property of $\hat{x}$ (see (3.4)), and the second inequality holds since $w_{i_{\inf(k)}} \leq 4\epsilon\cdot\text{capacity}(\mathcal{B}_k)$, as explained earlier for $w_{i_{\mu+1}}$. Consequently,

$$\sum_{i\in L_{k+1}} q_{ik}\hat{x}^{-}_{ik} = \sum_{m\in[\mu]} q_{i_m,k}$$

$$\begin{aligned} &\geq \frac{\sum_{m\in[\mu]} w_{i_m}}{\sum_{m\in[M]} w_{i_m}} \cdot \sum_{m\in[M]} q_{i_m,k} \\ &\geq \frac{1-4\epsilon}{1+4\epsilon} \cdot \sum_{m\in[M]} q_{i_m,k} \\ &\geq (1-8\epsilon) \cdot \sum_{i\in L_{k+1}} q_{ik}\hat{x}_{ik} \ , \end{aligned}$$

where the first inequality holds since $\frac{q_{i_1,k}}{w_{i_1}} \geq \cdots \geq \frac{q_{i_M,k}}{w_{i_M}}$, and the second inequality is obtained by plugging in (3.5) and (3.6). □

Performance guarantee. We conclude by noting that, since $\hat{x}^-$ is a feasible solution to (GAP-IP), Lemma 3.2.8 allows us to construct a permutation π_{light} with an overall profit of

$$\begin{aligned} \Psi(\pi_{\text{light}}) &\geq \sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik}\hat{x}^-_{ik} \\ &\geq (1-8\epsilon) \cdot \text{OPT(GAP-LP)} \\ &\geq (1-13\epsilon) \cdot \Psi_{\text{light}}(\pi^*) \ , \end{aligned}$$

where the second and third inequalities follow from Lemmas 3.2.10 and 3.2.9, respectively.

From a running time perspective, the computational bottleneck of our approach is the Shmoys-Tardos algorithm [60]. As the latter is applied to a maximum generalized assignment instance consisting of n items and $O(K) = O(\frac{|\mathcal{I}|}{\epsilon})$ buckets, it requires $O((\frac{|\mathcal{I}|}{\epsilon})^{O(1)})$ time in total. Beyond that, restoring the feasibility of $\hat{x}$ and translating the resulting solution $\hat{x}^-$ back to a permutation can both be implemented in $O((nK)^{O(1)})$ time.

3.3 QPTAS for bounded weight ratio

In this section, we develop an approximation scheme for the generalized incremental knapsack problem by embedding our LP-based approach for competing against light contributions within a self-improving algorithm. As formally stated in Theorem 3.3.1 below, the running time of this algorithm will be exponentially-dependent on $\log(n \cdot \frac{w_{\max}}{w_{\min}})$, meaning that it provides a

quasi-polynomial time approximation scheme (QPTAS) when the ratio between the extremal item weights is polynomial in the input size. In Section 3.4, these ideas will be exploited within an approximate dynamic programming framework to derive a true QPTAS, without making any assumptions on the ratio $\frac{w_{\max}}{w_{\min}}$.

Theorem 3.3.1. *For any error parameter $\epsilon \in (0, 1)$, the generalized incremental knapsack problem can be approximated within a factor of $1 - \epsilon$ in time $O((nT)^{O(\frac{1}{\epsilon^5} \cdot \log(n \cdot \frac{w_{\max}}{w_{\min}}))} \cdot |\mathcal{I}|^{O(1)})$.*

Outline. As an instructive step, we dedicate Section 3.3.1 to explaining how, given any feasible chain, one can define a residual instance on the remaining (non-inserted) items. In this context, we establish a number of structural properties that relate between the solution spaces of the original and residual instances, which will be useful moving forward. As explained in Section 3.3.2, the basic idea behind our "self-improving" algorithm resides in arguing that, given a black-box α-approximation for the generalized incremental knapsack problem, efficient guessing methods can be utilized to construct a solution that optimally competes against heavy contributions, and simultaneously, α-competes against light contributions. In Section 3.3.3, we combine this result with our near-optimal algorithm for light contributions and attain a performance guarantee of $\frac{1}{2-\alpha}$, up to lower-order terms. Repeated applications of these $\alpha \mapsto \frac{1}{2-\alpha}$ improvements will be shown to obtain a $(1 - \epsilon)$-fraction of the optimal profit within $O(\frac{1}{\epsilon})$ rounds. It is important to mention that each such application by itself incurs an exponential dependency on $\log(n \cdot \frac{w_{\max}}{w_{\min}})$, meaning that the results of this section are incomparable to those stated in Theorem 1.3.2, where the running time involved is truly polynomial for any fixed $\epsilon > 0$.

3.3.1 Residual instances and their properties

Instance representation. Due to working with modified instances in subsequent sections, we will designate the underlying set of items in a given instance by $\mathcal{N}$. As before, each item $i \in \mathcal{N}$ is associated with a weight of w_i, each time period $t \in [T]$ has a capacity of W_t, and we gain a profit of p_{it} for introducing item i in period t. That said, what differentiates between one instance and

the other are two ingredients: The item set $\mathcal{N}$ and the time period capacities $W = (W_1, \ldots, W_T)$ with respect to which these instances are defined. It is important to point out that, regardless of the instance being considered, the item weights w_i, the number of time periods T, and the item-to-period profits p_{it} will be kept unchanged. For these reasons, we denote a generalized incremental knapsack instance simply by $\mathcal{I} = (\mathcal{N}, W)$.

The $|_G$-operator. In the following, we introduce additional definitions, notation, and structural properties related to modified instances and their solution space. For a pair of chains, $\mathcal{S} = (S_1, \ldots, S_T)$ and $\mathcal{G} = (G_1, \ldots, G_T)$, we define the *union of $\mathcal{S}$ and $\mathcal{G}$* as $\mathcal{S} \cup \mathcal{G} = (S_1 \cup G_1, \ldots, S_T \cup G_T)$, which is clearly a chain itself. For a chain $\mathcal{S}$ and a subset of items $G \subseteq \mathcal{N}$, we denote by $\mathcal{S}|_G$ the restriction of $\mathcal{S}$ to G, namely, $\mathcal{S}|_G = (S_1 \cap G, \ldots, S_T \cap G)$; one can easily verify that $\mathcal{S}|_G$ is a chain as well. The next observation, whose straightforward proof is omitted, establishes the feasibility of $\mathcal{S}|_G$ whenever $\mathcal{S}$ is feasible.

Observation 3.3.2. *Let $\mathcal{S}$ be a feasible chain for $\mathcal{I}$. Then, for any set of items $G \subseteq \mathcal{N}$, the chain $\mathcal{S}|_G$ is feasible as well.*

The residual instance. Given a feasible chain $\mathcal{G} = (G_1, \ldots, G_T)$ for an instance $\mathcal{I} = (\mathcal{N}, W)$, we define the *residual* generalized incremental knapsack instance $\mathcal{I}^{-\mathcal{G}} = (\mathcal{N}^{-\mathcal{G}}, W^{-\mathcal{G}})$ as follows:

- The new set of items is $\mathcal{N}^{-\mathcal{G}} = \mathcal{N} \setminus G_T$. Namely, we eliminate all items that were introduced at any point in time by $\mathcal{G}$.
- The residual capacity of every time $t \in [T]$ is set to $W_t^{-\mathcal{G}} = \min_{t \le \tau \le T}(W_\tau - w(G_\tau))$.
- As previously mentioned, all item weights and profits remain unchanged.

To verify that the residual instance $\mathcal{I}^{-\mathcal{G}}$ is well defined, it suffices to show that the residual capacities $W^{-\mathcal{G}}$ are non-negative and non-decreasing over time. The former property holds since $w(G_t) \le W_t$ for every $t \in [T]$, by feasibility of $\mathcal{G}$. The latter property follows by observing that

$$W_t^{-\mathcal{G}} = \min_{t \le \tau \le T}(W_\tau - w(G_\tau)) \le \min_{t+1 \le \tau \le T}(W_\tau - w(G_\tau)) = W_{t+1}^{-\mathcal{G}} .$$

The next two claims, whose respective proofs appear in Appendices C.2.1 and C.2.2, explain the relationship between the solution spaces of the original instance $\mathcal{I}$ and its residual instance $\mathcal{I}^{-\mathcal{G}}$. For our purposes, the main implication of this relationship will be that, whenever we are able to "guess" a chain $\mathcal{G} = \mathcal{S}^*|_G$, where $\mathcal{S}^*$ is an optimal chain for $\mathcal{I}$, it suffices to focus on solving the residual instance $\mathcal{I}^{-\mathcal{G}}$. With an appropriate guess for the set of items G, this property will be a key idea within the approximation scheme we devise in the remainder of this section.

Lemma 3.3.3. *Let $\mathcal{G}$ be a feasible chain for $\mathcal{I}$ and let $\mathcal{R}$ be a feasible chain for $\mathcal{I}^{-\mathcal{G}}$. Then, $\mathcal{G} \cup \mathcal{R}$ is a feasible chain for $\mathcal{I}$ with profit $\Phi(\mathcal{G} \cup \mathcal{R}) = \Phi(\mathcal{G}) + \Phi(\mathcal{R})$.*

Lemma 3.3.4. *Let $\mathcal{S}$ be a feasible chain for $\mathcal{I}$ and let $\mathcal{G} = \mathcal{S}|_G$, for some set of items $G \subseteq \mathcal{N}$. Then, $\mathcal{S}|_{\mathcal{N}\setminus G}$ is a feasible chain for $\mathcal{I}^{-\mathcal{G}}$ with profit $\Phi(\mathcal{S}|_{\mathcal{N}\setminus G}) = \Phi(\mathcal{S}) - \Phi(\mathcal{G})$. Moreover, if $\mathcal{S}$ is optimal for $\mathcal{I}$, then $\mathcal{S}|_{\mathcal{N}\setminus G}$ is optimal for $\mathcal{I}^{-\mathcal{G}}$.*

3.3.2 The boosting algorithm

Given a generalized incremental knapsack instance $\mathcal{I} = (\mathcal{N}, W)$, let us focus our attention on a fixed optimal chain $\mathcal{S}^*$. As argued in Lemma 3.2.1, this chain can be mapped to a permutation $\pi_{\mathcal{S}^*} : \mathcal{N} \to [|\mathcal{N}|]$ whose objective value with respect to the corresponding sequencing formulation is $\Psi(\pi_{\mathcal{S}^*}) \geq \Phi(\mathcal{S}^*)$. By decomposing the overall profit $\Psi(\pi_{\mathcal{S}^*})$ into heavy and light contributions, as prescribed by Equation (3.1), we have:

$$\Psi(\pi_{\mathcal{S}^*}) = \underbrace{\sum_{k\in[K]_0} \sum_{\substack{i\in H_k:\\ C_{\pi_{\mathcal{S}^*}}(i)\in\mathcal{I}_k}} \varphi_{\pi_{\mathcal{S}^*}}(i)}_{\Psi_{\text{heavy}}(\pi_{\mathcal{S}^*})} + \underbrace{\sum_{k\in[K]_0} \sum_{\substack{i\in L_k:\\ C_{\pi_{\mathcal{S}^*}}(i)\in\mathcal{I}_k}} \varphi_{\pi_{\mathcal{S}^*}}(i)}_{\Psi_{\text{light}}(\pi_{\mathcal{S}^*})} . \tag{3.7}$$

Given these quantities, for $\alpha_H, \alpha_L \in [0, 1]$, we say that an algorithm $\mathcal{A}$ guarantees an (α_H, α_L)-approximation with respect to $\mathcal{S}^*$ when it computes a feasible chain $\mathcal{S}$ with $\Phi(\mathcal{S}) \geq \alpha_H \cdot \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \alpha_L \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$. We mention in passing that this definition depends on the specific permutation $\pi_{\mathcal{S}^*}$, and is generally different from the standard notion of an α-approximation, where the chain $\mathcal{S}$ is required to satisfy $\Phi(\mathcal{S}) \geq \alpha \cdot \Phi(\mathcal{S}^*)$.

From α-approximation to $(1, \alpha)$-approximation. In what follows, we show how to boost the profit performance of any approximation algorithm for the generalized incremental knapsack problem. For every $\alpha \in [0, 1]$, we explain how to combine a black-box α-approximation with further guesses for the positioning of heavy items with respect to the permutation $\pi_{\mathcal{S}^*}$ in order to derive a $(1, \alpha)$-approximation, incurring an extra multiplicative factor of $O((nT)^{O(\frac{1}{\epsilon^2}\log(n\rho))})$ in running time, where $\rho = \frac{w_{\max}}{w_{\min}}$. This result can be formally stated as follows.

Lemma 3.3.5. *Suppose that the algorithm $\mathcal{A}$ constitutes an α-approximation for generalized incremental knapsack, for some $\alpha \in [0, 1]$. Then, there exists a $(1, \alpha)$-approximation whose running time is $O((nT)^{O(\frac{1}{\epsilon^2}\log(n\rho))} \cdot \mathrm{Time}_{\mathcal{A}}(n, T))$. Here, $\mathrm{Time}_{\mathcal{A}}(n, T)$ designates the worst-case running time of $\mathcal{A}$ for instances with n items and T time periods.*

Preliminaries. We remind the reader that Section 3.2.2 has previously defined the intervals $\mathcal{I}_0 = [0, 1)$ and $\mathcal{I}_k = ((1+\epsilon)^{k-1}, (1+\epsilon)^k]$ for $k \in [K]$, where $K = \lceil \log_{1+\epsilon}(\sum_{i\in[n]} w_i) \rceil$; similarly, we assume without loss of generality that $w_{\min} \geq 3$. In this regard, an item i is k-heavy when $w_i \geq \epsilon^2 \cdot (1+\epsilon)^k$, with the convention that H_k stands for the collection of k-heavy items. Let $G^{*\mathrm{heavy}}$ be the set of items that are heavy for the interval that contains their completion time with respect to the permutation $\pi_{\mathcal{S}^*}$, i.e., $G^{*\mathrm{heavy}} = \bigcup_{k\in[K]_0} \{i \in H_k : C_{\pi_{\mathcal{S}^*}}(i) \in \mathcal{I}_k\}$. The following lemma, whose proof appears in Appendix C.2.3, provides an upper bound on the cardinality of this set.

Lemma 3.3.6. $|G^{*\mathrm{heavy}}| \leq \frac{3\log(n\rho)}{\epsilon^2}$.

We proceed by considering the restriction of the optimal chain $\mathcal{S}^*$ to the set of items $G^{*\mathrm{heavy}}$, which will be denoted by $\mathcal{H}^* = \mathcal{S}^*|_{G^{*\mathrm{heavy}}}$. By Observation 3.3.2, we know that $\mathcal{H}^*$ is a feasible chain for $\mathcal{I}$. The next lemma, whose proof can be found in Appendix C.2.4, relates between the profit of this chain and heavy contributions with respect to the permutation $\pi_{\mathcal{S}^*}$.

Lemma 3.3.7. $\Phi(\mathcal{H}^*) = \Psi_{\mathrm{heavy}}(\pi_{\mathcal{S}^*})$.

The algorithm. At a high level, our algorithm relies on "knowing" the restricted chain $\mathcal{H}^*$ in advance, which will be justified by guessing all items in $G^{*\text{heavy}}$ and their insertion times with respect to the optimal chain $\mathcal{S}^*$. This procedure will be implemented by enumerating over all possible configurations of these parameters. For each such guess, we construct the residual generalized incremental knapsack instance, to which the α-approximation algorithm $\mathcal{A}$ is applied. Formally, given an instance $\mathcal{I} = (\mathcal{N}, W)$ and an error parameter $\epsilon > 0$, we proceed as follows:

1. For every feasible chain $\mathcal{G} = (G_1, \ldots, G_T)$ with $|G_T| \leq \frac{3\log(n\rho)}{\epsilon^2}$:
   (a) Construct the residual instance $\mathcal{I}^{-\mathcal{G}}$.
   (b) Apply the algorithm $\mathcal{A}$ to obtain an α-approximate feasible chain $\mathcal{S}^{-\mathcal{G}} = (S_1^{-\mathcal{G}}, \ldots, S_T^{-\mathcal{G}})$ for $\mathcal{I}^{-\mathcal{G}}$.
2. Return the chain $\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}$ of maximum profit among those considered above.

Analysis: Feasibility and running time. We first observe that, for any feasible chain $\mathcal{G}$ constructed in step 1, since $\mathcal{S}^{-\mathcal{G}}$ is a feasible chain for $\mathcal{I}^{-\mathcal{G}}$, the feasibility of $\mathcal{G} \cup \mathcal{S}^{-\mathcal{G}}$ for $\mathcal{I}$ follows by Lemma 3.3.3. In terms of running time, we are considering only chains that introduce at most $\frac{3\log(n\rho)}{\epsilon^2}$ items over all time periods. Thus, the number of chains being enumerated is $O((nT)^{O(\frac{1}{\epsilon^2}\log(n\rho))})$. For each residual instance, consisting of T time periods and at most n items, we apply the algorithm $\mathcal{A}$ once, implying that the overall running time is indeed $O((nT)^{O(\frac{1}{\epsilon^2}\log(n\rho))} \cdot \text{Time}_{\mathcal{A}}(n, T))$.

Analysis: $(1, \alpha)$-approximation guarantee. We conclude the proof of Lemma 3.3.5 by arguing that $\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}$ is a $(1, \alpha)$-approximate chain with respect to $\mathcal{S}^*$ for the original instance $\mathcal{I}$.

Lemma 3.3.8. $\Phi(\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}) \geq \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \alpha \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$.

Proof. We begin by observing that the feasible chain $\mathcal{H}^* = \mathcal{S}^*|_{G^{*\text{heavy}}}$ is one of those considered in step 1. To verify this claim, note that $|G^{*\text{heavy}}| \leq \frac{3\log(n\rho)}{\epsilon^2}$ by Lemma 3.3.6, meaning that $\mathcal{H}^*$ introduces at most that many items across all time periods. As a result, since the chain $\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}$

attains a maximum profit among those considered, we have $\Phi(\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}) \geq \Phi(\mathcal{H}^* \cup \mathcal{S}^{-\mathcal{H}^*})$, and it remains to prove that $\Phi(\mathcal{H}^* \cup \mathcal{S}^{-\mathcal{H}^*}) \geq \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \alpha \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$.

For this purpose, let $\mathcal{L}^* = \mathcal{S}^*|_{\mathcal{N} \setminus G^{*\text{heavy}}}$ be the restriction of $\mathcal{S}^*$ to the set $\mathcal{N} \setminus G^{*\text{heavy}}$, which is a feasible chain for $\mathcal{I}$ by Observation 3.3.2. We next show that $\Phi(\mathcal{L}^*) = \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$. In order to derive this claim, note that since L_T^* and H_T^* are disjoint and $\mathcal{S}^* = \mathcal{H}^* \cup \mathcal{L}^*$, it follows that

$$\begin{aligned}
\Phi(\mathcal{L}^*) &= \Phi(\mathcal{S}^*) - \Phi(\mathcal{H}^*) \\
&= \Phi(\mathcal{S}^*) - \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) \\
&= \Psi(\pi_{\mathcal{S}^*}) - \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) \\
&= \Psi_{\text{light}}(\pi_{\mathcal{S}^*}) \,,
\end{aligned}$$

where the second equality holds due to Lemma 3.3.7, the third equality is obtained by recalling that $\Psi(\pi_{\mathcal{S}^*}) = \Phi(\mathcal{S}^*)$, as shown along the proof of Lemma 3.3.7, and the last equality follows from the profit decomposition (3.7).

However, the crucial observation is that $\mathcal{L}^*$ is a feasible chain for the residual instance $\mathcal{I}^{-\mathcal{H}^*}$, by Lemma 3.3.4. Consequently, since the algorithm $\mathcal{A}$ computes an α-approximate feasible chain $\mathcal{S}^{-\mathcal{H}^*}$ for the latter instance, $\Phi(\mathcal{S}^{-\mathcal{H}^*}) \geq \alpha \cdot \Phi(\mathcal{L}^*) = \alpha \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$, implying that $\mathcal{H}^* \cup \mathcal{S}^{-\mathcal{H}^*}$ indeed has a profit of $\Phi(\mathcal{H}^* \cup \mathcal{S}^{-\mathcal{H}^*}) = \Phi(\mathcal{H}^*) + \Phi(\mathcal{S}^{-\mathcal{H}^*}) \geq \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \alpha \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$. □

3.3.3 The ratio improvement and final algorithm

We proceed by revealing the self-improving feature of our approach, by showing that a $(1, \alpha)$-approximation for generalized incremental knapsack leads in turn to a $\frac{1-\delta}{2-\alpha}$-approximation, when combined with our algorithm for light items, presented in Section 3.2.4. We will then show how to recursively apply this self-improving idea to eventually derive an approximation scheme.

Lemma 3.3.9. *Suppose that, for some $\alpha \in [0, 1]$, the algorithm $\mathcal{A}$ constitutes an α-approximation. Then, for any error parameter $\delta > 0$, the generalized incremental knapsack problem can be approximated within factor $\frac{1-\delta}{2-\alpha}$ in time $O((nT)^{O(\frac{1}{\delta^2}\log(n\rho))} \cdot \text{Time}_{\mathcal{A}}(n, T) + (\frac{|\mathcal{I}|}{\delta})^{O(1)})$.*

Proof. As explained in Section 3.3.2, the optimal chain $\mathcal{S}^*$ can be mapped to a permutation $\pi_{\mathcal{S}^*}$ whose overall profit $\Psi(\pi_{\mathcal{S}^*})$ decomposes into heavy and light contributions, $\Psi(\pi_{\mathcal{S}^*}) = \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$. Now, on the one hand, Lemma 3.3.5 provides us with a $(1, \alpha)$-approximation in $O((nT)^{O(\frac{1}{\delta^2}\log(n\rho))} \cdot \text{Time}_{\mathcal{A}}(n, T))$ time. That is, we obtain a feasible chain $\mathcal{S}_{(1,\alpha)}$ with $\Phi(\mathcal{S}_{(1,\alpha)}) \geq \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \alpha \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$. On the other hand, the main result of Section 3.2.4 allows us to compute in $O((\frac{|\mathcal{I}|}{\delta})^{O(1)})$ time a permutation π_{light} with a profit of $\Psi(\pi_{\text{light}}) \geq (1-\delta) \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$. By converting this permutation to a feasible chain $\mathcal{S}_{(0,1-\delta)}$ along the lines of Lemma 3.2.1, we clearly obtain a $(0, 1-\delta)$-approximation, meaning that $\Phi(\mathcal{S}_{(0,1-\delta)}) \geq (1-\delta) \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})$. Our combined approach independently employs both algorithms and returns the more profitable of the two feasible chains computed, $\mathcal{S}_{(1,\alpha)}$ and $\mathcal{S}_{(0,1-\delta)}$, to obtain a profit of

$$\begin{aligned}
\max\left\{\Phi(\mathcal{S}_{(1,\alpha)}), \Phi(\mathcal{S}_{(0,1-\delta)})\right\} &\geq \max\Big\{\Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \alpha \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*}), \\
&\qquad (1-\delta) \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})\Big\} \\
&\geq \frac{1}{2-\alpha} \cdot \left(\Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \alpha \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*})\right) \\
&\quad + \left(1 - \frac{1}{2-\alpha}\right) \cdot (1-\delta) \cdot \Psi_{\text{light}}(\pi_{\mathcal{S}^*}) \\
&\geq \frac{1-\delta}{2-\alpha} \cdot \left(\Psi_{\text{heavy}}(\pi_{\mathcal{S}^*}) + \Psi_{\text{light}}(\pi_{\mathcal{S}^*})\right) \\
&= \frac{1-\delta}{2-\alpha} \cdot \Psi(\pi_{\mathcal{S}^*}) \\
&\geq \frac{1-\delta}{2-\alpha} \cdot \Phi(\mathcal{S}^*),
\end{aligned}$$

where the last inequality follows from Lemma 3.2.1. □

The final approximation scheme. We conclude by explaining how our $\alpha \mapsto \frac{1-\delta}{2-\alpha}$ improvement, outlined in Lemma 3.3.9, can be iteratively applied to derive an approximation scheme for the generalized incremental knapsack problem, thereby completing the proof of Theorem 3.3.1.

For the purpose of ensuring a $(1-\epsilon)$-fraction of the optimal profit, we will set the error parameter δ in Lemma 3.3.9 as a function of ϵ, where the exact dependency will be determined later on.

Given this self-improving result, we define a sequence of algorithms $\mathcal{A}_0, \mathcal{A}_1, \ldots$, with the convention that the approximation ratio of each such algorithm $\mathcal{A}_r$ is denoted by α_r. Specifically, this sequence begins with the trivial algorithm $\mathcal{A}_0$ that returns an empty solution $(\emptyset, \ldots, \emptyset)$, meaning that $\alpha_0 = 0$. Then, by applying Lemma 3.3.9 with respect to $\mathcal{A}_0$, we obtain the algorithm $\mathcal{A}_1$, for which $\alpha_1 = \frac{1-\delta}{2}$. Subsequently, by a similar application with respect to $\mathcal{A}_1$, we obtain $\mathcal{A}_2$, with $\alpha_2 = \frac{1-\delta}{2-\alpha_1}$. In general, for every integer $r \geq 1$, the resulting algorithm $\mathcal{A}_r$ guarantees an approximation ratio of $\alpha_r = \frac{1-\delta}{2-\alpha_{r-1}}$. The next lemma, whose proof is presented in Appendix C.2.5, provides a closed-form lower bound on α_r.

Lemma 3.3.10. *$\alpha_r \geq \frac{r}{r+1} - r\delta$, for every $r \geq 0$.*

By choosing $\delta = \frac{\epsilon^2}{2}$, the above lemma implies that $\lceil \frac{2}{\epsilon} \rceil$ self-improving rounds produce an algorithm $\mathcal{A}_{\lceil \frac{2}{\epsilon} \rceil}$ for computing a feasible chain $\mathcal{S}$ with a profit of $\Phi(\mathcal{S}) \geq (\frac{\lceil 2/\epsilon \rceil}{\lceil 2/\epsilon \rceil + 1} - \lceil \frac{2}{\epsilon} \rceil \cdot \frac{\epsilon^2}{2}) \cdot \Phi(\mathcal{S}^*) \geq (1-\epsilon) \cdot \Phi(\mathcal{S}^*)$, thereby deriving the approximation guarantee of Theorem 3.3.1. Furthermore, it is not difficult to verify that algorithm $\mathcal{A}_{\lceil \frac{2}{\epsilon} \rceil}$ runs in $O((nT)^{O(\frac{1}{\epsilon^5} \cdot \log(n\rho))} \cdot |\mathcal{I}|^{O(1)})$ time, by induction on r.

3.4 QPTAS for general instances

Thus far, we have developed an approximation scheme whose running time includes an exponential dependency on $\log(n \cdot \frac{w_{\max}}{w_{\min}})$, leading to a quasi-PTAS for problem instances where the ratio $\frac{w_{\max}}{w_{\min}}$ is polynomial in the input size. In what follows, we show how to obtain a true quasi-PTAS, without any assumptions on $\frac{w_{\max}}{w_{\min}}$.

Theorem 3.4.1. *For any error parameter $\epsilon \in (0, 1)$, the generalized incremental knapsack problem can be approximated within a factor of $1 - \epsilon$ in time $O(|\mathcal{I}|^{O((\frac{1}{\epsilon} \log |\mathcal{I}|)^{O(1)})})$.*

Interestingly, our algorithmic approach shows that any $(1-\epsilon)$-approximation running in $\mathcal{T}(n, T, \frac{w_{\max}}{w_{\min}})$ time can be executed in black-box fashion on appropriately-constructed instances with $\frac{w_{\max}}{w_{\min}} = O(n^{1/\epsilon})$, leading to a $(1-\epsilon)$-approximation for the general problem formulation in $O(|\mathcal{I}|^{O((\frac{1}{\epsilon} \log |\mathcal{I}|)^{O(1)})}) \cdot \mathcal{T}(n, T, n^{1/\epsilon})$ time.

3.4.1 Technical overview

Step 1: Creating a well-spaced instance. We begin by slightly altering a given instance $\mathcal{I} = (\mathcal{N}, W)$, with the objective of creating nearly-ideal circumstances for the approximation scheme of Section 3.3 to operate, losing negligible profits along the way. For this purpose, given an error parameter $\epsilon > 0$, we say that the instance $\mathcal{I}$ is well-spaced when its set of items $\mathcal{N}$ can be partitioned into clusters $C_1, \ldots, C_M$ satisfying the following properties:

1. *Weight ratio within clusters*: For every $m \in [M]$, the weights of any two items in cluster C_m differ by a multiplicative factor of at most $n^{1/\epsilon}$.

2. *Weight gap between clusters*: For every $m_1, m_2 \in [M]$ with $m_1 < m_2$, the weight of any item in cluster C_{m_2} is greater than the weight of any item in cluster C_{m_1} by a multiplicative factor of at least $n^{1+(m_2-m_1-1)/\epsilon}$.

In Section 3.4.2, we show that one can efficiently identify a subset of items over which the induced instance is well-spaced, while still admitting a near-optimal solution. We derive this result, as formally stated below, through an application of the shifting method (see, for instance, [5, 41]).

Lemma 3.4.2. *There exists an item set $\mathcal{N}_{\text{spaced}} \subseteq \mathcal{N}$ for which $\mathcal{I}_{\text{spaced}} = (\mathcal{N}_{\text{spaced}}, W)$ is a well-spaced instance, whose optimal chain $\mathcal{S}_{\text{spaced}}$ guarantees a profit of $\Phi(\mathcal{S}_{\text{spaced}}) \geq (1 - \epsilon) \cdot \Phi(\mathcal{S}^*)$. Such a set can be determined in $O((n/\epsilon)^{O(1)})$ time.*

Step 2: Proving the sparse-crossing property. For simplicity of notation, we assume from this point on that the instance $\mathcal{I} = (\mathcal{N}, W)$ is well-spaced, with clusters $C_1, \ldots, C_M$. Now suppose that the optimal permutation π^* for the sequencing-based formulation of this instance was known to be "crossing-free", namely, items belonging to cluster C_1 appear first in π^*, followed by those belonging to cluster C_2, so on and so forth. In other words, a left-to-right scan of the permutation π^* reveals that it is weakly-increasing by cluster. In this ideal situation, the approximation scheme we propose in Section 3.3 can be sequentially employed to the clusters $C_1, \ldots, C_M$ in increasing

order. This way, we would have obtained a $(1-\epsilon)$-approximation in truly quasi-polynomial time, since the extremal weight ratio within each cluster is $n^{1/\epsilon}$-bounded, by property 1.

Unfortunately, elementary examples show that an optimal permutation π^* may not be crossing-free, in the sense that items in any given cluster can be preceded by items belonging to higher-index clusters. That said, a suitable relaxation of these ideas can still be exploited. Formally, let us denote by $\mathrm{cross}_m(\pi)$ the number of items in clusters $C_{m+1}, \ldots, C_M$ that appear in the permutation π before the last item belonging to cluster C_m; note that crossing-free is equivalent to having $\mathrm{cross}_1(\pi) = \cdots = \mathrm{cross}_M(\pi) = 0$. Our next structural result, formally established in Section 3.4.3, proves the existence of a near-optimal permutation with very few items crossing each cluster.

Lemma 3.4.3. *There exist an item set $\mathcal{N}_{\mathrm{sparse}} \subseteq \mathcal{N}$ and a permutation $\pi_{\mathrm{sparse}} : \mathcal{N}_{\mathrm{sparse}} \to [|\mathcal{N}_{\mathrm{sparse}}|]$ satisfying:*

1. *Sparse crossing:* $\max_{m \in [M]} \mathrm{cross}_m(\pi_{\mathrm{sparse}}) \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$.
2. *Near-optimal profit:* $\Psi(\pi_{\mathrm{sparse}}) \geq (1-\epsilon) \cdot \Psi(\pi^*)$.

Technically speaking, our proof is based on applying a sequence of recursive transformations with respect to the unknown optimal permutation π^*. To convey the high-level idea, let i_{mid} be the last-appearing item in π^* out of clusters $C_1, \ldots, C_{M/2}$. When fewer than $1/\epsilon$ items in clusters $C_{(M/2)+1}, \ldots, C_M$ appear before i_{mid}, each of the clusters $C_1, \ldots, C_{M/2}$ has at most $1/\epsilon$ crossings due to items in $C_{(M/2)+1}, \ldots, C_M$. We can therefore recursively proceed into the left part of π^*, stretching up to the item i_{mid}, and into its right part, consisting of the remaining items. In the opposite case, where at least $1/\epsilon$ items in clusters $C_{(M/2)+1}, \ldots, C_M$ appear before i_{mid}, the important observation is that we can eliminate the cheapest out of the first $1/\epsilon$ such items while losing only an $O(\epsilon)$-fraction of their combined profit. However, since this item is heavier than any item in lower-index clusters by a factor of at least n (see property 2), the gap we have just created is sufficiently large to pull back each and every item in clusters $C_1, \ldots, C_{M/2}$, only increasing their profit contributions. We can now recursively proceed into the left and right parts.

Step 3: The external dynamic program. Given the sparse-crossing property, we dedicate Section 3.4.4 to proposing a dynamic programming approach for computing a near-optimal permutation. For this purpose, by recycling some of the notation introduced in Section 3.2.3, our state description $(m, \psi_m, Q_{>m})$ will consists of the following parameters:

- The index of the current cluster, m.
- The profit requirement, ψ_m.
- The set of items $Q_{>m}$ belonging to clusters $C_{m+1}, \ldots, C_M$ that will be crossing into lower-index clusters, noting that Lemma 3.4.3 allows us to consider only small sets, of size $O(\frac{\log M}{\epsilon})$.

At a high level, the value function $F(m, \psi_m, Q_{>m})$ will represent the minimum makespan $w(S)$ that can be attained, over all subset of items S within the union of $Q_{>m}$ and the clusters $C_1, \ldots, C_m$ (namely, $S \subseteq Q_{>m} \uplus (\biguplus_{\mu \in [m]} C_\mu)$) and over all permutations $\pi : S \rightarrow [|S|]$ that generate a total profit of at least ψ_m. Clearly, the best-possible profit of a sparse-crossing permutation corresponds to the maximal value ψ_M that satisfies $F(M, \psi_M, \emptyset) < \infty$, which is at least $(1 - \epsilon) \cdot \Psi(\pi^*)$, by Lemma 3.4.3.

As formally explained in Section 3.4.4, within the recursive equations for computing $F(m, \psi_m, Q_{>m})$, evaluating the marginal makespan increase of each possible action involves solving a single-cluster subproblem. Specifically for the latter, the approximation scheme we have devised in Section 3.3 will be shown to incur a quasi-polynomial running time. In parallel, the dominant factor in determining the underlying number of states emerges from the set of items $Q_{>m}$, taking $O(n^{O(\frac{1}{\epsilon} \log M)})$ possible values, respectively, thus forming the second source of quasi-polynomiality in our approach and concluding the proof of Theorem 3.4.1.

3.4.2 Proof of Lemma 3.4.2: Creating a well-spaced instance

Bucketing. For the purpose of identifying the desired subset $\mathcal{N}_{\text{spaced}}$, we initially partition the overall collection of items $\mathcal{N}$ into buckets $\mathcal{B}_1, \ldots, \mathcal{B}_L$ according to their weights. This partition

will be geometric, by powers of n, meaning that $L = \lceil \log_n(\frac{w_{\max}}{w_{\min}}) \rceil + 1$. Specifically, the first bucket $\mathcal{B}_1$ consists of items whose weight resides in $[w_{\min}, n \cdot w_{\min})$, the second bucket $\mathcal{B}_2$ consists of those with weight in $[n \cdot w_{\min}, n^2 \cdot w_{\min})$, so on and so forth, where in general, bucket $\mathcal{B}_\ell$ corresponds to the interval $[n^{\ell-1} \cdot w_{\min}, n^\ell \cdot w_{\min})$. It is easy to verify that $\mathcal{B}_1, \ldots, \mathcal{B}_L$ is indeed a partition of $\mathcal{N}$.

Creating clusters. Now let $r \in \{0, \ldots, \frac{1}{\epsilon} - 1\}$ be an integer parameter whose value will be determined later. Accordingly, we create a subset of items $\mathcal{N}_r \subseteq \mathcal{N}$, that will be clustered into $C_1^r, \ldots, C_M^r$ with $M = O(\epsilon L)$, as follows. Intuitively, we introduce "gaps" within the sequence of buckets $\mathcal{B}_1, \ldots, \mathcal{B}_L$, spaced apart by $\frac{1}{\epsilon}$ indices, through eliminating every bucket $\mathcal{B}_\ell$ with $\ell \mod \frac{1}{\epsilon} = r$; then, between every pair of successive gaps, buckets will be unified to form a single cluster. That is, the first cluster is defined as $C_1^r = \biguplus_{\ell=1}^{r-1} \mathcal{B}_\ell$, the second cluster is $C_2^r = \biguplus_{\ell=r+1}^{r-1+1/\epsilon} \mathcal{B}_\ell$, the third is $C_3^r = \biguplus_{\ell=r+1+1/\epsilon}^{r-1+2/\epsilon} \mathcal{B}_\ell$, and so on. Finally, we define the subset of items $\mathcal{N}_r$ as the union of all clusters, i.e., $\mathcal{N}_r = \biguplus_{m \in [M]} C_m^r$, with a corresponding generalized incremental knapsack instance $\mathcal{I}_r = (\mathcal{N}_r, W)$.

Analysis. In what follows, we argue that for every $r \in \{0, \ldots, \frac{1}{\epsilon} - 1\}$, the instance $\mathcal{I}_r$ we have just constructed is in fact well-spaced, via the partition of $\mathcal{N}_r$ into clusters $C_1^r, \ldots, C_M^r$. For this purpose, we separately prove each of the required well-spaced properties.

1. *Weight ratio within clusters*: Consider two items i_1 and i_2 belonging to the same cluster C_m^r. Letting $\mathcal{B}_{\ell_1}$ and $\mathcal{B}_{\ell_2}$ be the buckets containing these items, respectively, their weight ratio can be upper bounded by observing that

$$\begin{aligned} \frac{w_{i_2}}{w_{i_1}} &\leq \frac{\max_{i \in \mathcal{B}_{\ell_2}} w_i}{\min_{i \in \mathcal{B}_{\ell_1}} w_i} \\ &\leq n^{\ell_2 - (\ell_1 - 1)} \\ &\leq n^{(1/\epsilon)-1}, \end{aligned}$$

   where the second inequality holds since each bucket $\mathcal{B}_\ell$ contains items whose weight falls within $[n^{\ell-1} \cdot w_{\min}, n^\ell \cdot w_{\min})$, and the third inequality follows by noting that each cluster

represents the union of at most $\frac{1}{\epsilon} - 1$ successive buckets, implying that $\ell_2 - \ell_1 \leq \frac{1}{\epsilon} - 2$.

2. *Weight gap between clusters*: Similarly, let i_1 and i_2 be a pair of items that belong to clusters $C^r_{m_1}$ and $C^r_{m_2}$, respectively, with $m_1 < m_2$. In this case, when we denote the corresponding buckets by $\mathcal{B}_{\ell_1}$ and $\mathcal{B}_{\ell_2}$, their weight ratio can be lower bounded by

$$\begin{aligned} \frac{w_{i_2}}{w_{i_1}} &\geq \frac{\min_{i \in \mathcal{B}_{\ell_2}} w_i}{\max_{i \in \mathcal{B}_{\ell_1}} w_i} \\ &\geq n^{(\ell_2 - 1) - \ell_1} \\ &\geq n^{1 + (m_2 - m_1 - 1)/\epsilon}\,, \end{aligned}$$

where the last inequality holds since $\ell_1 \in \{r + 1 + \frac{m_1 - 2}{\epsilon}, \ldots, r - 1 + \frac{m_1 - 1}{\epsilon}\}$ and $\ell_2 \in \{r + 1 + \frac{m_2 - 2}{\epsilon}, \ldots, r - 1 + \frac{m_2 - 1}{\epsilon}\}$, by definition of $C^r_{m_1}$ and $C^r_{m_2}$.

We conclude the proof of Lemma 3.4.2 by showing that at least one of the well-spaced instances $\mathcal{I}_0, \ldots \mathcal{I}_{\frac{1}{\epsilon} - 1}$ is associated with an optimal profit of at least $(1 - \epsilon) \cdot \Phi(\mathcal{S}^*)$. To this end, with respect to the optimal chain $\mathcal{S}^*$ for the original instance $\mathcal{I}$, note that the restriction of this chain $\mathcal{S}^*|_{\mathcal{N}_r}$ to the item set $\mathcal{N}_r$ is clearly feasible for $\mathcal{I}_r$, by Observation 3.3.2. Letting $\mathcal{S}^{r*}$ be an optimal chain for $\mathcal{I}_r$, we consequently have

$$\begin{aligned} \max_{0 \leq r \leq (1/\epsilon) - 1} \Phi(\mathcal{S}^{r*}) &\geq \max_{0 \leq r \leq (1/\epsilon) - 1} \Phi(\mathcal{S}^*|_{\mathcal{N}_r}) \\ &\geq \epsilon \cdot \sum_{r=0}^{(1/\epsilon) - 1} \Phi(\mathcal{S}^*|_{\mathcal{N}_r}) \\ &= \epsilon \cdot \sum_{r=0}^{(1/\epsilon) - 1} \sum_{t \in [T]} \sum_{i \in (S^*_t \setminus S^*_{t-1}) \cap \mathcal{N}_r} p_{it} \\ &= \epsilon \cdot \sum_{t \in [T]} \sum_{i \in S^*_t \setminus S^*_{t-1}} \left| \left\{ r \in \left\{0, \ldots, \frac{1}{\epsilon} - 1\right\} : \right. \right. \\ &\qquad \left. \left. i \in (S^*_t \setminus S^*_{t-1}) \cap \mathcal{N}_r \right\} \right| \cdot p_{it} \\ &= (1 - \epsilon) \cdot \sum_{t \in [T]} \sum_{i \in S^*_t \setminus S^*_{t-1}} p_{it} \\ &= (1 - \epsilon) \cdot \Phi(\mathcal{S}^*)\,, \end{aligned}$$

where the next-to-last equality holds since every item introduced in the optimal chain $\mathcal{S}^*$ appears in all but one of the sets $\mathcal{N}_0, \ldots, \mathcal{N}_{(1/\epsilon)-1}$.

3.4.3 Proof of Lemma 3.4.3: The sparse-crossing property

Preliminaries. We begin by introducing some additional definitions and notation that will be utilized throughout this proof. For a set of cluster indices $\mathcal{M} \subseteq [M]$, we use $C_{\mathcal{M}}$ to designate the union of $\mathcal{M}$-indexed clusters, i.e., $C_{\mathcal{M}} = \biguplus_{m \in \mathcal{M}} C_m$. Expanding upon the definition of $\text{cross}_m(\pi)$, given disjoint sets, $\mathcal{M}_1 \subseteq [M]$ and $\mathcal{M}_2 \subseteq [M]$, let $\text{cross}_{\mathcal{M}_1,\mathcal{M}_2}(\pi)$ denote the number of items in $C_{\mathcal{M}_2}$ that appear in the permutation π before the last item in $C_{\mathcal{M}_1}$, namely,

$$\text{cross}_{\mathcal{M}_1,\mathcal{M}_2}(\pi) = \left| \left\{ i \in C_{\mathcal{M}_2} : \pi(i) < \max_{j \in C_{\mathcal{M}_1}} \pi(j) \right\} \right|.$$

When $\text{cross}_{\mathcal{M}_1,\mathcal{M}_2}(\pi) \geq \frac{1}{\epsilon}$, we use $\mathcal{X}_{\mathcal{M}_1,\mathcal{M}_2}(\pi)$ to designate the set comprised of the first $\frac{1}{\epsilon}$ items in $\mathcal{M}_2$-indexed clusters in the permutation π. When $\text{cross}_{\mathcal{M}_1,\mathcal{M}_2}(\pi) < \frac{1}{\epsilon}$, we simply set $\mathcal{X}_{\mathcal{M}_1,\mathcal{M}_2}(\pi) = \emptyset$.

Fixing permutations. In order to formalize the notion of "pulling back" items within a given permutation, as briefly sketched in Section 3.4.1, we define a fixing procedure, $\text{FixCrossing}(\pi, \mathcal{M}^-, \mathcal{M}^+)$. Here, we receive as input a permutation $\pi : Q \to [|Q|]$ over an item set $Q \subseteq \mathcal{N}$, along with two disjoint sets of cluster indices, $\mathcal{M}^-$ and $\mathcal{M}^+$, which are assumed to satisfy $\max \mathcal{M}^- < \min \mathcal{M}^+$, i.e., any index in $\mathcal{M}^-$ is strictly smaller than any index in $\mathcal{M}^+$. As explained below, this procedure constructs in polynomial time a modified permutation $\bar{\pi} : \bar{Q} \to [|\bar{Q}|]$ over a subset $\bar{Q} \subseteq Q$, that satisfies the following properties:

(P_1) *Sparse* $(\mathcal{M}^-, \mathcal{M}^+)$*-crossing:* $\text{cross}_{\mathcal{M}^-,\mathcal{M}^+}(\bar{\pi}) \leq \frac{1}{\epsilon}$.

(P_2) *Completion times:* $C_{\bar{\pi}}(i) \leq C_{\pi}(i)$, for every $i \in \bar{Q}$.

(P_3) *Difference:* $Q \setminus \bar{Q}$ consists of at most one item, which is a member of $\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi)$.

For this purpose, when $\text{cross}_{\mathcal{M}^-,\mathcal{M}^+}(\pi) < \frac{1}{\epsilon}$, the procedure FixCrossing$(\pi, \mathcal{M}^-, \mathcal{M}^+)$ returns exactly the same permutation (i.e., $\bar{\pi} = \pi$), without any alterations. In the opposite case, when $\text{cross}_{\mathcal{M}^-,\mathcal{M}^+}(\pi) \geq \frac{1}{\epsilon}$, let $i_{\mathcal{M}^-,\mathcal{M}^+}$ be the least profitable item in $\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi)$ with respect to the permutation π, namely, $i_{\mathcal{M}^-,\mathcal{M}^+} = \text{argmin}\{\varphi_\pi(i) : i \in \mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi)\}$. Our construction consists of eliminating $i_{\mathcal{M}^-,\mathcal{M}^+}$ and placing instead all items in $C_{\mathcal{M}^-}$ appearing in π after $i_{\mathcal{M}^-,\mathcal{M}^+}$; this alteration results in a permutation $\bar{\pi}$ over $\mathcal{Q} \setminus \{i_{\mathcal{M}^-,\mathcal{M}^+}\}$. Formally, let $\mathcal{A}^-$ and $\bar{\mathcal{A}}^-$ be the items appearing after $i_{\mathcal{M}^-,\mathcal{M}^+}$ out of $C_{\mathcal{M}^-}$ and $\mathcal{N} \setminus C_{\mathcal{M}^-}$, respectively, i.e.,

$$\mathcal{A}^- = \left\{i \in C_{\mathcal{M}^-} : \pi(i) > \pi(i_{\mathcal{M}^-,\mathcal{M}^+})\right\}$$
$$\text{and } \bar{\mathcal{A}}^- = \left\{i \in \mathcal{N} \setminus C_{\mathcal{M}^-} : \pi(i) > \pi(i_{\mathcal{M}^-,\mathcal{M}^+})\right\} .$$

For simplicity, we index the items in $\mathcal{A}^-$ according to their order within the permutation π, which results in having $\mathcal{A}^- = \{i_1, \ldots, i_{|\mathcal{A}^-|}\}$ with $\pi(i_1) < \cdots < \pi(i_{|\mathcal{A}^-|})$. Now, the modified permutation $\bar{\pi}$ is constructed as follows:

- *Before $i_{\mathcal{M}^-,\mathcal{M}^+}$:* Items in positions $1, \ldots, \pi(i_{\mathcal{M}^-,\mathcal{M}^+}) - 1$ of the permutation π remain within their original positions, meaning that $\bar{\pi}(i) = \pi(i)$ for every item i with $\pi(i) \leq \pi(i_{\mathcal{M}^-,\mathcal{M}^+}) - 1$.
- *Instead of $i_{\mathcal{M}^-,\mathcal{M}^+}$:* Items in $\mathcal{A}^-$ will appear in place of $i_{\mathcal{M}^-,\mathcal{M}^+}$ following their relative order in π. That is, $\bar{\pi}(i_k) = \pi(i_{\mathcal{M}^-,\mathcal{M}^+}) - 1 + k$ for every $k \in [|\mathcal{A}^-|]$.
- *After $i_{\mathcal{M}^-,\mathcal{M}^+}$:* Items in $\bar{\mathcal{A}}^-$ will appear after those in $\mathcal{A}^-$, again following their relative order in π. In other words, $\bar{\pi}(i) = \pi(i) - 1 + |\{k \in [|\mathcal{A}^-|] : \pi(i_k) > \pi(i)\}|$ for every item $i \in \bar{\mathcal{A}}^-$.

In Appendix C.3.1, we show that the resulting permutation satisfies its desired properties, as formally stated below.

Lemma 3.4.4. *The permutation $\bar{\pi}$ satisfies properties* (P_1)-(P_3).

The recursive construction. We are now ready to explain how recursive applications of the fixing procedure allow us to conclude the proof of Lemma 3.4.3. At a high level, we bisect the

cluster indices $[M]$, such that in each step the indices being considered are split into their lower half $\mathcal{M}^-$ and upper half $\mathcal{M}^+$, with respect to which the fixing procedure FixCrossing$(\cdot, \mathcal{M}^-, \mathcal{M}^+)$ will be applied. The resulting permutation will then be divided into left and right parts, which are recursively bisected along the same lines.

To present the specifics of this bisection as simply as possible, we assume without loss of generality that the number of clusters M is a power of 2; otherwise, empty clusters can be appended to the sequence $C_1, \ldots, C_M$. At the upper level of the recursion, we bisect the entire collection of cluster indices $[M]$ into $\mathcal{M}_{[1,\frac{M}{2}]} = \{1, \ldots, \frac{M}{2}\}$ and $\mathcal{M}_{[\frac{M}{2}+1,M]} = \{\frac{M}{2}+1, \ldots, M\}$. Designating the optimal permutation by $\pi_{[1,M]} = \pi^*$, we employ our fixing procedure with FixCrossing$(\pi_{[1,M]}, \mathcal{M}_{[1,\frac{M}{2}]}, \mathcal{M}_{[\frac{M}{2}+1,M]})$, to obtain the permutation $\bar{\pi}_{[1,M]}$. Now, we break the latter into its left and right part, $\pi_{[1,\frac{M}{2}]}$ and $\pi_{[\frac{M}{2}+1,M]}$, such that the left permutation $\pi_{[1,\frac{M}{2}]}$ is the prefix of $\bar{\pi}_{[1,M]}$ ending at the last item in $C_{\mathcal{M}_{[1,\frac{M}{2}]}} \cup \mathcal{X}_{\mathcal{M}_{[1,\frac{M}{2}]},\mathcal{M}_{[\frac{M}{2}+1,M]}}(\pi_{[1,M]})$, whereas the right permutation $\pi_{[\frac{M}{2}+1,M]}$ is comprised of the remaining suffix.

In the second level of the recursion, for the left permutation $\pi_{[1,\frac{M}{2}]}$, we bisect $\mathcal{M}_{[1,\frac{M}{2}]}$ into $\mathcal{M}_{[1,\frac{M}{4}]} = \{1, \ldots, \frac{M}{4}\}$ and $\mathcal{M}_{[\frac{M}{4}+1,\frac{M}{2}]} = \{\frac{M}{4}+1, \ldots, \frac{M}{2}\}$, followed by applying FixCrossing$(\pi_{[1,\frac{M}{2}]}, \mathcal{M}_{[1,\frac{M}{4}]}, \mathcal{M}_{[\frac{M}{4}+1,\frac{M}{2}]})$. Similarly, for the right permutation $\pi_{[\frac{M}{2}+1,M]}$, its corresponding set of cluster indices $\mathcal{M}_{[\frac{M}{2}+1,M]}$ is bisected into $\mathcal{M}_{[\frac{M}{2}+1,\frac{3M}{4}]} = \{\frac{M}{2}+1, \ldots, \frac{3M}{4}\}$ and $\mathcal{M}_{[\frac{3M}{4}+1,M]} = \{\frac{3M}{4}+1, \ldots, M\}$, in which case we apply FixCrossing$(\pi_{[\frac{M}{2}+1,M]}, \mathcal{M}_{[\frac{M}{2}+1,\frac{3M}{4}]}, \mathcal{M}_{[\frac{3M}{4}+1,M]})$. This recursive procedure continues up until the resulting sets of cluster indices are singletons. At that point in time, our final permutation π_{sparse} is obtained by concatenating $\pi_{[1,1]}, \pi_{[2,2]}, \ldots, \pi_{[M,M]}$.

Analysis. For ease of presentation, we make use of Ω to denote the set of pairs of cluster index sets with respect to which FixCrossing$(\cdot, \cdot, \cdot)$ is employed throughout our recursive construction,

meaning that

$$\Omega = \Bigg\{ \begin{array}{ll} \left(\mathcal{M}_{[1,\frac{M}{2}]}, \mathcal{M}_{[\frac{M}{2}+1,M]}\right), & \text{[level 1]} \\ \left(\mathcal{M}_{[1,\frac{M}{4}]}, \mathcal{M}_{[\frac{M}{4}+1,\frac{M}{2}]}\right), \left(\mathcal{M}_{[\frac{M}{2}+1,\frac{3M}{4}]}, \mathcal{M}_{[\frac{3M}{4}+1,M]}\right), & \text{[level 2]} \\ \ldots & \\ \left(\mathcal{M}_{[1,1]}, \mathcal{M}_{[2,2]}\right), \ldots, \left(\mathcal{M}_{[M-1,M-1]}, \mathcal{M}_{[M,M]}\right) \Bigg\}. & \text{[level } \log_2 M\text{]} \end{array}$$

With this notation, we show in the next two claims that the permutation π_{sparse} indeed satisfies the sparse crossing and near-optimal profit properties of Lemma 3.4.3.

Lemma 3.4.5. $\text{cross}_m(\pi_{\text{sparse}}) \leq \frac{\log_2 M}{\epsilon}$*, for every* $m \in [M]$.

Proof. By construction of π_{sparse}, every item belonging to one of the clusters $C_{m+1}, \ldots, C_M$ that appears in this permutation before the last item in cluster C_m necessarily resides in $\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$, for some pair $(\mathcal{M}^-, \mathcal{M}^+) \in \Omega$ with $m \in \mathcal{M}^-$. To verify this claim, consider such a crossing item i, say belonging to cluster C_{m^+}. By the way our recursive construction of Ω is defined, there exists a unique pair of cluster index sets $(\mathcal{M}^-, \mathcal{M}^+) \in \Omega$ for which $m \in \mathcal{M}^-$ and $m^+ \in \mathcal{M}^+$; we argue that $i \in \mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$. Indeed, in the next recursion level, the left permutation $\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^-]}$ is the prefix of $\bar{\pi}_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}$ ending with the last item in $C_{\mathcal{M}^-} \cup \mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$. Furthermore, by construction, all items in the right permutation $\pi_{[\min \mathcal{M}^+, \max \mathcal{M}^+]}$ will appear in π_{sparse} after all items in the left permutation $\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^-]}$. Therefore, since $i \in C_{m^+}$ with $m^+ \in \mathcal{M}^+$ and since this item appears in π_{sparse} before the last item in cluster C_m, we know that i appears as part of the left permutation $\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^-]}$, implying that $i \in \mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$.

As any such item $i \in \mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$ contributes at most once toward $\text{cross}_{\mathcal{M}^-,\mathcal{M}^+}(\bar{\pi}_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$, we have

$$\text{cross}_m(\pi_{\text{sparse}}) \leq \sum_{\substack{(\mathcal{M}^-,\mathcal{M}^+) \in \Omega: \\ m \in \mathcal{M}^-}} \text{cross}_{\mathcal{M}^-,\mathcal{M}^+}\left(\bar{\pi}_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}\right)$$

$$\leq \frac{1}{\epsilon} \cdot \left|\{(\mathcal{M}^-, \mathcal{M}^+) \in \Omega : m \in \mathcal{M}^-\}\right|$$
$$\leq \frac{\log_2 M}{\epsilon} .$$

Here, the second inequality holds since $\text{cross}_{\mathcal{M}^-,\mathcal{M}^+}(\bar{\pi}_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}) \leq \frac{1}{\epsilon}$ by property (P_1) of the fixing procedure. The third inequality is obtained by observing that, as the definition of Ω shows, all sets appearing in a single level of the recursion form a partition of $[M]$, implying that $m \in \mathcal{M}^-$ for at most one pair $(\mathcal{M}^-, \mathcal{M}^+)$ in that level. As there are $\log_2 M$ levels overall, it follows that $|\{(\mathcal{M}^-, \mathcal{M}^+) \in \Omega : m \in \mathcal{M}^-\}| \leq \log_2 M$. □

Lemma 3.4.6. $\Psi(\pi_{\text{sparse}}) \geq (1 - \epsilon) \cdot \Psi(\pi^*)$.

Proof. To prove the desired claim, we begin by relating the profits $\Psi(\pi_{\text{sparse}})$ and $\Psi(\pi^*)$, with the corresponding proof in Appendix C.3.2. The main idea is that our fixing procedure eliminates the least profitable item out of $\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$, without increasing the completion time of any other item. Hence, every execution of this procedure loses a profit of at most $\epsilon \cdot \varphi_{\pi^*}(\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}))$.

Claim 3.4.7. $\Psi(\pi_{\text{sparse}}) \geq \Psi(\pi^*) - \epsilon \cdot \sum_{(\mathcal{M}^-,\mathcal{M}^+) \in \Omega} \varphi_{\pi^*}(\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}))$.

The next claim establishes the disjointness of $\mathcal{X}_{\mathcal{M}_1^-,\mathcal{M}_1^+}(\pi_{[\min \mathcal{M}_1^-, \max \mathcal{M}_1^+]})$ and $\mathcal{X}_{\mathcal{M}_2^-,\mathcal{M}_2^+}(\pi_{[\min \mathcal{M}_2^-, \max \mathcal{M}_2^+]})$, for all distinct pairs $(\mathcal{M}_1^-, \mathcal{M}_1^+)$ and $(\mathcal{M}_2^-, \mathcal{M}_2^+)$ in Ω. Informally, this property holds since, once an item appears in some $\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$, it will not appear in any item set belonging to future levels of the recursion, due to the specific way we are pulling back items in the fixing procedure. For ease of presentation, the formal proof is deferred to Appendix C.3.3.

Claim 3.4.8. *For any two distinct pairs* $(\mathcal{M}_1^-, \mathcal{M}_1^+)$ *and* $(\mathcal{M}_2^-, \mathcal{M}_2^+)$ *in* Ω*, the item sets* $\mathcal{X}_{\mathcal{M}_1^-,\mathcal{M}_1^+}(\pi_{[\min \mathcal{M}_1^-, \max \mathcal{M}_1^+]})$ *and* $\mathcal{X}_{\mathcal{M}_2^-,\mathcal{M}_2^+}(\pi_{[\min \mathcal{M}_2^-, \max \mathcal{M}_2^+]})$ *are disjoint.*

Consequently, the profit attained by the permutation π_{sparse} can be bounded by noting that

$$\begin{aligned}\Psi(\pi_{\text{sparse}}) &\geq \Psi(\pi^*) - \epsilon \cdot \sum_{(\mathcal{M}^-,\mathcal{M}^+)\in\Omega} \varphi_{\pi^*}\left(\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}\left(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}\right)\right) \\ &\geq \Psi(\pi^*) - \epsilon \cdot \sum_{i\in\mathcal{N}} \varphi_{\pi^*}(i) \\ &= (1-\epsilon)\cdot\Psi(\pi^*),\end{aligned}$$

where the first inequality is precisely Claim 3.4.7, and the second inequality follows from Claim 3.4.8. □

3.4.4 The external dynamic program

Given the sparse-crossing property of the near-optimal permutation π_{sparse}, whose existence has been established in Lemma 3.4.3, we turn our attention to formally presenting a dynamic programming approach for computing a permutation with a profit of at least $(1-2\epsilon)\cdot\Psi(\pi_{\text{sparse}})$.

States. Building on the intuition provided in Section 3.4.1, we remind the reader that each state $(m, \psi_m, Q_{>m})$ of our dynamic program consists of the following parameters:

- The index of the current cluster m, taking values in $[M]_0$.
- The total profit ψ_m collected thus far. Initially, ψ_m will be treated as a continuous parameter, taking values in $[0, np_{\max}]$, where $p_{\max}$ is the maximum profit attainable by any single item, i.e., $p_{\max} = \max\{p_{it} : i \in [n], t \in [T], \text{ and } w_i \leq W_t\}$.
- The set of items $Q_{>m}$ belonging to clusters $C_{m+1}, \ldots, C_M$ that will be crossing into lower-index clusters. Motivated by the sparse-crossing property established in Lemma 3.4.3, we only consider sets $Q_{>m}$ of cardinality at most $\frac{\lceil \log_2 M \rceil}{\epsilon}$.

Value function. For a subset of items $S \subseteq \mathcal{N}$ and a permutation $\pi : S \to [|S|]$, we say that the pair (S, π) is thin when $\text{cross}_m(\pi) \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$ for all $m \in [M]$. Given this definition, the value

function $F(m, \psi_m, Q_{>m})$ represents the minimum makespan $w(S)$ that can be attained over all thin pairs (S, π) that satisfy the following conditions:

1. *Allowed items*: The set S consists of items that belong to one of the clusters $C_1 \ldots, C_m$ or to $Q_{>m}$. In other words, $S \subseteq C_{[1,m]} \uplus Q_{>m}$, where $C_{[1,m]} = \biguplus_{\mu \in [1,m]} C_\mu$ by convention.

2. *Required crossing items*: The set S contains all items in $Q_{>m}$, meaning that $Q_{>m} \subseteq S$.

3. *Total profit*: $\Psi(\pi) \geq \psi_m$.

Recycling some of the notation introduced in Section 3.2.3.2, we use $\text{Thin}(m, \psi_m, Q_{>m})$ to denote the collection of thin pairs that meet conditions 1-3 above. When the latter set is empty, we define $F(m, \psi_m, Q_{>m}) = \infty$. With these definitions, Lemma 3.4.3 proves the existence of a thin pair $(S, \pi) \in \text{Thin}(M, \Psi(\pi_{\text{sparse}}), \emptyset)$ with $F(M, \Psi(\pi_{\text{sparse}}), \emptyset) \leq W_T$. It is worth pointing out that, for the item set $\mathcal{N}_{\text{sparse}}$ over which the permutation π_{sparse} is defined, we can indeed assume that $w(\mathcal{N}_{\text{sparse}}) \leq W_T$, as all items whose completion time with respect to π_{sparse} exceeds W_T can be eliminated, leaving us with a permutation that still satisfies Lemma 3.4.3. Therefore, had we been able to compute the maximal value ψ^* for which $F(M, \psi^*, \emptyset) \leq W_T$, its corresponding permutation would have guaranteed a profit of at least $\psi^* \geq \Psi(\pi_{\text{sparse}}) \geq (1 - \epsilon) \cdot \Psi(\pi^*)$. Once again, since ψ_m is a continuous parameter, we will eventually explain how to discretize ψ_m to take polynomially-many values, incurring only an ϵ-loss in profit.

Optimal substructure. In what follows, we identify the optimal substructure that allows us to compute the value function F by means of dynamic programming. To this end, suppose that (S, π) is a thin pair that minimizes $w(S)$ over $\text{Thin}(m, \psi_m, Q_{>m})$. We will argue that by eliminating from (S, π) a carefully-selected suffix of the permutation π consisting of items in clusters $C_m, \ldots, C_M$, one obtains a thin pair that attains $F(m-1, \psi_{m-1}, Q_{>m-1})$ for an appropriately defined state $(m-1, \psi_{m-1}, Q_{>m-1})$. We proceed by first defining the latter state, for which a suitable alteration of (S, π) will be shown to be optimal:

- *Crossing set*: $Q_{>m-1}$ is defined as the set of items in $C_m \uplus Q_{>m}$ that appear before the last item in $C_1, \ldots, C_{m-1}$ with respect to the permutation π. Namely,

$$Q_{>m-1} = \left\{ i \in S \cap (C_m \uplus Q_{>m}) : \pi(i) < \max_{j \in S \cap C_{[1,m-1]}} \pi(j) \right\} . \tag{3.8}$$

- *Profit requirement*: $\psi_{m-1} = [\psi_m - \sum_{i \in S \setminus (C_{[1,m-1]} \uplus Q_{>m-1})} \varphi_\pi(i)]^+$.

It is worth pointing out that, for this state to be well-defined, we should ensure that $Q_{>m-1}$ indeed consists of at most $\frac{\lceil \log_2 M \rceil}{\epsilon}$ items. To understand why this property is satisfied, note that since every item in $Q_{>m-1}$ appears in the permutation π before the last item in $S \cap C_{[1,m-1]}$, we have $|Q_{>m-1}| \leq \max_{\mu \in [m-1]} \mathrm{cross}_\mu(\pi) \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$, where the last inequality holds since (S, π) is thin.

Now, let us define the pair $(\hat{S}, \hat{\pi})$, in which $\hat{S} = S \cap (C_{[1,m-1]} \uplus Q_{>m-1})$, meaning that $\hat{S}$ is the restriction of S to items belonging to either one of the clusters $C_1, \ldots, C_{m-1}$ or to $Q_{>m-1}$. It is not difficult to verify that any item in $\hat{S}$ appears in π before any item in $S \setminus \hat{S}$, as any item in $S \cap C_{[m,M]}$ that appears before an item in $C_{[1,m-1]}$ is necessarily a member of $Q_{>m-1}$. Therefore, the items in $\hat{S}$ form a prefix of π, whereas those in $S \setminus \hat{S}$ form the remaining suffix. Given this observation, we define the permutation $\hat{\pi} : \hat{S} \to [|\hat{S}|]$ as the former prefix, or equivalently, as the restriction of π to the items in $\hat{S}$.

In Lemma 3.4.9 below, we show that the pair $(\hat{S}, \hat{\pi})$ indeed resides within $\mathrm{Thin}(m - 1, \psi_{m-1}, Q_{>m-1})$. Subsequently, we prove in Lemma 3.4.10 that this pair is in fact makespan-optimal over the latter set. At a high level, this claim will be established by showing that, for any pair $(\tilde{S}, \tilde{\pi}) \in \mathrm{Thin}(m-1, \psi_{m-1}, Q_{>m-1})$, the item sets $\tilde{S}$ and $S \setminus \hat{S}$ are disjoint. Therefore, had there been such a pair with $w(\tilde{S}) < w(\hat{S})$, it could be extended to a pair in $\mathrm{Thin}(m, \psi_m, Q_{>m})$ via an appropriate addition of $S \setminus \hat{S}$, contradicting the optimality of (S, π). To avoid deviating from the overall flow of this section, the proofs of the next two lemmas are presented in Appendices C.3.4 and C.3.5, respectively.

Lemma 3.4.9. $(\hat{S}, \hat{\pi}) \in \mathrm{Thin}(m - 1, \psi_{m-1}, Q_{>m-1})$.

Lemma 3.4.10. $w(\hat{S}) = F(m - 1, \psi_{m-1}, Q_{>m-1})$.

Recursive equations. Given the optimal substructure characterization discussed above, we proceed by explaining how to express $F(m, \psi_m, Q_{>m})$ in recursive form. In essence, had we known what the preceding state $(m-1, \psi_{m-1}, Q_{>m-1})$ is, the remaining question would have been that of identifying the lightest set of "extra" items $\mathcal{E}$ to be appended, along with their internal permutation $\pi_{\mathcal{E}} : \mathcal{E} \to [|\mathcal{E}|]$, under a marginal profit constraint. Formally, to capture the agreement between crossing items, we say that state $(m-1, \psi_{m-1}, Q_{>m-1})$ is conceivable for state $(m, \psi_m, Q_{>m})$ when $Q_{>m-1} \setminus C_m \subseteq Q_{>m}$. In the opposite direction, $(m, \psi_m, Q_{>m})$ is reachable from $(m-1, \psi_{m-1}, Q_{>m-1})$ when there exist an item set $\mathcal{E}$ and permutation $\pi_{\mathcal{E}} : \mathcal{E} \to [|\mathcal{E}|]$ that simultaneously satisfy the following constraints:

1. *Extra items*: The collection of extra items can be written as $\mathcal{E} = \mathcal{E}_m \uplus (Q_{>m} \setminus Q_{>m-1})$. Here, items in $\mathcal{E}_m$ are to be picked out of cluster C_m, with the exclusion of those appearing in $Q_{>m-1}$, meaning that we have the constraint $\mathcal{E}_m \subseteq C_m \setminus Q_{>m-1}$. Concurrently, each and every item in $Q_{>m} \setminus Q_{>m-1}$ should be picked.

2. *Marginal profit*: $\sum_{i \in \mathcal{E}} \varphi^{\rightsquigarrow}_{\pi_{\mathcal{E}}}(i) \geq \psi_m - \psi_{m-1}$, where the term $\varphi^{\rightsquigarrow}_{\pi_{\mathcal{E}}}(i)$ denotes the profit of item i with respect to the permutation $\pi_{\mathcal{E}}$, when its completion time is increased by $F(m-1, \psi_{m-1}, Q_{>m-1})$. This constraint guarantees that, by appending $\pi_{\mathcal{E}}$ to the permutation that achieves $F(m-1, \psi_{m-1}, Q_{>m-1})$, we obtain a total profit of at least ψ_m.

Letting $\text{Extra}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$ denote the collection of item sets and permutations that satisfy these constraints, we mention in passing that this set may be empty. Moreover, it will be utilized only for purposes of analysis, and in particular, we will not assume that $\text{Extra}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$ can be efficiently constructed. Nevertheless, the function value $F(m, \psi_m, Q_{>m})$ can still be expressed by minimizing $F(m-1, \psi_{m-1}, Q_{>m-1}) + w(\mathcal{E})$ over all conceivable states $(m-1, \psi_{m-1}, Q_{>m-1})$ and over all item sets and permutations $(\mathcal{E}, \pi_{\mathcal{E}}) \in \text{Extra}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$. For convenience, when $F(m, \psi_m, Q_{>m}) \leq W_T$, we use $\text{Best}(m, \psi_m, Q_{>m})$ to denote an arbitrary state $(m-1, \psi_{m-1}, Q_{>m-1})$ chosen out of those for which the minimum value $F(m, \psi_m, Q_{>m})$ is attained. As mentioned earlier, we wish to compute the maximal value ψ^* that satisfies $F(M, \psi^*, \emptyset) \leq W_T$, as its corresponding

permutation guarantees a profit of at least $(1-\epsilon)\cdot\Psi(\pi^*)$.

Approximate recursion. That said, due to having a lower bound on the marginal profit, even when $\text{Best}(m,\psi_m,Q_{>m})$ is known, the recursive formulation above is expected to identify an item set and permutation $(\mathcal{E},\pi_{\mathcal{E}})\in\text{Extra}[\,{}^{(m,\psi_m,Q_{>m})}_{\text{Best}(m,\psi_m,Q_{>m})}\,]$ for which $w(\mathcal{E})$ is minimized. This setting can be viewed as an "inverse" generalized incremental knapsack problem, where the objective is to minimize makespan rather than to maximize profit. To deal with this obstacle, we employ our QPTAS for bounded weight ratio instances (see Section 3.3) in order to approximately solve these recursive equations.

Specifically, for $\Delta\geq 0$, we say that constraint 2 is (ϵ,Δ)-satisfied when $\sum_{i\in\mathcal{E}}\varphi^{+\Delta}_{\pi_{\mathcal{E}}}(i)\geq(1-\epsilon)\cdot(\psi_m-\psi_{m-1})$, where $\varphi^{+\Delta}_{\pi_{\mathcal{E}}}(i)$ is the profit of item i with respect to the permutation $\pi_{\mathcal{E}}$, when its completion time is increased by Δ. As such, the standard sense of satisfying this constraint can be recovered by picking $\epsilon=0$ and $\Delta=F(m-1,\psi_{m-1},Q_{>m-1})$. With this definition, we say that state $(m,\psi_m,Q_{>m})$ is (ϵ,Δ)-reachable from state $(m-1,\psi_{m-1},Q_{>m-1})$ when there exist an item set $\mathcal{E}$ and permutation $\pi_{\mathcal{E}}:\mathcal{E}\to[|\mathcal{E}|]$ that satisfy constraint 1 and (ϵ,Δ)-satisfy constraint 2; as before, $\text{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$ will stand for the collection of such item sets and permutations. In what follows, we devise an auxiliary procedure for approximately solving the recursive equations, as summarized in the next claim; for readability purposes, the proof is deferred to Appendix C.3.6.

Lemma 3.4.11. *Suppose that $(m,\psi_m,Q_{>m})$ and $(m-1,\psi_{m-1},Q_{>m-1})$ are two given states, such that $F(m,\psi_m,Q_{>m})\leq W_T$ and $(m-1,\psi_{m-1},Q_{>m-1})=\text{Best}(m,\psi_m,Q_{>m})$. Given a parameter $\Delta\leq F(m-1,\psi_{m-1},Q_{>m-1})$, we can identify an item set $\hat{\mathcal{E}}$ and permutation $\hat{\pi}_{\hat{\mathcal{E}}}:\hat{\mathcal{E}}\to[|\hat{\mathcal{E}}|]$ for which:*

1. $(\hat{\mathcal{E}},\hat{\pi}_{\hat{\mathcal{E}}})\in\text{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$.
2. $w(\hat{\mathcal{E}})\leq F(m,\psi_m,Q_{>m})-F(m-1,\psi_{m-1},Q_{>m-1})$.

The running time of our algorithm is $O((nT)^{O(\frac{1}{\epsilon^6}\cdot(\log n+\log M))}\cdot|\mathcal{I}|^{O(1)})$, regardless of whether the assumptions above hold or not.

With this procedure in-hand, we define an approximate value function $\hat{F}$, whose state space is identical to that of F. However, rather than attempting to solve an inverse generalized incremental knapsack problem, the recursive equations through which $\hat{F}$ is defined will tackle the latter problem in an approximate way via our auxiliary procedure. To formalize this approach, the function value $\hat{F}(m, \psi_m, Q_{>m})$ is evaluated as follows:

- *Terminal states ($m = 0$)*: Here, we simply define $\hat{F}(0, \psi_0, Q_{>0}) = F(0, \psi_0, Q_{>0})$. While F-values are unknown in general, $F(0, \psi_0, Q_{>0})$ evaluates to either $w(Q_{>0})$, when there exists a permutation $\pi_{Q_{>0}} : Q_{>0} \to [|Q_{>0}|]$ with profit $\Psi(\pi_{Q_{>0}}) \geq \psi_0$, or to ∞ otherwise. This distinction can be made by enumerating over all permutations of $Q_{>0}$ in time $O((\frac{1}{\epsilon}\log M)^{O(\frac{1}{\epsilon}\log M)}) = O(|\mathcal{I}|^{O((\frac{1}{\epsilon}\log|\mathcal{I}|)^{O(1)})})$, since $|Q_{>0}| \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$.

- *General states ($m \in [M]$)*: For each state $(m-1, \psi_{m-1}, Q_{>m-1})$, we instantiate Lemma 3.4.11 with $\Delta = \hat{F}(m-1, \psi_{m-1}, Q_{>m-1})$, to obtain the item set $\hat{\mathcal{E}}$ and its permutation $\hat{\pi}_{\hat{\mathcal{E}}} : \hat{\mathcal{E}} \to [|\hat{\mathcal{E}}|]$. The value $\hat{F}(m, \psi_m, Q_{>m})$ is determined by minimizing $\hat{F}(m-1, \psi_{m-1}, Q_{>m-1}) + w(\hat{\mathcal{E}})$ over all conceivable states $(m-1, \psi_{m-1}, Q_{>m-1})$ for which $(\hat{\mathcal{E}}, \hat{\pi}_{\hat{\mathcal{E}}}) \in \mathrm{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$, noting that the latter condition can easily be tested.

It is important to emphasize that, when employing our auxiliary procedure above, we have no way of knowing a-priori whether the assumptions made in Lemma 3.4.11 hold or not. Nevertheless, as we show in the next lemma, whose proof is provided in Appendix C.3.8, any profit requirement which is attainable by the original dynamic program F can be attained up to factor $1-\epsilon$ by our approximate program $\hat{F}$. The precise relationship we establish between these functions can be formally stated as follows.

Lemma 3.4.12. *Let $(m, \psi_m, Q_{>m})$ be a state for which $F(m, \psi_m, Q_{>m}) \leq W_T$. Then, $\hat{F}(m, \psi_m, Q_{>m}) \leq F(m, \psi_m, Q_{>m})$, where the makespan $\hat{F}(m, \psi_m, Q_{>m})$ is attained by an item set $\hat{S}_m$ and a permutation $\hat{\pi}_{\hat{S}_m} : \hat{S}_m \to [|\hat{S}_m|]$ for which:*

- *Allowed and required items: $\hat{S}_m \subseteq C_{[1,m]} \uplus Q_{>m}$ and $Q_{>m} \subseteq \hat{S}_m$.*

- *Profit:* $\Psi(\hat{\pi}_{\hat{S}_m}) \geq (1-\epsilon) \cdot \psi_m$.

As previously mentioned, the primary intent of this section is to compute a permutation with a profit of at least $(1-2\epsilon) \cdot \Psi(\pi_{\text{sparse}})$. To argue that we have nearly achieved this objective, recall that Lemma 3.4.3 proves the existence of a thin pair $(S, \pi) \in \text{Thin}(M, \Psi(\pi_{\text{sparse}}), \emptyset)$ with $F(M, \Psi(\pi_{\text{sparse}}), \emptyset) \leq W_T$. Therefore, as an immediate consequence of Lemma 3.4.12, we infer that $\hat{F}(M, \Psi(\pi_{\text{sparse}}), \emptyset) \leq W_T$, which is attained by a permutation π with a profit of $\Psi(\pi) \geq (1-\epsilon) \cdot \Psi(\pi_{\text{sparse}})$.

The discrete program $\tilde{F}$. That said, the above-mentioned existence proof still does not correspond to a constructive algorithm, due to the continuity of the profit requirement parameter ψ_m. To discretize this parameter, similarly to Section 3.2.3.3, we restrict ψ_m to a finite set of values, $\mathcal{D}_\psi = \{d \cdot \frac{\epsilon p_{\max}}{2n} : d \in [\frac{2n^2}{\epsilon}]_0\}$. In turn, we use $\tilde{F}(m, \psi_m, \mathcal{Q}_{>m})$ to denote the resulting dynamic program over the discretized set of states, whose recursive equations are identical to those of $\hat{F}$, except for instantiating Lemma 3.4.11 with $\Delta = \tilde{F}(m-1, \psi_{m-1}, \mathcal{Q}_{>m-1})$.

We conclude our analysis by lower-bounding the best-possible profit achievable through this dynamic program, showing that it indeed matches that of the permutation π_{sparse} up to ϵ-related terms. To avoid redundancy, we omit the corresponding proof, as it is nearly identical to that of Lemma 3.2.6.

Lemma 3.4.13. *There exists a value $\tilde{\psi} \in \mathcal{D}_\psi$ such that $\tilde{\psi} \geq (1-\epsilon) \cdot \Psi(\pi_{\text{sparse}})$ and such that $\tilde{F}(M, \tilde{\psi}, \emptyset) \leq W_T$. This makespan is attained by an item set $\tilde{S}$ and a permutation $\tilde{\pi}_{\tilde{S}}$ whose profit is $\Psi(\tilde{\pi}_{\tilde{S}}) \geq (1-\epsilon) \cdot \tilde{\psi} \geq (1-2\epsilon) \cdot \Psi(\pi_{\text{sparse}})$.*

Running time. We first observe that the function $\tilde{F}(m, \psi_m, \mathcal{Q}_{>m})$ is being evaluated over $O(n^{O(\frac{1}{\epsilon}\log M)} \cdot |\mathcal{I}|^{O(1)})$ possible states. To verify this claim, note that there are $O(M) = O(|\mathcal{I}|)$ choices for the cluster index m, and that the discretized profit parameter ψ_m takes values in $\mathcal{D}_\psi$, with $|\mathcal{D}_\psi| = O(\frac{n^2}{\epsilon})$. In addition, the set of crossing items $\mathcal{Q}_{>m}$ is of cardinality at most $\frac{\lceil \log_2 M \rceil}{\epsilon}$, implying that there are only $O(n^{O(\frac{1}{\epsilon}\log M)})$ subsets to consider for this parameter. Now, evaluating

$\tilde{F}(m, \psi_m, Q_{>m})$ for a given state depends on its type:

- *Terminal states ($m = 0$)*: As previously explained, such states are handled by enumerating over all permutations of $Q_{>0}$ in time $O(|\mathcal{I}|^{O((\frac{1}{\epsilon}\log|\mathcal{I}|)^{O(1)})})$.
- *General states* ($m \in [M]$): Here, each state $(m-1, \psi_{m-1}, Q_{>m-1})$ would involve a single application of our auxiliary procedure, running in $O((nT)^{O(\frac{1}{\epsilon^6}\cdot(\log n+\log M))} \cdot |\mathcal{I}|^{O(1)})$ according to Lemma 3.4.11. As argued above, there are only $O(n^{O(\frac{1}{\epsilon}\log M)} \cdot |\mathcal{I}|^{O(1)})$ states of the form $(m-1, \psi_{m-1}, Q_{>m-1})$ to be considered.

Overall, we incur a running time of $O(|\mathcal{I}|^{O((\frac{1}{\epsilon}\log|\mathcal{I}|)^{O(1)})})$, as stated in Theorem 3.4.1.

3.5 Experimental results

In this section, we give computational results that compare the performance of the $(\frac{1}{2}-\epsilon)$-approximated algorithm presented in Section 3.2 with the performance of a LP rounding procedure as well as with the integer programming solver Gurobi. In Section 3.5.1, we discuss in detail each of the algorithms tested. In Section 3.5.2, we discuss the generation of the instances and the settings of the experiments. Finally, in Section 3.5.3, we compare the results of the algorithms and discuss their implications.

3.5.1 Algorithms tested

LP rounding approach. In the LP rounding approach, we give the solver the linear relaxation of (GIK-IP) given in Section 1.2.1 where the binary constraints for each variable $x_{i,t}$ are replaced with $0 \le x_{i,t} \le 1$. The linear relaxation is solved to optimality as to obtain $\bar{x}$. Since $\bar{x}$ may contain fractional coordinates, it may not be feasible for the generalized incremental knapsack instance. To restore feasibility, we return the rounded integral solution $\lfloor \bar{x} \rfloor$ as our final solution. It is easy to see that $\lfloor \bar{x} \rfloor$ is feasible for (GIK-IP).

$(\frac{1}{2}-\epsilon)$-approximation. We implement the $(\frac{1}{2}-\epsilon)$-approximated algorithm from Section 3.2 for $\epsilon = \frac{1}{3}$. Recall that the algorithm relies on reformulating the problem as a sequencing problem on

a single machine. Following the reformulation, the profit function is decomposed into heavy and light items contributions. We then proposed two approximation schemes, one for heavy contributions and one for light contributions. We separately present the computational results for each of these two approximation schemes.

3.5.2 Instance generation and experimental setup

Table 3.1 (resp. Table 3.2) tests the same instances as Table 2.1 (resp. Table 2.2). The instance generation details can be found in Section 2.5.2. Thus, the performance between the c-flexible algorithms from Section 2.3, the $(\frac{1}{2} - \epsilon)$-approximated algorithm, and the LP rounding approach are directly comparable across these tables. For the $(\frac{1}{2} - \epsilon)$-approximation, the heavy algorithm relies on dynamic programming ideas which is polynomial in theory, but requires a state space that is too large to be solved efficiently in practice, even for relatively small problem sizes. Hence, we only report the number of states required for the dynamic program.

As is the case from Section 2.5.2, for all problem sizes except $n = T = 3000$, for each of the algorithm tested, the "Mean difference" columns report the mean difference with respect to the solution that Gurobi outputs. For $n = T = 3000$, Gurobi and LP rounding runs out of memory and are not able to return feasible solutions. Hence, the "Mean difference" columns for the light algorithm in Tables 3.1 and 3.2 report the mean difference with respect to the fully-flexible algorithm from Tables 2.1 and 2.2 respectively.

3.5.3 Results and discussion

Similar to the running time of the fully flexible and 2-flexible algorithms tested in Tables 2.1 and 2.2, the light algorithm and the LP rounding approach generally takes slightly longer to solve for problems with correlated weights and profits compared to problems with random and uncorrelated weights and profits. We remark that the simple LP rounding approach typically takes longer to solve compared to the c-flexible algorithms from Tables 2.1 and 2.2, even though the c-flexible algorithms require the solutions to T knapsack problems.

Problem Size		Gurobi Results		LP Rounding		$(\frac{1}{2}-\epsilon)$-approximation Light		Heavy
n	T	Optimality Gap	Time (Sec)	Mean difference	Time (Sec)	Mean difference	Time (Sec)	States
50	50	1%	175	49.7%	0.1	20.5%	0.2	3.8×10^{22}
100	100	5% 3% 2%	4.3 210 1534	35.4% 35.9% 36.0%	1.1	14.6% 15.4% 15.5%	1.1	1.9×10^{38}
500	500	5% 3% 2%	98 100 125	15.0% 15.1% 15.6%	69	9.6% 9.7% 10.3%	2.7	1.5×10^{160}
3000	3000	100%	-	100%	-	7.8%	63.9	2.4×10^{914}

Table 3.1: Correlated weights and profits, Gurobi and $(\frac{1}{2}-\epsilon)$-approximated algorithm for $\epsilon = \frac{1}{3}$. The mean difference of the light algorithm is with respect to the solution obtained by the fully flexible algorithm from Table 2.1.

Problem Size		Gurobi Results		LP Rounding		$(\frac{1}{2}-\epsilon)$-approximation Light		Heavy
n	T	Optimality Gap	Time (Sec)	Mean difference	Time (Sec)	Mean difference	Time (Sec)	States
50	50	1%	0.6	5.9%	0.1	68.3%	0.3	3.6×10^{22}
100	100	1%	2.4	5.2%	0.3	65.5%	0.4	1.9×10^{38}
500	500	1%	25.3	2.1%	30.6	57.9%	2.6	1.5×10^{160}
3000	3000	100%	-	100%	-	18.8%	84.9	2.4×10^{914}

Table 3.2: Random and uncorrelated weights and profits, Gurobi and $(\frac{1}{2}-\epsilon)$-approximated algorithm for $\epsilon = \frac{1}{3}$. The mean difference of the light algorithm is with respect to the solution obtained by the fully flexible algorithm from Table 2.2.

For the LP rounding based algorithm, since it requires only the solution of a linear program, it is quite efficient for relative small problem sizes. However, without a theoretical approximation guarantee, it can perform relatively poorly, especially for instances of small problem sizes on correlated instances (See Table 3.1). For the problem size of $n = T = 3000$, even solving a linear program proves to be too memory-intensive. The light items algorithm runs efficiently, even on large instances. However, while the better of the heavy and light algorithm achieves an approximation factor of $\frac{1}{2}-\epsilon$, each algorithm by itself does not have any theoretical approximation guarantee. Hence, the light algorithm solutions are generally considerably worse than those given by Gurobi and the c-flexible algorithm, especially for random and uncorrelated weights and profits

(See Tables 2.2 and 3.2).

We remark that the algorithms presented in Sections 3.3 and 3.4 were not implemented for comparison, as even one round of the algorithm given in Section 3.3 is quasi-polynomial in running time and more expensive than the implementation of the heavy algorithm.

Chapter 4: Some easier, and some not harder, incremental knapsack problems

4.1 Introduction

In this chapter, we investigate variations to the incremental knapsack problems. Our first and most interesting result of this chapter deals with the incremental knapsack problem whose objective function is defined through a monotone submodular all-or-nothing function (IK−AoN). Such functions are a common generalization of linear functions and rank functions of matroids. Somewhat surprisingly, we show that IK−AoN can be reduced to the linear case, and hence, using results from the literature, has a PTAS [4]. This is shown in Section 4.2.

In Section 4.3, we give a PTAS for the generalized incremental knapsack problem when T is bounded. This approach relies on the fact that there always exists an optimal solution to the linear relaxation of (GIK-IP) with the number of fractional components depending only on T (in particular, independent of n). Hence, standard guessing and rounding techniques can be employed when the number of times is assumed to be bounded.

In Section 4.4, we study the generalized incremental problem when, for every item $i \in [n]$, there is a unique time t such that $p_{i,t} > 0$. For all other times $\tau \neq t$, we assume the profit is $p_{i,\tau} = 0$. In such a case, for each item, there is a unique time in which it should be inserted into the knapsack, or it will not be inserted into the knapsack at all. With this insight, we modify the dynamic programming that gives an FPTAS for the classical knapsack problem to also give an FPTAS for this special case.

4.2 Algorithm for the monotone submodular all-or-nothing incremental knapsack problem

In this section, we give an algorithm that proves Theorem 1.3.4, reprinted below for convenience. Recall from Section 1.4.1 that IK is the linear variant to IK−AoN. Namely, the two problems have identical feasibility conditions, and the same input parameters p_i for every $i \in [n]$ and Δ_t for every $t \in [T]$. For every item i in the knapsack at time t, it earns the full profit of $\Delta_t p_i$. The profit of any chain $\mathcal{S}$ in an IK instance is therefore given by $\Phi(\mathcal{S}) = \sum_{t\in[T]} \Delta_t \sum_{i\in S_t} p_i$. For a family C of instances of IK−AoN, we call a family $\overline{C}$ of IK instances the linearization of C if $\overline{C}$ consists of $\mathcal{I}'$ if and only if $\mathcal{I}'$ is defined as follows: starting from an instance $\mathcal{I} \in C$ with the ground set of items $[n]$ with profits p, after dropping some items (and possibly after renaming), we have $\mathcal{I}'$ on the ground set of item $[n'] \subseteq [n]$ and a new (linear) function γ' such that for every $S \subseteq [n']$, we have the linear profit $\gamma'(S) = p(S)$.

Theorem 1.3.4. *Let C be a family of IK−AoN instances and let $\alpha \in [0, 1]$. If there is an α-approximation algorithm for instances in $\overline{C}$, then there is an α-approximation algorithm for instances in C running in time $O(\mathrm{Time}_\alpha(n, T) + nT)$, where $\mathrm{Time}_\alpha(n, T)$ is the running time of the α-approximation algorithm for instances in $\overline{C}$ with n items and T time periods.*

In Section 4.2.1, we give the algorithm that, given an instance of IK−AoN, creates a linearized instance of the problem. We then give a high-level overview of how the algorithm leads to the proof of Theorem 1.3.4. In Section 4.2.2, we show some properties of the monotone submodular all-or-nothing function γ and give conditions for when a set has linear profits for this function. In Section 4.2.3, we show further properties of γ when we assume every item in the ground set has the same profit. This leads us to a decomposition theorem in Section 4.2.4 that may be of independent interest. Specifically, we show that monotone submodular all-or-nothing functions can be characterized by the sum of scaled ranked functions of matroids. Finally, in Section 4.2.5, we tie the properties together to give a proof of Theorem 1.3.4.

4.2.1 The linearization algorithm

In this section, we first present the algorithm that achieves Theorem 1.3.4. We then give a high-level overview before proceeding with the proofs in subsequent sections.

Preliminaries. We mentioned in Section 1.4.1 that the IK−AoN is a generalization of the incremental knapsack problem and the matroid rank function maximization. In the following example, we discuss the matroid rank function maximization in more detail.

Example 4.2.1 (Matroid rank profits)**.** *Let $\Delta_t = 1$ for every $t \in [T]$ and γ be the rank function of a matroid. Hence, our goal is to find sequence of sets $S_1 \subseteq \cdots \subseteq S_T$ so that $w(S_t) \leq W_t$ for every $t \in [T]$ and the sum of their ranks is maximized. By a perturbation of the classical proof of the optimality of the greedy algorithm to find an independent set of maximum weight, see, for example [18, Chapter 8], or by Lemma 4.2.10 proved later in this chapter, one deduces that the optimal solution can be obtained with a greedy procedure. In particular: Sort the items $[n] = \{1, 2, \ldots, n\}$ such that $w_1 \leq \cdots \leq w_n$. For $t = 1, \ldots, T$, first set $S_t = S_{t-1}$ (with $S_0 = \emptyset$). Then, for $i \in [n]$, let $S_t = S_t \cup \{i\}$ if $w(S_t \cup \{i\}) \leq W_t$ and $S_t \cup \{i\}$ is independent.*

Motivated by the rank function of r of a matroid, where $r(S) = |S|$ if and only if S is independent, we call a set $S \subseteq [n]$ *independent* if $\gamma(S) = \sum_{i \in S} p_i$, *dependent* otherwise.

The algorithm.

1. Let $\mathcal{I}$ be an instance of on items $[n]$ with profits p and profit function γ.

2. Partition $[n]$ into $(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ so that, for $\ell \in [k]$, all items in $\mathcal{P}_\ell$ have profit $p^{(\ell)}$, and $0 < p^{(1)} < p^{(2)} < \cdots < p^{(k)}$.

3. For each $\ell \in [k]$, we create independent sets $\mathcal{P}_\ell^I \subseteq \mathcal{P}_\ell$ as follows:

   (a) Let $\mathcal{P}_\ell = \{e_1^\ell, \ldots, e_{m^\ell}^\ell\}$ such that $w_{e_1^\ell} \leq \cdots \leq w_{e_{m^\ell}^\ell}$.

   (b) Set $\mathcal{P}_\ell^I = \emptyset$.

(c) For each $j \in [m^\ell]$,

- if $\mathcal{P}_\ell^I \cup \{e_j^\ell\}$ is independent, set $\mathcal{P}_\ell^I = \mathcal{P}_\ell^I \cup \{e_j^\ell\}$.

4. Define an incremental knapsack instance $\mathcal{I}'$ with items $\cup_{\ell \in [k]} \mathcal{P}_\ell^I$, same weights and capacities as $\mathcal{I}$, and $p'_i = p_i$ for all $i \in \cup_{\ell \in [k]} \mathcal{P}_\ell^I$.

5. Apply the α-approximation algorithm to the incremental knapsack instance $\mathcal{I}'$ as to obtain feasible chain $\overline{\mathcal{S}}$.

6. Output $\overline{\mathcal{S}}$.

A high-level view of the proof of Theorem 1.3.4. We saw that the linear incremental knapsack case and the matroid case are special settings of IK−AoN. Our approach shows conversely that combining tools from these two settings suffices to solve IK−AoN.

First, we can assume that all sets of the form $\{i\}$ for $i \in [n]$ are independent – else, by submodularity, $\gamma(S) = \gamma(S \setminus \{i\})$ for all $S \subseteq [n]$, and we can consider the problem restricted to $[n] \setminus \{i\}$. In our setting, independent sets share with the matroid setting classical properties, e.g., independence is preserved under taking subsets, see Lemma 4.2.3.

We call a chain $\mathcal{S} = (S_1, \ldots, S_T)$ *independent* if $S_1, \ldots, S_T$ are independent (equivalently, if S_T is independent). Observe that, in Example 4.2.1, there is always an optimal chain that is independent. The same happens if the instance is an incremental knapsack problem, since all chains are trivially independent. As our first step, we show in Lemma 4.2.6 that, also in the IK−AoN setting, we can restrict to consider independent chains only, because one such chain is optimal.

How do independent chains look like? As our second step, we show that dependency can only be created among sets with the same profit. More formally, let $(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ be the partition of $[n]$ given by the algorithm. That is, for $\ell \in [k]$, all items in the *profit class* $\mathcal{P}_\ell$ have profit $p^{(\ell)}$, and $0 < p^{(1)} < p^{(2)} < \cdots < p^{(k)}$. If for $\ell \in [k]$, $S^\ell \subseteq \mathcal{P}_\ell$ is independent, $\cup_{\ell \in [k]} S^\ell$ is also independent. See Lemma 4.2.7 and Lemma 4.2.8.

Hence, we can "slice" the ground set $[n]$ by profit, and focus on understanding independent sets contained in each given profit class $\mathcal{P}_\ell$ separately. Lemma 4.2.9 implies that the classical greedy algorithm for matroids can be employed to find an independent set $\mathcal{P}_\ell^I$ of $\mathcal{P}_\ell$ of minimum weight. This leads to a characterization of monotone submodular all-or-nothing function as the sum of scaled ranks of matroids: see Theorem 4.2.11.

As we show in Lemma 4.2.10, we can assume that the restriction to $\mathcal{P}_\ell$ of any optimal independent chain is, without loss of generality, contained in $\mathcal{P}_\ell^I$. So we can restrict to consider independent sets contained in $\mathcal{P}_\ell^I$. By monotonicity, all subsets of $\mathcal{P}_\ell^I$ are independent. So we can just consider the linearized problem where we remove from each profit class items in $\mathcal{P}_\ell \setminus \mathcal{P}_\ell^I$. On this problem, we can apply the α-approximation algorithm whose existence is guaranteed by the hypothesis of Theorem 1.3.4, and the thesis follows.

4.2.2 Independent sets

To study independent sets, we first introduce some relevant concepts and properties, mostly extending the analogous ones for matroids.

Dependent sets, cycles and monotonicity. We call a set $C \subseteq [n]$ a *cycle* if C is dependent and for every $i \in C$, $C \setminus \{i\}$ is independent.

The next lemma guarantees that every dependent set contains a cycle. The proof can be found in Appendix D.1.1.

Lemma 4.2.2. *Let $S \subseteq [n]$ be dependent. Then there exists $C \subseteq S$ such that C is a cycle.*

We show in the next lemma that the property of being independent is monotone with respect to set inclusion. Its proof is provided in Appendix D.1.2.

Lemma 4.2.3. *Let $S' \subseteq S \subseteq [n]$. If S is independent, then S' is independent.*

The next lemma shows that for any set S, there exists an independent subset of S with equal profit. Its proof can be found in Appendix D.1.3.

Lemma 4.2.4. *For any set $S \subseteq [n]$, there exists an independent set $S' \subseteq S$ such that $\gamma(S) = \gamma(S') = \sum_{i \in S'} p_i$.*

Restriction to independent chains. Recall that we call a chain $\mathcal{S} = (S_1, \ldots, S_T)$ *independent* if $S_1, \ldots, S_T$ are independent (using Lemma 4.2.3, this is equivalent to S_T being independent). In the case of incremental knapsack problems where the objective function is linear, all chains are trivially independent. It is immediate to see that the converse is also true.

Observation 4.2.5. *Let $\mathcal{I}$ be an* `IK-AoN` *instance. All chains of $\mathcal{I}$ are independent if and only if $\mathcal{I}$ is an instance of* `IK`*.*

As we show next, in `IK-AoN`, we can also restrict our attention to independent chains.

Lemma 4.2.6. *Every* `IK-AoN` *instance admits an optimal chain that is independent.*

Proof. Let $\mathcal{S}^* = (S_1^*, \ldots, S_T^*)$ denote an optimal chain. For every $i \in S_T^*$, let $t(i)$ be the insertion time of item i with respect to $\mathcal{S}^*$. We assume without loss of generality that

$$p_i = \gamma(S_{t(i)}^*) - \gamma(S_{t(i)}^* \setminus \{i\}). \tag{4.1}$$

Else, we claim that the chain $\overline{\mathcal{S}} = (\overline{S}_1, \overline{S}_2, \ldots, \overline{S}_T)$ with $\overline{S}_t = S_t^* \setminus \{i\}$ for $t \in [T]$ is also optimal, and we can iteratively remove items that do not satisfy (4.1). Suppose in fact $\gamma(S_{t(i)}^*) - \gamma(S_{t(i)}^* \setminus \{i\}) = 0$. Then

$$0 = \gamma(S_{t(i)}^*) - \gamma(S_{t(i)}^* \setminus \{i\}) \geq \gamma(S_t^*) - \gamma(S_t^* \setminus \{i\})$$

for every $t \geq t(i)$ by submodularity and the definition of a chain. Hence, $\gamma(S_t^*) = \gamma(S_t^* \setminus \{i\})$ for all $t \geq t(i)$. Since for $t \in [t(i) - 1]$ we have $\overline{S}_t = S_t^*$, the claim follows. We therefore assume that (4.1) holds for all $i \in S_T^*$.

By way of contradiction, suppose $\mathcal{S}^*$ is not independent. Thus, there exists some $t \in [T]$ and $i \in S_t^*$ such that $\gamma(S_t^* \setminus \{i\}) = \gamma(S_t^*)$. Let $\tau(i)$ be the smallest time $t \in [T]$ where $0 =$

$\gamma(S_t^*) - \gamma(S_t^* \setminus \{i\})$. We know that $\tau(i) > t(i) \geq 1$. Furthermore, $S^*_{\tau(i)} \setminus S^*_{\tau(i)-1} \neq \emptyset$, else

$$\gamma(S^*_{\tau(i)}) - \gamma(S^*_{\tau(i)} \setminus \{i\}) = \gamma(S^*_{\tau(i)-1}) - \gamma(S^*_{\tau(i)-1} \setminus \{i\}) \neq 0,$$

contradicting the choice of $\tau(i)$. Let $\{j_1, \ldots, j_k\} = S^*_{\tau(i)} \setminus S^*_{\tau(i)-1}$. For $\ell \in [k]$, since the insertion time of j_ℓ is $\tau(i)$, we know that $p_{j_\ell} = \gamma(S^*_{\tau(i)}) - \gamma(S^*_{\tau(i)} \setminus \{j_\ell\})$. Furthermore,

$$p_{j_\ell} = \gamma(S^*_{\tau(i)-1} \cup \{j_1, \ldots, j_\ell\}) - \gamma(S^*_{\tau(i)-1} \cup \{j_1, \ldots j_{\ell-1}\}). \tag{4.2}$$

by submodularity since $S^*_{\tau(i)-1} \cup \{j_1, \ldots, j_{\ell-1}\} \subseteq S^*_{\tau(i)} \setminus \{j_\ell\}$. By a similar reasoning,

$$p_{j_\ell} = \gamma(S^*_{\tau(i)-1} \setminus \{i\} \cup \{j_1, \ldots, j_\ell\}) - \gamma(S^*_{\tau(i)-1} \setminus \{i\} \cup \{j_1, \ldots j_{\ell-1}\}). \tag{4.3}$$

We have

$$\sum_{\ell=1}^{k} p_{j_\ell} = \gamma(S^*_{\tau(i)-1} \cup \{j_1, \ldots, j_k\}) - \gamma(S^*_{\tau(i)-1}) = \gamma(S^*_{\tau(i)}) - \gamma(S^*_{\tau(i)-1}),$$

where the first equality follows by summing together (4.2) for all $\ell \in [k]$, and then by telescoping cancellations, and the second equality follows from definition.

Similarly, summing together (4.3) for all $\ell \in [k]$, we have

$$\sum_{\ell=1}^{k} p_{j_\ell} = \gamma(S^*_{\tau(i)-1} \setminus \{i\} \cup \{j_1, \ldots, j_k\}) - \gamma(S^*_{\tau(i)-1} \setminus \{i\}) = \gamma(S^*_{\tau(i)} \setminus \{i\}) - \gamma(S^*_{\tau(i)-1} \setminus \{i\}).$$

Combining the above two equalities, we have

$$\gamma(S^*_{\tau(i)}) - \gamma(S^*_{\tau(i)-1}) = \gamma(S^*_{\tau(i)} \setminus \{i\}) - \gamma(S^*_{\tau(i)-1} \setminus \{i\}) \quad \Rightarrow \quad \gamma(S^*_{\tau(i)-1}) = \gamma(S^*_{\tau(i)-1} \setminus \{i\}),$$

where the second equality follows since by assumption $0 = \gamma(S^*_{\tau(i)}) - \gamma(S^*_{\tau(i)} \setminus \{i\})$. We have reached a contradiction since we assumed $\tau(i)$ is the smallest time such that $0 = \gamma(S_t^*) - \gamma(S_t^* \setminus$

$\{i\})$. □

Splitting and merging sets. The next lemma shows that each dependent set contains a dependent set containing only items of equal profit.

Lemma 4.2.7. *Let $S \subseteq [n]$ be dependent. Then there exists $S' \subseteq S$ where S' is dependent and for any $i, j \in S'$, $p_i = p_j$.*

Proof. We prove the statement by induction on $|S|$. Let $|S| = 1$. Since we assume all sets of cardinality 1 are independent, the statement is vacuously true. For the general case, let $C \subseteq S$ be a cycle, whose existence is guaranteed by Lemma 4.2.2. If $|C| < |S|$, then by inductive hypothesis, there exists $S' \subseteq C \subseteq S$ such that S' is dependent and contain only items with equal profits. Else, S is a cycle. For all $i \in S$, $S \setminus \{i\}$ is independent, that is, $\gamma(S \setminus \{i\}) = \sum_{j \in S \setminus \{i\}} p_j$. Furthermore, $\gamma(S) = \gamma(S \setminus \{i\})$, else if

$$\gamma(S) > \gamma(S \setminus \{i\}) = \sum_{j \in S \setminus \{i\}} p_j,$$

we must have $\gamma(S) = \sum_{j \in S} p_j$, and S is independent as well, a contradiction.

Thus, for $i, j \in S$,

$$\sum_{\ell \in S \setminus \{i\}} p_\ell = \gamma(S) = \sum_{\ell \in S \setminus \{j\}} p_\ell.$$

Cancelling out all p_ℓ for $\ell \notin \{i, j\}$ in the equality above, we get $p_i = p_j$. Taking $S' = S$ concludes the proof. □

Let $S^1, \ldots, S^k$ be the restriction of S to each profit class $\mathcal{P}_1, \ldots, \mathcal{P}_k$. The next lemma shows that, in order to compute the function γ on a set S, we can sum the profits of $S^1, \ldots, S^k$. In particular, if $S^1, \ldots, S^k$ are all independent, so is their union.

Lemma 4.2.8. *Let $S^1 \subseteq \mathcal{P}_1, \ldots, S^k \subseteq \mathcal{P}_k$. Then $\gamma(\cup_{\ell \in [k]} S^\ell) = \sum_{\ell \in [k]} \gamma(S^\ell)$. In particular, if $S^1 \ldots, S^k$ are independent, so is $\cup_{\ell \in [k]} S^\ell$.*

Proof. First, we will show that $\gamma(\cup_{\ell\in[k]} S^\ell) \leq \sum_{\ell\in[k]} \gamma(S^\ell)$. This follows by submodularity of γ. In particular,

$$\begin{aligned}
\gamma(S^1) + \cdots + \gamma(S^k) &\geq \gamma(S^1 \cup S^2) + \gamma(S^3) + \cdots + \gamma(S^k) \\
&\geq \gamma(\cup_{k\in[3]} S^k) + \gamma(S^4) + \cdots + \gamma(S^k) \\
&\vdots \\
&\geq \gamma(\cup_{\ell\in[k]} S^\ell).
\end{aligned}$$

Here, in every inequality, we are using, for every $k' \in [k]$ and $k' \geq 2$,

$$\begin{aligned}
\gamma(\cup_{\ell\in[k'-1]} S^\ell) + \gamma(S^{k'}) &\geq \gamma(\cup_{\ell\in[k']} S^\ell) + \gamma((\cup_{\ell\in[k'-1]} S^\ell) \cap S^{k'}) \\
&= \gamma(\cup_{\ell\in[k']} S^\ell),
\end{aligned}$$

where the first inequality follows since γ is submodular, and the second inequality follows since $(\cup_{\ell\in[k'-1]} S^\ell) \cap S^{k'} = \emptyset$ and $\gamma(\emptyset) = 0$.

Next, we will show that $\gamma(\cup_{\ell\in[k]} S^\ell) \geq \sum_{\ell\in[k]} \gamma(S^\ell)$. For each S^ℓ, by Lemma 4.2.4, there exists $S^{\ell,I} \subseteq S^\ell$ such that $S^{\ell,I}$ is independent and $\gamma(S^\ell) = \gamma(S^{\ell,I}) = \sum_{i\in S^{\ell,I}} p_i$. Summing the equality over all $\ell \in [k]$, we have

$$\sum_{\ell\in[k]} \gamma(S^\ell) = \sum_{\ell\in[k]} \gamma(S^{\ell,I}). \tag{4.4}$$

Next, we claim that $\cup_{\ell\in[k]} S^{\ell,I}$ is also independent. To see this, by the contrapositive of Lemma 4.2.7, if there does not exist $S' \subseteq \cup_{\ell\in[k]} S^{\ell,I}$ such that S' is dependent and all elements in S' have equal profits, then $\cup_{\ell\in[k]} S^{\ell,I}$ must be independent. If such an S' exist, $S' \subseteq S^{\ell,I}$ for some $\ell \in [k]$, else S' contains elements of distinct profits. By construction, $S^{\ell,I}$ is independent. Therefore, all subsets of $S^{\ell,I}$ must be independent by Lemma 4.2.3. Thus, S' does not exist and we conclude $\cup_{\ell\in[k]} S^{\ell,I}$ is independent. Hence, we have

$$\begin{aligned}\gamma(\cup_{\ell\in[k]}S^{\ell,I}) &= \sum_{i\in\cup_{\ell\in[k]}S^{\ell,I}} p_i \\ &= \sum_{\ell\in[k]}\sum_{i\in S^{\ell,I}} p_i \\ &= \sum_{\ell\in[k]}\gamma(S^{\ell,I}) \\ &= \sum_{\ell\in[k]}\gamma(S^{\ell}),\end{aligned}$$

where the first equality follows from the independence of $\cup_{\ell\in[k]}S^{\ell,I}$, the second equality follows since $S^{1,I},\dots S^{k,I}$ are pairwise disjoint, the third equality follows since $S^{\ell,I}$ is independent for $\ell\in[k]$, and the final equality follows from (4.4). Finally, note that since $\cup_{\ell\in[k]}S^{\ell,I}\subseteq\cup_{\ell\in[k]}S^{\ell}$, we have $\gamma(\cup_{\ell\in[k]}S^{\ell})\geq\gamma(\cup_{\ell\in[k]}S^{\ell,I})$ by monotonicity. We conclude that $\gamma(\cup_{\ell\in[k]}S^{\ell})\geq\sum_{\ell\in[k]}\gamma(S^{\ell})$.

As a direct consequence of $\gamma(\cup_{\ell\in[k]}S^{\ell})=\sum_{\ell\in[k]}\gamma(S^{\ell})$, if $S^1,\dots,S^k$ are independent, then

$$\begin{aligned}\gamma(\cup_{\ell\in[k]}S^{\ell}) &= \sum_{\ell\in[k]}\gamma(S^{\ell}) \\ &= \sum_{\ell\in[k]}\sum_{i\in S^{\ell}} p_i \\ &= \sum_{i\in\cup_{\ell\in[k]}S^{\ell}} p_i,\end{aligned}$$

where the second equality follows since S^ℓ is independent and the final equality follows since $S^1,\dots,S^k$ are pairwise disjoint. We conclude that $\cup_{\ell\in[k]}S^{\ell}$ is also independent. □

4.2.3 Independent sets in single profit classes

Slicing by profit. Let $[n]$ be the ground set of an IK−AoN instance. We partition $[n]$ into sets $\mathcal{P}_1,\dots,\mathcal{P}_k$ by profit. That is, for any $\ell\in[k]$ and for any $i,j\in\mathcal{P}_\ell$, we have $p_i=p_j$; for any $\ell\neq\ell'\in[k]$ and any $i\in\mathcal{P}_\ell$, $j\in\mathcal{P}_{\ell'}$, we have $p_i\neq p_j$.

Lemma 4.2.9. *For any $\ell\in[k]$, let $\mathcal{M}_\ell\subseteq 2^{\mathcal{P}_\ell}$ denote the family of independent sets of $\mathcal{P}_\ell$. Then*

$(\mathcal{P}_\ell, \mathcal{M}_\ell)$ *forms a matroid.*

Proof. Trivially, $\emptyset \in \mathcal{M}_\ell$. For any set $S \in \mathcal{M}_\ell$ and any $S' \subseteq S$, if S is independent, so is S' by Lemma 4.2.3.

It remains to verify that given any $A \subseteq \mathcal{P}_\ell$ and inclusionwise maximal independent sets $S, S' \subseteq A$, we have $|S| = |S'|$. By way of contradiction, without loss of generality, assume $|S| > |S'|$. Let $\{e_1, \ldots, e_k\} = S \setminus S'$, which we know to be nonempty since $|S| > |S'|$. Let $p^{(\ell)}$ denote the unique profit of every item $i \in \mathcal{P}_\ell$. Since we know that both S and S' are independent, by monotonicity

$$\gamma(S \cup S') \geq \gamma(S) = |S| \cdot p^{(\ell)} > |S'| \cdot p^{(\ell)} = \gamma(S'). \tag{4.5}$$

Since $S' \cup (S \setminus S') = S \cup S'$ and $\gamma(S \cup S') > \gamma(S')$, we claim that there must exist item $e \in S \setminus S' \subseteq A$ such that $\gamma(S' \cup \{e\}) = \gamma(S') + p^{(\ell)}$, a contradiction since we assumed S' to be maximal in A. Indeed, if $\gamma(S' \cup \{e\}) = \gamma(S')$ for all $e \in S \setminus S'$, we have by submodularity that, for $i \in [k]$,

$$0 = \gamma(S' \cup \{e\}) - \gamma(S') \geq \gamma(S' \cup \{e_1, \ldots, e_i\}) - \gamma(S' \cup \{e_1, \ldots, e_{i-1}\}).$$

Hence by telescoping sum

$$\gamma(S \cup S') - \gamma(S') = \gamma(S' \cup \{e_1, \ldots, e_k\}) - \gamma(S') \leq 0,$$

contradicting (4.5). □

Restriction to independent sets of minimum weight. Let $\mathcal{P}_1, \ldots, \mathcal{P}_k$ be the partition of $[n]$ into profit classes and let $\mathcal{P}_1^I, \ldots, \mathcal{P}_k^I$ be the corresponding independent sets as defined by the algorithm in Section 4.2.1. Note that the set $\mathcal{P}_\ell^I$ output by the algorithm is an inclusionwise maximal independent set of $\mathcal{P}_\ell$. We create a sub-instance of $\mathcal{I}$, which we denote $\mathcal{I}_\ell$ by restricting to items in $\mathcal{P}_\ell$ with the same profit, weight and feasibility constraints as the original instance, $\mathcal{I}$. Similarly, we create a sub-instance of $\mathcal{I}_\ell$, which we denote by $\mathcal{I}_\ell^I$, restricted to the items in $\mathcal{P}_\ell^I$, with the same

profit, weight, and feasibility constraints. Note that since $\mathcal{P}_\ell^I$ is an independent set, all subsets of $\mathcal{P}_\ell^I$ are also independent by Lemma 4.2.3. Therefore, by Observation 4.2.5, $\mathcal{I}_\ell^I$ is an instance of the incremental knapsack problem.

Lemma 4.2.10. *Given any independent chain $\mathcal{S}$ for $\mathcal{I}_\ell$, there exists an independent chain $\mathcal{S}'$ for $\mathcal{I}_\ell^I$ such that $\gamma(S_t) = \gamma(S'_t)$ and $w(S'_t) \leq w(S_t)$ for all $t \in [T]$.*

Proof. We construct $\mathcal{S}'$ as follows. For every $t \in [T]$, let S'_t be the $|S_t|$ items of minimum weight in $\mathcal{P}_\ell^I$, breaking ties following the order of the items in $\mathcal{P}_\ell$. Since $\mathcal{S}$ is a chain, $|S_1| \leq \cdots \leq |S_T|$. Therefore $S'_1 \subseteq \cdots \subseteq S'_T$, that is, $\mathcal{S}'$ is a chain.

By Lemma 4.2.9, $(\mathcal{P}_\ell, \mathcal{M}_\ell)$ forms a matroid. Since $\mathcal{P}_\ell^I$ is an inclusionwise maximal independent set in $\mathcal{P}_\ell$ and since S_T is also an independent set in $\mathcal{P}_\ell$, we have $|S_T| \leq |\mathcal{P}_\ell^I|$. Thus, $\mathcal{S}'$ is well defined. Since, for all $t \in [T]$, S'_t is an independent set and all items in $\mathcal{P}_\ell$ have equal profit, clearly $\gamma(S_t) = \gamma(S'_t)$.

Suppose by contradiction there exists $t \in [T]$ such that $w(S'_t) > w(S_t)$. Let $S'_t = \{e_1, \ldots, e_m\}$ such that $w_{e_1} \leq \cdots \leq w_{e_m}$. Similarly, let $S_t = \{q_1, \ldots, q_m\}$ such that $w_{q_1} \leq \cdots \leq w_{q_m}$. Let k be the smallest index such that $w_{q_k} < w_{e_k}$. The existence of such an index follows from $w(S_t) < w(S'_t)$ and $|S_t| = |S'_t|$. Take $A = \{e_1, \ldots, e_{k-1}, q_1, \ldots, q_k\} \subseteq \mathcal{P}_\ell$. We claim that $\{e_1, \ldots, e_{k-1}\}$ is a maximal independent set in A. By definition, $w_{q_1} \leq w_{q_2} \leq \cdots \leq w_{q_k} < w_{e_k}$. Since $e_k \in S'_t$, for any item $q_s \in \{q_1, \ldots, q_k\} \setminus \{e_1, \ldots, e_{k-1}\}$, we must have $q_s \notin \mathcal{P}_\ell^I$. In constructing $\mathcal{P}_\ell^I$, since e_k is added but q_s is not, we must have that $\{e_1, \ldots, e_{k-1}\} \cup \{q_s\}$ is dependent. Hence, $\{e_1, \ldots, e_{k-1}\}$ is an inclusionwise maximal independent set of A. Now, clearly $\{q_1, \ldots, q_k\} \subseteq A$ is an independent set with larger cardinality, a contradiction since $(\mathcal{P}_\ell, \mathcal{M}_\ell)$ forms a matroid by Lemma 4.2.9. Hence, it must be that $w(S'_t) \leq w(S_t)$ for all $t \in [T]$, concluding the proof. □

4.2.4 A decomposition theorem for monotone submodular all-or-nothing functions

The previous results imply a decomposition theorem for monotone submodular all-or-nothing functions. Although not strictly needed for our purposes, we present it nevertheless, since it could be of interest for further applications of those functions.

Theorem 4.2.11. *Let $\gamma : 2^{[n]} \to \mathbb{N}$ be a function, and let $\mathcal{P}_1, \ldots, \mathcal{P}_k$ be its profit classes, corresponding to profits $p^{(1)} < \cdots < p^{(k)}$. γ is a monotone submodular all-or-nothing function if and only if there exist matroid rank functions $\{r_\ell\}_{\ell \in [k]}$ such that, for each $S \subseteq [n]$, we have:*

$$\gamma(S) = \sum_{\ell \in [k]} p^{(\ell)} \cdot r_\ell(S \cap \mathcal{P}_\ell).$$

Proof. First, we prove the "only if" direction. By Lemma 4.2.9, $(\mathcal{P}_\ell, \mathcal{M}_\ell)$ forms a matroid where $\mathcal{M}_\ell$ is the family of independent sets of $\mathcal{P}_\ell$. Let r_ℓ be the rank function of the associated matroid. For every $\ell \in [k]$, let $S^\ell = S \cap \mathcal{P}_\ell$. Observe that since $\mathcal{P}_1, \ldots, \mathcal{P}_k$ is a partition of $[n]$, we have $S = \cup_{\ell \in [k]} S^\ell$. By Lemma 4.2.8, $\gamma(S) = \sum_{\ell \in k} \gamma(S^\ell)$.

For each $\ell \in [k]$, by definition of the rank function r_ℓ, the independent set of maximum cardinality $S^{\ell,I} \subseteq S^\ell$ has $|S^{\ell,I}| = r_\ell(S^\ell)$. Thus, by Lemma 4.2.4 and monotonicity of γ, for $\ell \in [k]$,

$$\gamma(S^\ell) = \gamma(S^{\ell,I}) = \sum_{i \in S^{\ell,I}} p_i = p^{(\ell)} \cdot r_\ell(S^\ell).$$

Summing the above equality over all $\ell \in [k]$, we obtain:

$$\gamma(S) = \sum_{\ell \in k} p^{(\ell)} \cdot r_\ell(S^\ell) = \sum_{\ell \in k} p^{(\ell)} \cdot r_\ell(S \cap \mathcal{P}_\ell).$$

In the reverse direction, let $(\mathcal{P}_1, \mathcal{M}_1), \ldots, (\mathcal{P}_k, \mathcal{M}_k)$ be k matroids with rank functions $r_1, \ldots, r_k$ where $\mathcal{P}_1, \ldots \mathcal{P}_k$ form a partition of $[n]$. We will show that if

$$\gamma(S) = \sum_{\ell \in [k]} p^{(\ell)} \cdot r_\ell(S \cap \mathcal{P}_\ell),$$

for all $S \subseteq [n]$, then $\gamma : 2^{[n]} \to \mathbb{N}$ is a monotone submodular all-or-nothing function. By definition of monotone submodular all-or-nothing function, it is sufficient to show that γ satisfies the following properties:

- *All or nothing profits:* Let $i \in \mathcal{P}_{\ell^*}$ for some arbitrary $\ell^* \in [k]$. Let $S \subseteq [n]$. We consider

two cases:

1. If $r_{\ell^*}((S \cup \{i\}) \cap \mathcal{P}_{\ell^*}) = r_{\ell^*}(S \cap \mathcal{P}_{\ell^*})$, then

$$\begin{aligned}\gamma(S \cup \{i\}) &= \sum_{\ell \in [k]} p^{(\ell)} \cdot r_\ell((S \cup \{i\}) \cap \mathcal{P}_\ell) \\ &= \sum_{\ell \in [k]} p^{(\ell)} \cdot r_\ell(S \cap \mathcal{P}_\ell) \\ &= \gamma(S),\end{aligned}$$

   where the first and final equality are by definition of γ, the second equality is by noting $(S \cup \{i\}) \cap \mathcal{P}_\ell = S \cap \mathcal{P}_\ell$ for all $\ell \in [k] \setminus \{\ell^*\}$ and $r_{\ell^*}((S \cup \{i\}) \cap \mathcal{P}_{\ell^*}) = r(S \cap \mathcal{P}_{\ell^*})$ by assumption.

2. If $r_{\ell^*}((S \cup \{i\}) \cap \mathcal{P}_{\ell^*}) = r_{\ell^*}(S \cap \mathcal{P}_{\ell^*}) + 1$, then

$$\begin{aligned}\gamma(S \cup \{i\}) &= \sum_{\ell \in [k]} p^{(\ell)} \cdot r_\ell((S \cup \{i\}) \cap \mathcal{P}_\ell) \\ &= \sum_{\ell \in [k] \setminus \ell^*} p^{(\ell)} \cdot r_\ell(S \cap \mathcal{P}_\ell) + p^{(\ell^*)} \cdot r_{\ell^*}((S \cup \{i\}) \cap \mathcal{P}_{\ell^*}) \\ &= \sum_{\ell \in [k] \setminus \ell^*} p^{(\ell)} \cdot r_\ell(S \cap \mathcal{P}_\ell) + p^{(\ell^*)} \cdot r_{\ell^*}(S \cup \mathcal{P}_{\ell^*}) + p^{(\ell^*)} \\ &= \gamma(S) + p^{(\ell^*)},\end{aligned}$$

   where the first and final equality is by definition of γ, the second equality is by noting $(S \cup \{i\}) \cap \mathcal{P}_\ell = S \cap \mathcal{P}_\ell$ for all $\ell \in [k] \setminus \{\ell^*\}$, the third equality is by assumption.

- *Monotonicity:* Since each rank function is monotone, γ is clearly monotone.

- *Submodularity:* Since each rank function is submodular, $p^{(\ell)} \cdot r_\ell(\cdot)$ is submodular. Sum of submodular functions is submodular, hence γ is submodular.

□

4.2.5 Proof of Theorem 1.3.4

Let $\mathcal{I}$ be an IK−AoN instance in the input of the algorithm from Section 4.2.1 and let $\mathcal{I}_{\text{IK}}$ be obtained from $\mathcal{I}$ following the algorithm. Notice that $\mathcal{I}_{\text{IK}}$ belongs to the linearization of $\{\mathcal{I}\}$. Given any chain $\mathcal{S}$, let $\Phi(\mathcal{S}) = \sum_{t\in T} \Delta_t \cdot \gamma(S_t)$ denote the profit of the chain $\mathcal{S}$ in the instance $\mathcal{I}$. Similarly, let $\Phi_{\text{IK}}(\mathcal{S})$ denote the profit chain $\mathcal{S}$ earns in the instance $\mathcal{I}_{\text{IK}}$. Since $\mathcal{I}_{\text{IK}}$ is an instance of the incremental knapsack problem, for any chain $\mathcal{S}$ feasible in $\mathcal{I}_{\text{IK}}$, its profit is given by $\Phi_{\text{IK}}(\mathcal{S}) = \sum_{t\in T} \Delta_t \sum_{i\in S_t} p_i$. Note that $\overline{\mathcal{S}}$, as output by the algorithm, is a feasible chain of both $\mathcal{I}_{\text{IK}}$ (by construction) and $\mathcal{I}$ (since linearization does not affect feasibility).

The next lemma shows that in order to solve $\mathcal{I}$, it is sufficient to solve $\mathcal{I}_{\text{IK}}$.

Lemma 4.2.12. *Let $\mathcal{S}^*$ be the optimal solution of $\mathcal{I}$ and let $\mathcal{S}^*_{IK}$ be the optimal solution of $\mathcal{I}_{IK}$. Then $\Phi_{IK}(\mathcal{S}^*_{IK}) \geq \Phi(\mathcal{S}^*)$. Furthermore, given any solution $\mathcal{S}_{IK}$ feasible for $\mathcal{I}_{IK}$, $\mathcal{S}_{IK}$ is feasible for $\mathcal{I}$ with $\Phi(\mathcal{S}_{IK}) = \Phi_{IK}(\mathcal{S}_{IK})$.*

Proof. By Lemma 4.2.6, we can assume that $\mathcal{S}^*$ is independent. We decompose $\mathcal{S}^* = (S^*_1, \ldots, S^*_T)$ into k separate chains, each chain containing only items in $\mathcal{P}_\ell$ for some $\ell \in [k]$. More precisely, for each $\ell \in [k]$, let $\mathcal{S}^*|_{\mathcal{P}_\ell} = (S^*_1 \cap \mathcal{P}_\ell, \ldots, S^*_T \cap \mathcal{P}_\ell)$. One easily verifies that $\mathcal{S}^*|_{\mathcal{P}_\ell}$ is indeed a chain for each $\ell \in [k]$. Moreover, since $\mathcal{S}^*$ is independent, $\mathcal{S}^*|_{\mathcal{P}_\ell}$ is also independent by Lemma 4.2.3. It follows that

$$\Phi(\mathcal{S}^*) = \sum_{t\in[T]} \Delta_t \cdot \gamma(S^*_t) = \sum_{t\in[T]} \Delta_t \cdot \sum_{\ell\in[k]} \gamma((S^*|_{\mathcal{P}_\ell})_t), \tag{4.6}$$

where the second equality is by Lemma 4.2.8.

By Lemma 4.2.10, for each independent chain $\mathcal{S}^*|_{\mathcal{P}_\ell}$, there exists an independent chain $\mathcal{S}'|_{\mathcal{P}^I_\ell}$ restricted to the items in $\mathcal{P}^I_\ell$ such that $\gamma((S'|_{\mathcal{P}^I_\ell})_t) = \gamma((S^*|_{\mathcal{P}_\ell})_t)$ and $w((S'|_{\mathcal{P}^I_\ell})_t) \leq w((S^*|_{\mathcal{P}_\ell})_t)$.

Let $\mathcal{S}' = \cup_{\ell\in[k]} \mathcal{S}'|_{\mathcal{P}^I_\ell}$. It is easy to see that $\mathcal{S}'$ is a well defined chain for the instance $\mathcal{I}_{\text{IK}}$, since

it only consists of items in $\cup_{\ell\in[k]}\mathcal{P}_\ell^I$. To see that $\mathcal{S}'$ is feasible for $\mathcal{I}_{\mathtt{IK}}$, note that for each $t \in [T]$,

$$w(S'_t) = \sum_{\ell\in[k]} w((S'|_{\mathcal{P}_\ell^I})_t)) \leq \sum_{\ell\in[k]} w((S^*|_{\mathcal{P}_\ell})_t) = w(S^*_t) \leq W_t,$$

where the first inequality follows by Lemma 4.2.10, and the final inequality follows by feasibility of $\mathcal{S}^*$ in $\mathcal{I}$. To compute the profit of $\mathcal{S}'$, note that

$$\begin{aligned}
\Phi_{\mathtt{IK}}(\mathcal{S}') &= \sum_{t\in[T]} \Delta_t \sum_{\ell\in[k]} \sum_{i\in S'_t\cap\mathcal{P}_\ell^I} p_i \\
&= \sum_{t\in[T]} \Delta_t \sum_{\ell\in[k]} \gamma((S'|_{\mathcal{P}_\ell^I})_t)) \\
&= \sum_{t\in[T]} \Delta_t \sum_{\ell\in[k]} \gamma((S^*|_{\mathcal{P}_\ell})_t) \\
&= \Phi(\mathcal{S}^*),
\end{aligned}$$

where the first equality follows by definition, the second equality follows since for every $\ell \in [k]$, $\mathcal{S}'|_{\mathcal{P}_\ell^I}$ is independent in $\mathcal{I}$, the third equality follows again by Lemma 4.2.10, and the final equality follows by (4.6). Following the analysis above, $\mathcal{S}'$ is feasible in $\mathcal{I}_{\mathtt{IK}}$ with $\Phi_{\mathtt{IK}}(\mathcal{S}') = \Phi(\mathcal{S}^*)$. It follows that the optimal chain $\mathcal{S}^*_{\mathtt{IK}}$ has $\Phi_{\mathtt{IK}}(\mathcal{S}^*_{\mathtt{IK}}) \geq \Phi(\mathcal{S}^*)$. This concludes the proof of the first part of the lemma. To prove the second part of the lemma, let $\mathcal{S}^*_{\mathtt{IK}}$ be feasible for $\mathcal{I}_{\mathtt{IK}}$. Then it is clearly feasible for $\mathcal{I}$. To show that $\Phi(\mathcal{S}_{\mathtt{IK}}) = \Phi_{\mathtt{IK}}(\mathcal{S}_{\mathtt{IK}})$, it suffices to show that $\mathcal{S}_{\mathtt{IK}}$ is an independent chain in $\mathcal{I}$. We again decompose $\mathcal{S}_{\mathtt{IK}}$ into k separate chains. For each $\ell \in [k]$, let $\mathcal{S}_{\mathtt{IK}}|_{\mathcal{P}_\ell^I} = ((S_{\mathtt{IK}})_1 \cap \mathcal{P}_\ell^I, \ldots, (S_{\mathtt{IK}})_T \cap \mathcal{P}_\ell^I)$. $(S_{\mathtt{IK}})_T \cap \mathcal{P}_\ell^I$ is independent by Lemma 4.2.3 since $\mathcal{P}_\ell^I$ is independent by construction. Then $\cup_{\ell\in[k]}((S_{\mathtt{IK}})_T \cap \mathcal{P}_\ell^I)$ is independent by Lemma 4.2.8. Equivalently, $\mathcal{S}_{\mathtt{IK}}$ is independent in $\mathcal{I}$. □

Finally, we conclude this section by showing the algorithm given in Section 4.2.1 implies Theorem 1.3.4.

Proof of Theorem 1.3.4. Correctness follows immediately from Lemma 4.2.12.

To see the statement on running time, notice that in additional to implementing the α-

approximated algorithm, the procedure requires constructing $\mathcal{I}_{\mathtt{IK}}$. This construction can clearly be done in $O(nT)$ time, as it consists of constructing up to n disjoint independent sets, with all other parameters identical to $\mathcal{I}$. □

4.3 A PTAS for the generalized incremental knapsack problem with a bounded number of times

In this section, we give a PTAS to the generalized incremental knapsack problem when T is assumed to be a constant. The idea of the PTAS is to "guess" the $O(\frac{T^3}{\epsilon})$ most profitable items that are inserted into the knapsack and their insertion times, and then solve the linear relaxation of the residual problem with the remaining less profitable items. In Section 4.3.1, we give the algorithm in detail. In Section 4.3.2, we introduce a rounding procedure that guarantees that the optimal solution of the linear relaxation has at most $O(T^3)$ fractional elements. This solution is then rounded down to an integer solution, proven to be near optimal in Section 4.3.3.
The formal statement, presented in Section 1.3, is reprinted here for convenience.

Theorem 1.3.5. *For any $\epsilon > 0$, the generalized incremental knapsack problem can be approximated within a factor of $(1-\epsilon)$ in time $O((nT)^{O(\frac{T^3}{\epsilon})})$. Therefore, it admits a PTAS when T is a constant.*

4.3.1 Preliminaries and algorithm

Preliminaries. Extending notations from Section 3.3.1, given a feasible chain $\mathcal{G} = (G_1, \ldots, G_T)$ for an instance of the generalized incremental knapsack problem $\mathcal{I} = (\mathcal{N}, W)$, we define the *reduced residual* generalized incremental knapsack instance $\mathcal{I}_{p^-}^{-\mathcal{G}} = (\mathcal{N}^{-\mathcal{G}}, W^{-\mathcal{G}})$ as follows:

- The new set of items is $\mathcal{N}^{-\mathcal{G}} = \mathcal{N} \setminus G_T$.
- The residual capacity of every time $t \in [T]$ is set to $W_t^{-\mathcal{G}} = \min_{t \le \tau \le T}(W_\tau - w(G_\tau))$.
- All item weights remain unchanged.

- Let $p^- = \min\{p_{i,t} : i \in G_t \setminus G_{t-1}, t \in [T]\}$, so p^- is the profit the least profitable item earns among all items in G_T. For all $i \in \mathcal{N}^{-\mathcal{G}}$, let its profit $p^-_{i,t} = p_{i,t}$ if $p_{i,t} \leq p^-$. Otherwise, let $p^-_{i,t} = 0$.

Given any feasible chain $\mathcal{S}$ of $\mathcal{I}^{-\mathcal{G}}_{p^-}$, we let $\Phi_{p^-}(\mathcal{S})$ denote the profit $\mathcal{S}$ earns in the instance $\mathcal{I}^{-\mathcal{G}}_{p^-}$. Notice that the above definition differs from the definition of $\mathcal{I}^{-\mathcal{G}} = (\mathcal{N}^{-\mathcal{G}}, W^{-\mathcal{G}})$ given in Section 3.3.1, where profits remain unchanged for all items. Other than the difference in definitions on the profit of items, these two definitions are identical. Therefore, since $\mathcal{I}^{-\mathcal{G}}$ is well defined, so is $\mathcal{I}^{-\mathcal{G}}_{p^-}$. Furthermore, observe that since profits can only decrease from $\mathcal{I}$ to $\mathcal{I}^{-\mathcal{G}^+}_{p^-}$, given any chain $\mathcal{S}$ feasible in $\mathcal{I}^{-\mathcal{G}^+}_{p^-}$, we have the following inequality:

$$\Phi(\mathcal{S}) \geq \Phi_{p^-}(\mathcal{S}). \tag{4.7}$$

The algorithm. Given a generalized incremental knapsack instance $\mathcal{I} = (\mathcal{N}, \mathcal{W})$ and any $\epsilon > 0$ where we assume without loss of generality $\frac{1}{\epsilon}$ is an integer, the algorithm is as follows:

1. For every feasible chain $\mathcal{G} = (G_1, \ldots, G_T)$ with $|G_T| \leq \frac{T^2(T+1)}{2\epsilon}$,
   (a) Construct the residual instance $\mathcal{I}^{-\mathcal{G}}_{p^-}$.
   (b) Solve the linear relaxation of (GIK-IP) for the instance $\mathcal{I}^{-\mathcal{G}}_{p^-}$ as to obtain x^*.
   (c) Round x^* to $\bar{x}$ as described by Lemma 4.3.1.
   (d) Let $x' = \lfloor \bar{x} \rfloor$ and construct $\mathcal{S}^{-\mathcal{G}}$ where for all $t \in [T]$, $S^{-\mathcal{G}}_t = \{i \in \mathcal{N}^{-\mathcal{G}} : x'_{i,t} = 1\}$.
2. Return the chain $\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}$ of maximum profit among those constructed in Step 1 .

Analysis: feasibility and running time. For any feasible chain $\mathcal{G}$ constructed in step 1 above, x' is feasible for (GIK-IP) since x^* is feasible for its linear relaxation. Feasibility of $\mathcal{S}^{-\mathcal{G}}$ in $\mathcal{I}^{-\mathcal{G}}_{p^-}$ follows by construction. Feasibility of $\mathcal{G} \cup \mathcal{S}^{-\mathcal{G}}$ in $\mathcal{I}$ follows by Lemma 3.3.3. Since we only consider chains that introduce at most $\frac{T^2(T+1)}{2\epsilon}$, the number of chains being enumerated over in step 1 is $O((nT)^{O(\frac{T^3}{\epsilon})})$. The remaining procedure is dominated by solving a linear program, which can

be done in $O(|\mathcal{I}|^{O(1)})$ time. As seen in Lemma 4.3.1, the rounding procedure can be done in time $O((nT)^{O(1)})$. All in all, this algorithm incurs a running time of $O((nT)^{O(\frac{T^3}{\epsilon})})$.

4.3.2 The LP rounding procedure

Before proving the correctness of the algorithm, we first give the rounding procedure to obtain $\bar{x}$ in step 1(c). Since the rounding procedure relies on the formulation of (GIK-IP) extensively, we reprint it here for convenience. We let (GIK-LP) denote the linear relaxation of (GIK-IP) where the binary constraints of $x_{i,t}$ are replaced by $0 \leq x_{i,t} \leq 1$.

$$\begin{aligned}
\max \quad & \sum_{i\in[n]} \sum_{t\in[T]} p_{i,t}(x_{i,t} - x_{i,t-1}) & \\
\text{s.t.} \quad & \sum_{i\in[n]} w_i x_{i,t} \leq W_t & \forall\, t \in [T] \\
& x_{i,t} \leq x_{i,t+1} & \forall\, i \in [n], t \in [T-1] \\
& x_{i,t} \in \{0,1\} & \forall\, i \in [n], t \in [T]
\end{aligned} \tag{GIK-IP}$$

Lemma 4.3.1. *Given an optimal solution x^* to* (GIK-LP), we can obtain in time $O((nT)^{O(1)})$ a solution $\bar{x}$ of equal profit such that the number of fractional variables is no more than $\frac{T^2(T+1)}{2}$.

Proof. For an item i, we define its *starting time* $s(i)$ as the first time t such that $0 < x^*_{i,t} < 1$ and its *ending time* $e(i)$ the last time t such that $0 < x^*_{i,t} < 1$. Notice that if $s(i)$ does not exist, then for all $t \in [T]$, we have $x^*_{i,t} \in \{0,1\}$. On the other hand, if $s(i)$ exists, so does $e(i)$. If $0 < x^*_{i,T} < 1$, by definition, we let $e(i) = T$.

Now restrict the attention to items i for which both $s(i)$ and $e(i)$ exist – we call them *fractionally inserted*. For a fractionally inserted item i, we call the pair $(s(i), e(i))$ the *signature* of i.

We claim that there exists an optimal solution to (GIK-LP) such that no two fractionally inserted items have the same signature. We construct it by suitably modifying x^* as follows. Suppose there exists $s(j) = s(k) = t'$ and $e(j) = e(k) = t''$ for some $j \neq k$. Assume without loss of

generality that

$$\frac{p_{j,t'} - p_{j,t''+1}}{w_j} \leq \frac{p_{k,t'} - p_{k,t''+1}}{w_k}, \tag{4.8}$$

where by convention we let $p_{i,T+1} = 0$ for all $i \in [n]$. Let $\epsilon = \min\{1 - x^*_{k,t''}\,;\, x^*_{j,t'}\frac{w_j}{w_k}\}$. Notice that by construction $\epsilon > 0$. Let $\bar{x}_{k,t} = x^*_{k,t} + \epsilon$ and $\bar{x}_{j,t} = x^*_{j,t} - \frac{\epsilon w_k}{w_j}$, for all $t' \leq t \leq t''$. For all other pairs $(i,t) \in [n] \times [T]$, let $\bar{x}_{i,t} = x^*_{i,t}$. The next two claims establish that $\bar{x}$ is feasible, has less fractional components, while still maintaining optimality. We defer the proofs to Appendices D.2.1 and D.2.2 respectively.

Claim 4.3.2. *$\bar{x}$ is a feasible solution to* (GIK-LP), and and has at least one fractional component less than x^*.

Claim 4.3.3. *$\bar{x}$ is an optimal solution to* (GIK-LP).

Hence, starting from x^*, while there are two items with the same signature, we can construct another optimal solution with at least one fractional component less. As this process clearly never loops, it terminates after at most nT steps with an optimal solution where no two items have the same signature. Since each step can be executed in constant time, the running time follows. Any item without a signature is never fractional. There are at most $\frac{T(T+1)}{2}$ distinct signatures. Each item with a signature has at most T fractional components. Thus, there are at most $\frac{T^2(T+1)}{2}$ fractional components. □

4.3.3 Proof of Theorem 1.3.5

Finally, in this section, we prove the following lemma lower bounding the profit of the chain $\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}$, which, along with the feasibility and running time analysis in Section 4.3.1, concludes the proof for Theorem 1.3.5.

Lemma 4.3.4. *Let $\mathcal{S}^*$ denote the optimal chain for the generalized incremental knapsack problem,*

$$\Phi(\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}) \geq (1 - \epsilon)\Phi(\mathcal{S}^*).$$

Proof. We assume that $|S_T^*| > \frac{T^2(T+1)}{2\epsilon}$, else the problem can be solved optimally through enumeration. For all $i \in S_T^*$, let $t^*(i)$ denote its insertion time. Rank S_T^* by $p_{i,t^*(i)}$ in non-increasing order and let G be the first $\frac{T^2(T+1)}{2\epsilon}$ items in S_T^*, breaking ties arbitrarily. Following the notation from Section 3.3.1, we let $\mathcal{G} = \mathcal{S}^*|_G = (S_1^* \cap G, \ldots, S_T \cap G)$. Since $|G| = \frac{T^2(T+1)}{2\epsilon}$, $\mathcal{G}$ is one of the chains considered in Step 1 of the algorithm. Let $\mathcal{R} = \mathcal{S}^*|_{\mathcal{N}\setminus\mathcal{G}}$. Since the set of available items and capacity are identical between $\mathcal{I}_{p^-}^{-\mathcal{G}}$ and $\mathcal{I}^{-\mathcal{G}}$, feasibility of $\mathcal{R}$ in $\mathcal{I}_{p^-}^{-\mathcal{G}}$ is guaranteed by Lemma 3.3.4. The following claim gives the profit of $\mathcal{R}$ in $\mathcal{I}_{p^-}^{-\mathcal{G}}$. Its straightforward proof is deferred until Appendix D.2.3.

Claim 4.3.5.

$$\Phi_{p^-}(\mathcal{R}) = \Phi(\mathcal{R}).$$

The next equality is an immediate result of the above claim and Lemma 3.3.4.

$$\Phi(\mathcal{S}^*) = \Phi_{p^-}(\mathcal{R}) + \Phi(\mathcal{G}) \tag{4.9}$$

Let $\mathcal{R}'$ be the chain returned by Step 1(d) of the algorithm for the instance $\mathcal{I}_{p^-}^{-\mathcal{G}}$. The next claim gives a lower bound on the profit of $\mathcal{R}'$ in terms of the profit of $\mathcal{R}$ and $\mathcal{G}$. Its proof is given in Appendix D.2.4.

Claim 4.3.6.

$$\Phi_{p^-}(\mathcal{R}') \geq \Phi_{p^-}(\mathcal{R}) - \epsilon\Phi(\mathcal{G}).$$

Putting the above analysis together, we conclude:

$$\begin{aligned}
\Phi(\mathcal{G}^* \cup \mathcal{S}^{-\mathcal{G}^*}) &\geq \Phi(\mathcal{G} \cup \mathcal{R}') \\
&= \Phi(\mathcal{G}) + \Phi(\mathcal{R}') \\
&\geq \Phi(\mathcal{G}) + \Phi_{p^-}(\mathcal{R}') \\
&\geq \Phi(\mathcal{G}) + \Phi_{p^-}(\mathcal{R}) - \epsilon\Phi(\mathcal{G}) \\
&= \Phi(\mathcal{S}^*) - \epsilon\Phi(\mathcal{G})
\end{aligned}$$

$$\geq \quad (1-\epsilon)\Phi(\mathcal{S}^*).$$

Here, the first inequality follows by Step 2 of the algorithm. The first equality follows since G_T and R'_T are disjoint. The second inequality follows by (4.7). The third inequality follows from Claim 4.3.6. The final equality follows by (4.9) and the final inequality follows since $\Phi(\mathcal{G}) \leq \Phi(\mathcal{S}^*)$. □

4.4 An FPTAS for the generalized incremental knapsack - single profit problem

We consider the *generalized incremental knapsack single profit* problem (`GIK-SP`), where, for each item i, $p_{i,t} > 0$ for exactly one time t, that we denote by $t(i)$. For all other times $t \neq t(i)$, $p_{i,t} = 0$. Since there is a unique nonzero profit for every item, we abbreviate $p_i = p_{i,t(i)}$. Thus, we can assume that, for each item i, we either insert it at time $t(i)$ or not at all. Notice that if we assume $T = 1$, the `GIK-SP` problem is exactly the classical knapsack problem, and hence the former is also NP-hard. We provide the following FPTAS based on a dynamic programming approach. The formal result from Section 1.3 is reprinted below.

Theorem 1.3.6. *For any $\epsilon > 0$, the generalized incremental knapsack single profit problem can be approximated within a factor of $1-\epsilon$ in time $O(\frac{n^3}{\epsilon})$.*

4.4.1 Continuous dynamic program

We order the items such that for items $i = 1, 2, \ldots, n_1$ we have $t(i) = 1$, for items $i = n_1 + 1, \ldots n_2$ we have $t(i) = 2$, and so on.

States and value function. Each state (i, p) of our dynamic programming consists of the following parameters:

- The index of the current item i, consisting of values $i \in [n]$.
- The total profit p collected thus far. Similarly to the dynamic programs presented in Sections 3.2.3.2 and 3.4.4, we initially treat p as a continuous parameter taking values in

$[0, np_{\max}]$ where $p_{\max} = \max\{p_{it} : i \in [n], t \in [T]\}$.

For each state (i, p), we let the value function $F(i, p)$ denote the minimum weight $w(S)$ that can be attained with items $S \subseteq [i]$ satisfying:

1. *Total profit:* $\sum_{j \in S} p_j \geq p$.

2. *Feasibility:* $\sum_{j \in S \cap [n_t]} w_j \leq W_t$ for all $t \in [T]$.

For every set $S \subseteq [i]$ that satisfies conditions 1 and 2 above, we call S a *candidate* to $F(i, p)$. When such a set S does not exist, we let $F(S) = \infty$. The next lemma shows that any candidate to $F(i, p)$ can be converted to a feasible chain to the generalized incremental knapsack problem with total profit of at least p. Its straightforward proof is deferred to Appendix D.3.1.

Lemma 4.4.1. *Given S that is a candidate to $F(i, p)$, we can obtain feasible chain $\mathcal{S}$ such that $\Phi(\mathcal{S}) \geq p$ in time $O(nT)$.*

Optimal substructure. Let S be the set that minimizes $F(i, p)$. Let $\hat{S} = S \setminus \{i\}$ and let

$$\hat{p} = \begin{cases} [p - p_i]^+ & \text{if } i \in S \\ p & \text{otherwise.} \end{cases}$$

Since $S \subseteq [i]$, $\hat{S} = S \setminus \{i\} \subseteq [i-1]$. Notice that the definition of $\hat{p}$ ensures that $\sum_{j \in \hat{S}} p_j \geq \hat{p}$ and feasibility of $\hat{S}$ follows from feasibility of S. Thus, $\hat{S}$ satisfies conditions 1 and 2 above. The next lemma shows that $\hat{S}$ is the set that minimizes $F(i-1, \hat{p})$.

Lemma 4.4.2. $F(i-1, p) = w(\hat{S})$.

Proof. Suppose by contradiction there exists $\tilde{S}$ that satisfies conditions 1 and 2 above such that $w(\tilde{S}) < w(\hat{S})$. First, suppose $i \notin S$. Then we claim $\tilde{S}$ is a candidate to $F(i, p)$. Clearly, $\tilde{S} \subseteq [i]$. $\tilde{S}$ satisfies $\sum_{j \in \tilde{S}} p_j \geq \hat{p} = p$ and $\sum_{j \in \hat{S} \cap [n_t]} w_j \leq W_t$ since $\tilde{S}$ is a candidate to $F(i-1, \hat{p})$. We have reached a contradiction since $w(\tilde{S}) < w(\hat{S}) = w(S)$.

Finally, assume $i \in S$. Then we claim $\tilde{S} \cup \{i\}$ is a candidate to $F(i, p)$. Clearly $\tilde{S} \cup \{i\} \subseteq [i]$. For the profit requirement, $p(\tilde{S} \cup \{i\}) = \sum_{j \in \tilde{S}} p_j + p_i \geq p - p_i + p_i = p$. For feasibility, for all $t < t(i)$,

$$\sum_{j \in (\tilde{S} \cup \{i\}) \cap [n_t]} w_j = \sum_{j \in \tilde{S} \cap [n_t]} w_j \leq W_t,$$

where the first equality is by noting that $i > n_t$ and the inequality is by noting that $\tilde{S}$ is a candidate to $F(i-1, \tilde{p})$. For $t \geq t(i)$, note that

$$\sum_{j \in \tilde{S} \cup \{i\} \cap [n_t]} w_i = \sum_{j \in \tilde{S} \cup \{i\}} w_j < \sum_{j \in \hat{S} \cup \{i\}} w_j \leq W_t,$$

where the equality is by noting $\tilde{S} \cup \{i\} \subseteq [i]$ and $i \leq n_t$, the first inequality follows by assumption and the final inequality follows since $\hat{S} \cup \{i\} = S$ is a candidate to $F(i, p)$. We have again arrived at a contradiction since $w(\tilde{S}) + w_i < w(\hat{S}) + w_i = w(S)$. □

Recursive equations. For states $(i, 0)$ for any $i \in [n]$, clearly the empty set achieves a profit of 0, so we initialize $F(i, 0) = 0$. For any $p > 0$, clearly there is no state that is a candidate to $F(0, p)$, thus we initialize $F(0, p) = \infty$. Given the optimal substructure unveiled above, for any other set $(i+1, p)$, we have the following recursion:

$$F(i+1, p) = \begin{cases} \min\{F(i, p), w_{i+1} + F(i, [p - p_{i+1}]^+)\}, & \text{if } w_{i+1} + W(i, p - p_{i+1}) \leq W_{t(i)} \\ F(i, p), & \text{otherwise.} \end{cases} \tag{4.10}$$

Obtaining the maximal value p and the associated S for which $F(n, p) < \infty$ gives the optimal solution to `GIK-SP`.

4.4.2 Discretization and analysis

As previously explained, due to the continuity of p, the above dynamic program is not algorithmic in nature. Similar to Sections 3.2.3.3 and 3.4.4, we restrict p to the finite set

$\mathcal{D}_p = \{d \cdot \frac{\epsilon p_{\max}}{2n} : d \in [\frac{2n^2}{\epsilon}]_0\}$. We let $\tilde{F}(i, p)$ to denote the dynamic program over the discretized set of states, whose recursive equations are identical to those of F. Let p^* be the maximal value satisfying $F(n, p^*) < \infty$. The next lemma shows that by restricting ourselves to values in $\mathcal{D}_p$, there exists state $\tilde{F}(n, \tilde{p}) < \infty$ achieving profit at least $(1 - \epsilon)p^*$. We omit the proof, as it is analogous to that of Lemma 3.2.6.

Lemma 4.4.3. *There exists $\tilde{p} \in \mathcal{D}_p$ such that $\tilde{p} \geq (1 - \epsilon) \cdot p^*$ and such that $\tilde{F}(n, \tilde{p}) < \infty$.*

By the above discretization, $\tilde{F}$ needs to be evaluated over $n \cdot |\mathcal{D}_p| = O(\frac{n^3}{\epsilon})$ number of states. Through the recursion of (4.10), each state can be evaluated in $O(1)$ time. Thus, the overall running time is $O(\frac{n^3}{\epsilon})$, concluding the proof of Theorem 1.3.6.

Chapter 5: Single-machine algorithms for incremental packing problems

5.1 Introduction

In this chapter, we show how formulating incremental packing problems as sequencing problems allows us to leverage on algorithms for single-bin problems to produce algorithms for general (multi-bin) packing problems. Recall in Section 3.2.1 we showed that the generalized incremental knapsack problem can be formulated as a single-machine sequencing problem. In this chapter, we extend the result to show that (multi-bin) incremental packing problems can be formulated as multi-machine sequencing problems. We will formally define multi-machine sequencing problems in Section 5.3. Broadly speaking, a multi-machine sequencing problem is a sequencing problem where multiple machines share the same ground set of items. The goal is to find a item-to-machine assignment and an optimal permutation for each machine on the the subset of items for which it is assigned. We explain how if there exists a constant-factor approximation for each single-machine sequencing problem, we can leverage it to also obtain a constant factor approximation for the multi-machine variant. Since such an algorithm uses as a black box the algorithm for the single-machine case, we call it a single-machine algorithm.

The remainder of the chapter is organized as follows. In Section 5.2 we give a general definition of incremental packing problems, and argue that the generalized incremental knapsack and the incremental generalized assignment problem can be seen as special cases of it. We then show how to formulate incremental packing problems as sequencing problems (see Section 5.3), and how to obtain approximate solution of the latter under certain conditions (see Theorem 5.3.1 and Section 5.4). We then apply this procedure to the incremental generalized assignment problem in Section 5.4.5 and prove Theorem 1.3.7, which we restate here for convenience.

Theorem 1.3.7. *For any $\epsilon > 0$, there exists a polynomial-time algorithm that in expectation gives*

a $(\frac{1}{4} - \epsilon)$*-approximation to the incremental generalized assignment problem.*

Our approach to translate algorithms from single- to multi-machines is inspired by the algorithm by [30] to obtain approximation algorithms for separable assignment problems. Our algorithm employs a different rounding scheme and applies to a more general class of problems, at the cost of worsening the guarantee on the approximation ratio. A comparison between the two approaches is discussed in Section 5.5.

5.2 Incremental packing problems

In this section, we give a formal definition of incremental packing problems. Starting from "static" packing problems, we first distinguish between single-bin and multi-bin problems. We then generalize them to incremental packing problems.

In a single-bin packing problem, we are given a ground set $[n]$ of items and a set of feasibility sets $\mathcal{F} \subseteq 2^{[n]}$. We say a set $S \subseteq [n]$ is feasible if $S \in \mathcal{F}$. We further assume that $\mathcal{F}$ is an independence set: $\emptyset \in \mathcal{F}$ and for $S \in \mathcal{F}$, we have that $X \subseteq S$ implies $X \in \mathcal{F}$. For each item $i \in [n]$, we are given a profit $p_i \in \mathbb{N}$. The goal is to find a feasible set $S \in \mathcal{F}$ that maximizes $p(S)$. We let the tuple $(n, \mathcal{F}, p)$ describe an instance of a single-bin packing problem. It is easy to see that the classical knapsack problem is an example of a single-bin packing problem, as well as other standard problems like matching, independent set, etc.

(Multi-bin) packing problems are a generalization of single-bin packing problems. In this context, we are given m single-bin packing problems $(n, \mathcal{F}_1, p_1), (n, \mathcal{F}_2, p_2), \ldots, (n, \mathcal{F}_m, p_m)$ sharing the same ground set $[n]$ of items. A feasible solution is a collection of pairwise-disjoint sets $S_1, \ldots, S_m$ with $S_\ell \in \mathcal{F}_\ell$ for each $\ell \in [m]$. The goal is to find a feasible solution that maximizes $\sum_{\ell \in [m]} p_\ell(S_\ell)$. We denote an instance of a packing problem by the tuple $(m, n, \{\mathcal{F}_\ell\}_{\ell=1,\ldots,m}, \{p_\ell\}_{\ell=1,\ldots,m})$. It is easy to see that the generalized assignment problem is an example of a packing problem. To distinguish between "static" packing problems and their incremental extensions – to be described next – we call the former classical packing problems.

In an incremental packing problem, we are given a collection of T time periods, a ground

set $[n]$ of items, and a set $[m]$ of bins. For each $t \in [T]$ and $\ell \in [m]$, we are given a set of feasible sets $\mathcal{F}_{\ell,t} \in 2^{[n]}$. For each $i \in [n]$, $\ell \in [m]$ and $t \in [T]$, we are given a profit $p_{i,\ell,t} \in \mathbb{N}$. A solution is in the form of m ordered families, one for each $\ell \in [m]$: $\mathcal{S}_\ell = (S_{\ell,1}, \ldots, S_{\ell,T})$. $\mathcal{S}_\ell$ is required to satisfy the following properties. First, it is a chain, that is, $S_{\ell,1} \subseteq S_{\ell,2} \subseteq \cdots \subseteq S_{\ell,T} \subseteq [n]$. Second, it is feasible, i.e., $S_{\ell,t} \in \mathcal{F}_{\ell,t}$ for every $\ell \in [m]$ and $t \in [T]$. Finally, different chains consist of sets of items that are pairwise-disjoint, i.e., for any distinct pair $\ell_1, \ell_2 \in [m]$ we have $S_{\ell_1,T} \cap S_{\ell_2,T} = \emptyset$. The goal is to find m feasible chains $\mathcal{S}_1, \ldots, \mathcal{S}_m$ that maximizes $\sum_{\ell \in [m]} \Phi_\ell(\mathcal{S}_\ell) = \sum_{\ell \in [m]} \sum_{t \in T} p_{\ell,t}(S_{\ell,t} \setminus S_{\ell,t-1})$,where again we assume $S_{\ell,0} = \emptyset$ for all $t \in [T]$. An incremental packing problem is therefore succinctly described by a tuple $\mathcal{I} = (m, n, T, \{\mathcal{F}_{\ell,t}\}_{\ell \in [m], t \in [T]}, \{p_{\ell,t}\}_{\ell \in [m], t \in [T]})$.

Note that the incremental packing problem $\mathcal{I}$ can be "sliced" across periods or across bins. In the former case, we obtain T classical packing problems: for each $t \in [T]$, we say that $(m, n, \{\mathcal{F}_{\ell,t}\}_{\ell \in [m]}, \{p_{\ell,t}\}_{\ell \in [m]})$ is a classical packing problem *time-underlying* $\mathcal{I}$. In the latter case, we obtain m incremental packing problems, each defined over a single bin: $(1, n, T, \{\mathcal{F}_{\ell,t}\}_{t \in [T]}, \{p_{\ell,t}\}_{t \in [T]})$ for $\ell \in [m]$. We say that each of those problems is *bin-underlying* $\mathcal{I}$.

It is easily observed that the generalized incremental knapsack problem is an incremental packing problem, where each time-underlying classical packing problem is an instance of the classical knapsack problem. By its definition in Section 1.2.3, one readily observes that the incremental generalized assignment problem is also an incremental packing problem, where each time-underlying classical packing problem is an instance of the generalized assignment problem.

5.3 Sequencing reformulation of incremental packing problems

Single-machine sequencing. A (*single-machine*) *sequencing problem with inclusionwise non-increasing profit profile* is defined by a set $[n]$ of feasible items, a number $R \in \mathbb{N}$, and, for each $S \subseteq [n]$ and π from the set Π_S of all permutations of S, a profit function $\varphi_\pi(i) : [n] \to \{0, 1, \ldots, R\}$ such that, for every $\pi, \pi' \in \Pi_S$, the following holds:

1. For any $S \subseteq [n]$, $\pi \in \Pi_S$, and $i \in [n] \setminus S$, we have $\varphi_\pi(i) = 0$.

2. If for some $S, S' \subseteq [n]$, $i \in S \cap S'$, $\pi \in \Pi_S, \pi' \in \Pi_{S'}$, we have $\{j : \pi(j) \leq \pi(i)\} \subseteq \{j : \pi'(j) \leq \pi'(i)\}$, then $\varphi_\pi(i) \geq \varphi_{\pi'}(i)$.

We assume that φ is given via an oracle, i.e., given $S \subseteq [n]$, $\pi \in \Pi_S$ and $i \in [n]$ we can compute $\varphi_\pi(i)$ in polynomial time. The profit function $\Psi : \cup_S \Pi_S \to \mathbb{Q}$ is given by $\Psi(\pi) = \sum_{i \in [n]} \varphi_\pi(i)$, and the goal is to find $S \subseteq [n], \pi \in \Pi_S$ maximizing $\Psi(\pi)$.

Notice that, in particular, from Section 3.2.1, the generalized incremental knapsack problem can be reformulated as a sequencing problem that satisfies the inclusionwise non-increasing profit profile. Indeed, all the sequencing problems considered in this chapter have inclusionwise non-increasing profit profile, so for the sake of readability we will refer to them simply as sequencing problems.

Let $\mathcal{C}$ be a family of sequencing problems. We call $\mathcal{C}$ *closed under profit shifting* (or, in short, *closed*) if, for every instance $\mathcal{I} \in \mathcal{C}$ with n items and every $\lambda \in \mathbb{R}^n$, the sequencing problem obtained by replacing, for each $S \subseteq [n]$, $\pi \in \Pi_S$, and $i \in [n]$, $\varphi_\pi(i)$ by $\max\{0, \varphi_\pi(i) - \lambda_i\}$ also belongs to $\mathcal{C}$.

Multi-machine sequencing. A *multi-machine sequencing problem* $\mathcal{I}$ is defined by a family $\mathcal{I}_1, \ldots, \mathcal{I}_m$ of m sequencing problems with profit functions $\Psi_1, \ldots, \Psi_m$ respectively, each defined over the same set $[n]$ of items and bound R. Let $\mathcal{P}$ be the family of ordered m-tuples of pairwise non-intersecting subsets of $[n]$. The goal is to compute

$$\max_{(S_1,\ldots,S_m) \in \mathcal{P}} \sum_{j=1}^{m} \max_{\pi \in \Pi_{S_j}} \Psi_j(\pi). \tag{5.1}$$

We call $\mathcal{I}_1, \ldots, \mathcal{I}_m$ the single-machine sequencing problems underlying $\mathcal{I}$. The main technical tool of the section is the following result, proved in Section 5.4.

Theorem 5.3.1. *Consider a family $\mathcal{C}$ of multi-machine sequencing problems. Assume that, for each $\mathcal{I} \in \mathcal{C}$, the following happens: there is a closed family $\mathcal{C}'$ of single-machine sequencing problems*

containing all single-machine problems underlying $\mathcal{I}$ and a polynomial-time β-approximation algorithm for problems in $\mathcal{C}'$, for some $\beta \in (0, 1]$. Then, for any $\delta > 0$, there is a polynomial-time algorithm that in expectation gives a $\frac{1}{2}(\beta - \delta)$-approximation to instances from $\mathcal{C}$.

From packing to sequencing. We argue next that sequencing (resp., multi-machine sequencing) problems subsume single-bin incremental packing (resp., incremental packing) problems. These results extend (following the same proof idea) those from Lemma 3.2.1, where the generalized incremental knapsack is formulated as a sequencing problem. Moreover, under this transformation, packing problems with the same feasibility set are mapped to closed families of sequencing problems. The proof of the following lemma is given in Appendix E.1.

Lemma 5.3.2. *Let $\mathcal{I} = (m, n, T, \{\mathcal{F}_{\ell,t}\}_{\ell\in[m],t\in[T]}, \{p_{\ell,t}\}_{\ell\in[m],t\in[T]})$ be an instance of an incremental packing problem.*

1. *The problem can be reformulated as an instance $\mathcal{I}'$ of a multi-machine sequencing problem in time polynomial in the size of $\mathcal{I}$ with the following features: any feasible chains $\mathcal{S}_1, \ldots, \mathcal{S}_m$ to the incremental packing problem can be mapped to feasible permutations $\pi_1, \ldots, \pi_m$ to the multi-machine sequencing problem such that $\sum_{j\in[m]} \Psi_j(\pi_j) \geq \sum_{j\in[m]} \Phi_j(\mathcal{S}_j)$; conversely, any feasible permutations $\pi_1, \ldots, \pi_m$ to the multi-machine sequencing problem can be mapped to feasible chains $\mathcal{S}_1, \ldots, \mathcal{S}_m$ such that $\sum_{j\in[m]} \Phi_j(\mathcal{S}_j) = \sum_{j\in[m]} \Psi_j(\pi_j)$. Moreover, from an optimal solution to $\mathcal{I}'$ we can obtain in time polynomial in the size of $\mathcal{I}$ an optimal solution to $\mathcal{I}$.*

2. *In particular, if $m = 1$, then $\mathcal{I}'$ is a single-machine sequencing problem.*

3. *Let $n \in \mathbb{N}$ and $(\mathcal{F}_1, \ldots, \mathcal{F}_T)$ be a feasibility set for a single-bin packing problem on n items. The family of single-machine scheduling problems obtained from all single-bin packing problems with feasibility set $(\mathcal{F}_1, \ldots, \mathcal{F}_T)$ via the reduction from Point 1 above is closed.*

In the lemma below, we illustrate an application of Lemma 5.3.2 by showing, in particular, how the incremental generalized assignment problem can be reformulated as a multi-machine se-

quencing problem. Its proof can be found in Appendix E.2. We then give an example of the reformulation.

Lemma 5.3.3. *An incremental generalized assignment problem $\mathcal{I}$ can be reformulated as a multi-machine sequencing problem $\mathcal{I}'$, where each single-machine sequencing problem underlying $\mathcal{I}'$ is obtained from a generalized incremental knapsack problem bin-underlying $\mathcal{I}$ via the reduction from Lemma 5.3.2. Moreover, the family of single-machine sequencing problems obtained as above is closed.*

5.3.1 Example

Consider the following incremental generalized assignment problem instance $\mathcal{I}$ consisting of 5 items, 2 bins, and 2 time periods. Bin 1 has capacities $[10, 15]^T$, where the first component denotes the capacity at time 1 and the second component denotes the capacity at time 2. Bin 2 has capacities $[5, 8]^T$. Items have weights given by the matrix w below, where each entry $w_{i,\ell}$ is the weight of item i when assigned to bin ℓ. Items have profits given by matrices $M(p)^1$ and $M(p)^2$, where in the first matrix (resp. second matrix), each entry $M(p)^1_{i,t}$ (resp. $M(p)^2_{i,t}$) gives the profit of item i when it is first inserted into bin 1 (resp. bin 2) at time t.

$$w = \begin{bmatrix} 5 & 3 \\ 6 & 5 \\ 3 & 4 \\ 10 & 3 \\ 10 & 5 \end{bmatrix}; \quad M(p)^1 = \begin{bmatrix} 10 & 5 \\ 8 & 15 \\ 2 & 5 \\ 5 & 2 \\ 2 & 5 \end{bmatrix}; \quad M(p)^2 = \begin{bmatrix} 3 & 6 \\ 5 & 10 \\ 1 & 3 \\ 5 & 1 \\ 1 & 5 \end{bmatrix}.$$

We next discuss the generalized incremental knapsack problems $\mathcal{I}_1$ and $\mathcal{I}_2$ bin-underlying $\mathcal{I}$. Namely, $\mathcal{I}_1$ has capacities $W_1 = [10, 15]^T$, matching the capacities of bin 1 in the incremental generalized assignment problem. Similarly, $\mathcal{I}_2$ has capacities $W_2 = [5, 8]^T$, matching the capacities of bin 1 in the incremental generalized assignment problem. Both instances $\mathcal{I}_1$ and $\mathcal{I}_2$ are defined over the same set of 5 items. Assigning item i to instance $\mathcal{I}_j$ at time t takes up capacity $w_{i,j}$ and

earns profit $p_{i,j,t}$.

Consider the feasible solution to the incremental generalized assignment problem

$$S_{1,1} = \{1\}, S_{1,2} = \{1,2,3\}; S_{2,1} = \{4\}, S_{2,2} = \{4,5\}.$$

One easily observes that this translates to feasible solutions $(S_{1,1}, S_{1,2})$ for $\mathcal{I}_1$ and $(S_{2,1}, S_{2,2})$ for $\mathcal{I}_2$. Moreover, the combined profit of these two solutions is equivalent to the profit of the solution to the incremental generalized assignment problem.

We next demonstrate how to convert some solution to each of the generalized incremental knapsack problems $\mathcal{I}_1, \mathcal{I}_2$ to solutions to the a single-machine sequencing problems obtained using Lemma 5.3.3. For $\mathcal{I}_1$, the solution $\mathcal{S}_1 = (\{1\}, \{1,2,3\})$ translates to the permutation $\pi_1 = [1,2,3]$, where $\varphi_{1,\pi_1}(1) = \max\{10,5\} = 10$, $\varphi_{1,\pi_1}(2) = 15$ and $\varphi_{1,\pi_1}(3) = 5$. Therefore, $\Psi_1(\pi_1) = \sum_{i=1}^{3} \varphi_{1,\pi_1}(i) = 30$. For $\mathcal{I}_2$, the solution $\mathcal{S}_2 = (\{4\}, \{4,5\})$ translates to the permutation $\pi_2 = [4,5]$, where $\varphi_{2,\pi_2}(4) = \max\{5,1\} = 5$ and $\varphi_{2,\pi_2}(5) = 5$. Similarly, $\Psi_2(\pi_2) = \sum_{i=4}^{5} \varphi_{2,\pi_2}(i) = 10$.

Finally, observe that

$$\begin{aligned}
\Phi_1(\mathcal{S}_1) + \Phi_2(\mathcal{S}_2) &= p_{1,1}(S_{1,1}) + p_{1,2}(S_{1,2}) + p_{2,1}(S_{2,1}) + p_{2,2}(S_{2,2}) \\
&= 10 + 20 + 5 + 5 \\
&= 30 + 10 \\
&= \Psi_1(\pi_1) + \Psi_2(\pi_2).
\end{aligned}$$

5.4 Ex uno plures: approximation algorithms to multi-machine problems

In this section, we prove Theorem 5.3.1. The proof proceeds as follows. In Section 5.4.1, we give an IP formulation for a multi-machine sequencing problem with exponentially many variables, and show how to obtain an approximate separation algorithm for the dual of its LP relaxation from an approximation algorithm for the single machine sequencing problem. This is used then in

Section 5.4.2 to produce an approximate solution to the LP relaxation, which is rounded in Section 5.4.3 to a feasible solution to our multi-machine sequencing problem. Finally, Section 5.4.4 ties together these results to prove Theorem 5.3.1.

5.4.1 LP relaxation and approximate dual separation

Suppose we are given a multi-machine sequencing problem $\mathcal{I}$ over a set of n items, whose underlying m single-machine sequencing problems $\mathcal{I}_1, \ldots, \mathcal{I}_m$ have objective functions $\Psi_1(\cdot), \ldots, \Psi_m(\cdot)$, respectively. For every $j \in [m], S \in 2^{[n]}$, $\pi \in \Pi_S$, let X_j^π be the indicator variable that denotes if we assign set S to machine j with the permutation π. The following is an integer programming formulation for the multi-machine sequencing problem:

$$\begin{aligned}
\max \quad & \sum_{j\in[m]} \sum_{S\in 2^{[n]}} \sum_{\pi\in\Pi_S} \Psi_j(\pi) X_j^\pi & & \\
\text{s.t.} \quad & \sum_{j\in[m]} \sum_{S\in 2^{[n]}: i\in S} \sum_{\pi\in\Pi_S} X_j^\pi & \leq 1 \qquad & \forall\, i \in [n] \\
& \sum_{S\in 2^{[n]}} \sum_{\pi\in\Pi_S} X_j^\pi & = 1 \qquad & \forall\, j \in [m] \\
& X_j^\pi \in \{0,1\} & & \forall\, j \in [m],\, S \in 2^{[n]},\, \pi \in \Pi_S
\end{aligned} \tag{MMS-IP}$$

We denote by (MMS-LP) the linear programming relaxation of (MMS-IP) obtained by replacing the binary constraints with nonnegativity constraints. (MMS-LP) has the following associated dual program:

$$\begin{aligned}
\min \quad & \sum_{j\in[m]} q_j + \sum_{i\in[n]} \lambda_i & & \\
\text{s.t.} \quad & q_j + \sum_{i\in S} \lambda_i & \geq \Psi_j(\pi) \qquad & \forall\, j \in [m],\, S \in 2^{[n]},\, \pi \in \Pi_S \\
& \lambda_i & \geq 0 \qquad & \forall\, i \in [n].
\end{aligned} \tag{MMS-DP}$$

Rewriting the dual program as a fractional covering problem, we obtain:

$$\begin{aligned} \min \quad & \sum_{j\in[m]} q_j + \sum_{i\in[n]} \lambda_i \\ \text{s.t.} \quad & (q_j, \lambda) \in \mathcal{P}_j && \forall\, j \in [m] \\ & \lambda_i \geq 0 && \forall\, i \in [n], \end{aligned}$$

where $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_n)^T$ and $\mathcal{P}_j$ is the polyhedron defined by constraints $q_j \geq \Psi_j(\pi) - \sum_{i\in S} \lambda_i$ for all $S \in 2^{[n]}$ and $\pi \in \Pi_S$.

A *β-approximate separation* for $\mathcal{P}_j$ is an algorithm that takes as input a point $(q_j, \lambda) \in \mathbb{Q}^{1+n}$ and either returns a violated constraint or guarantees that $(\frac{q_j}{\beta}, \lambda) \in \mathcal{P}_j$. In the remainder of the section, we prove the following:

Lemma 5.4.1. *Let $j \in [m]$. Suppose there is a polynomial-time algorithm that gives a β-approximation to a closed class C of single-machine sequencing problems containing $\mathcal{I}_j$. Then, there exists a β-approximate separation for $\mathcal{P}_j$ that runs in polynomial time.*

Proof. We employ the β-approximation algorithm for the sequencing problem as follows. First, we define a sequencing problem instance $\mathcal{I}'_j$ that is obtained from $\mathcal{I}_j$ by setting $\varphi'_{j,\pi}(i) = \max\{0, \varphi_{j,\pi}(i) - \lambda_i\}$, and then normalizing vector $\{\varphi'_{j,\pi}(i)\}_{j,\pi}$ so that it is integral. Clearly, φ'_j satisfies Property 1 from the definition of sequencing problems. Moreover, φ'_j also satisfies Property 2. Indeed, let $i \in [n]$ and π, π' be permutations of subsets of $[n]$ so that i is in the domain of both π and π', and $\{\ell : \pi(\ell) \leq \pi(i)\} \subseteq \{\ell : \pi'(\ell) \leq \pi'(i)\}$. If $\varphi_{j,\pi'}(i) - \lambda_i \leq 0$, then $\varphi'_{j,\pi}(i) \geq 0 = \varphi'_{j,\pi'}(i)$. Else,

$$\varphi'_{j,\pi}(i) \geq \varphi_{j,\pi}(i) - \lambda_i \geq \varphi_{j,\pi'}(i) - \lambda_i = \varphi'_{j,\pi'}(i).$$

Picking an appropriate R' whose encoding length is polynomial in the encoding length of R and λ, we deduce that $\mathcal{I}'_j$ is a well-defined single-machine scheduling problem whose input is of size polynomial in the original input size. Finally, notice that since C is a closed class of sequencing

problems and we have a β-approximate algorithm for $\mathcal{I}_j$, we also have a β-approximate algorithm for $\mathcal{I}'_j$.

For every $S \subseteq [n]$ and permutation $\pi \in \Pi_S$, let $\Psi'_j(\pi) = \sum_{i \in S} \varphi'_{j,\pi}(i)$. Let $\overline{\pi}$ be the permutation output by the β-approximated algorithm on $\mathcal{I}'_j$. We create S by removing from $[n]$ all items i such that $\varphi'_{j,\overline{\pi}}(i) = 0$. Let $\overline{\pi}_S$ be the permutation for the items in S following their internal ordering in $\overline{\pi}$. Observe that, by Property 2, we have $\varphi'_{j,\overline{\pi}_S}(i) \geq \varphi'_{j,\overline{\pi}}(i)$ for every $i \in S$. Let $q^*_j = \Psi'_j(\overline{\pi}_S)$.

For any $S' \subseteq [n]$ and any possible permutation $\pi' \in \Pi_{S'}$, we have:

$$\begin{aligned} q^*_j &= \Psi'_j(\overline{\pi}_S) \\ &\geq \Psi'_j(\overline{\pi}) \\ &\geq \beta\Psi'_j(\pi') \\ &\geq \beta \sum_{i \in S'} (\varphi_{j,\pi'(i)} - \lambda_i) \\ &= \beta(\Psi_j(\pi') - \sum_{i \in S'} \lambda_i). \end{aligned}$$

where the first equality follows from the definition of q^*_j, the first inequality follows since we only remove i such that $\Psi'_{j,\bar{\pi}} = 0$ and by Property 2. The second inequality follows since $\bar{\pi}$ is obtained by the β-approximated algorithm, the final inequality follows by definition of Ψ'_j, and the final equality follows by definition of Ψ_j. Thus, if $q^*_j \leq q_j$, we have:

$$q_j \geq q^*_j \geq \beta(\Psi_j(\pi') - \sum_{i \in S'} \lambda_i)$$

for all $S' \subseteq [n]$ and permutations $\pi' \in \Pi_{S'}$. Therefore $\frac{q_j}{\beta} \geq \Psi_j(\pi_{S'}) - \sum_{i \in S'} \lambda_i$, and we conclude that $(\frac{q_j}{\beta}, \lambda) \in \mathcal{P}_j$. Else, we have:

$$\begin{aligned} q_j &< q^*_j \\ &= \sum_{i \in S} (\varphi_{j,\overline{\pi}_S}(i) - \lambda_i) \end{aligned}$$

$$= \quad \Psi_j(\overline{\pi}_S) - \sum_{i \in S} \lambda_i,$$

where the first equality follows from the assumption that $\varphi'_{j,\overline{\pi}_S}(i) > 0$ for all $i \in S$. Thus we have a violated inequality, as desired. □

5.4.2 Approximate primal solution from approximate dual separation

In this section, we build upon the β-approximate dual separation routine from the previous section and show how it can be employed to obtain a $(\beta - \delta)$-approximate primal solution to (MMS-LP), for every $\delta > 0$. The proof is similar to the analogous statement in [30], hence we defer it to Appendix E.3.

Lemma 5.4.2. *Given a β-approximate separation algorithm for $\mathcal{P}_1, \ldots, \mathcal{P}_m$, we can obtain a $(\beta - \delta)$-approximate primal solution to* (MMS-LP) in polynomial time for any $\delta > 0$.

5.4.3 The rounding procedure

We next illustrate how to round a primal solution for (MMS-LP) to a feasible integer solution by losing at most $\frac{1}{2}$ of the profit in expectation.

Lemma 5.4.3. *Let $\hat{X}$ be a feasible solution to* (MMS-LP). $\hat{X}$ can be rounded in polynomial time to a feasible solution $\overline{X}$ to (MMS-IP) such that

$$\sum_{j \in [m]} \sum_{S \in 2^{[n]}} \sum_{\pi \in \Pi_S} \Psi_j(\pi) \mathbb{E}[\overline{X}_j^{\pi}] \geq \frac{1}{2} \sum_{j \in [m]} \sum_{S \in 2^{[n]}} \sum_{\pi \in \Pi_S} \Psi_j(\pi) \hat{X}_j^{\pi}.$$

Proof. First round $\hat{X}$ independently for every $j \in [m]$. We assign to $\mathcal{I}_j$ at most one $S \subseteq 2^{[n]}$ and $\pi_S \in \Pi_S$, by sampling each π_S with probability $p_{\pi_S} = \hat{X}_j^{\pi_S}$, and to the empty set with the remaining probability. Because of the second constraint of (MMS-LP), this procedure is well-defined. Let $\widetilde{X}$ be the binary vector corresponding to this integer assignment. $\widetilde{X}$ may be infeasible for (MMS-IP), since there may exist $j_1 \neq j_2 \in [m]$ such that $\widetilde{X}_{j_1}^{\pi_{S_1}} = 1$, $\widetilde{X}_{j_2}^{\pi_{S_2}} = 1$ and $S_1 \cap S_2 \neq \emptyset$.

To fix this feasibility issue, for every $i \in [n]$, let

$$M(i) = \{j \in [m] : \widetilde{X}_j^{\pi_S} = 1, \text{ for some } S \subseteq [n] \text{ such that } i \in S\},$$

so that $M(i)$ denotes the set of machines that are assigned to some set S that contains item i. For every item $i \in [n]$, assign item i uniformly at random to one set $j \in M(i)$, and remove item i from the remaining sets. Let $\overline{M}(i)$ denote the machine j to which item i is assigned, and $\overline{X}$ denote the binary vector corresponding to this integer assignment. Note that $\overline{X}$ is feasible for (MMS-IP).

Let us compare the objective function values of $\overline{X}$ and $\hat{X}$. For every $i \in [n]$, $j \in [m]$, $S \subseteq [n]$, and $\pi \in \Pi_S$,

$$\begin{aligned}
\mathbb{E}\left[\overline{X}_j^{\pi}\right] &\geq \mathbb{E}\left[\widetilde{X}_j^{\pi_S} \cdot \mathbb{1}[\overline{M}(i) = j]\right] \\
&= \mathbb{P}[\widetilde{X}_j^{\pi} = 1] \cdot \mathbb{P}[\overline{M}(i) = j | \widetilde{X}_j^{\pi} = 1] \\
&= \hat{X}_j^{\pi} \cdot \mathbb{P}[\overline{M}(i) = j | \widetilde{X}_j^{\pi} = 1]. \qquad (5.2)
\end{aligned}$$

For every $j \in [m]$ and $i \in [n]$, let $K_j^i = |M(i) \setminus \{j\}|$. Following this definition,

$$\begin{aligned}
\mathbb{P}[\overline{M}(i) = j | \widetilde{X}_j^{\pi} = 1] &= \sum_{k=0}^{m-1} \mathbb{P}[K_j^i = k | \widetilde{X}_j^{\pi} = 1] \mathbb{P}[\overline{M}(i) = j | \widetilde{X}_j^{\pi} = 1 \wedge K_j^i = k] \\
&= \sum_{k=0}^{m-1} \mathbb{P}[K_j^i = k] \cdot \frac{1}{k+1} \\
&= \mathbb{E}\left[\frac{1}{K_j^i + 1}\right] \\
&\geq \frac{1}{\mathbb{E}[K_j^i] + 1} \\
&\geq \frac{1}{2}, \qquad (5.3)
\end{aligned}$$

where, in the second equality, we used that $\mathbb{P}[K_j^i = k | \widetilde{X}_j^{\pi} = 1] = \mathbb{P}[K_j^i = k]$ since the number of machines to which item i is assigned not counting machine j is independent with the event that

item i is assigned to machine j, and $\mathbb{P}[\overline{M}(i) = j | \widetilde{X}_j^\pi = 1 \wedge K_j^i = k] = \frac{1}{k+1}$ since item i is assigned to machine j uniformly at random among $k+1$ machines; the first inequality follows by applying Jensen's inequality and by noting that $\frac{1}{K_j^i+1}$ is a convex function for $K_j^i \geq 0$; the final inequality follows since

$$\mathbb{E}[K_j^i] = \sum_{j' \in [m]} \sum_{S \in 2^{[n]}: i \in S} \sum_{\pi \in \Pi_S} \hat{X}_{j'}^\pi - \sum_{S \in 2^{[n]}: i \in S} \sum_{\pi \in \Pi_S} \hat{X}_j^\pi \leq 1 - \sum_{S \in 2^{[n]}: i \in S} \sum_{\pi \in \Pi_S} \hat{X}_j^\pi \leq 1.$$

Plugging (5.3) into (5.2), we get

$$\mathbb{E}\left[\overline{X}_j^\pi\right] \geq \frac{1}{2} \cdot \hat{X}_j^\pi.$$

Finally, summing over all $i \in [n]$, $j \in [m]$ and permutations $\pi \in \Pi_S$ where $S \in 2^{[n]}$, we have:

$$\sum_{j \in [m]} \sum_{i \in [n]} \sum_{S \in 2^{[n]}} \sum_{\pi \in \Pi_S} \varphi_{j,\pi}(i) \mathbb{E}\left[\overline{X}_j^\pi\right] \geq \frac{1}{2} \sum_{j \in [m]} \sum_{i \in [n]} \sum_{S \in 2^{[n]}} \sum_{\pi \in \Pi_S} \varphi_{j,\pi}(i) \hat{X}_j^\pi.$$

Noting that $\sum_{i \in [n]} \varphi_{j,\pi}(i) = \Psi_j(\pi)$ for all $\pi \in \Pi_S$, $S \in 2^{[n]}$ completes the proof. □

5.4.4 Proof of Theorem 5.3.1

Using the results proved thus far, we can deduce the proof of Theorem 5.3.1. Let $\hat{X}$ be the $(\beta - \delta)$-approximate primal solution obtained via Lemma 5.4.2. $\hat{X}$ can be rounded to a feasible integer solution $\overline{X}$ as guaranteed by Lemma 5.4.3. Let OPT(MMS-LP) and OPT(MMS-IP) denote the optimal objective function value of (MMS-LP) and (MMS-IP) respectively. We have:

$$\begin{aligned}
\sum_{j \in [m]} \sum_{S \in 2^{[n]}} \sum_{\pi \in \Pi_S} \Psi_j(\pi) \mathbb{E}[\overline{X}_j^\pi] &\geq \frac{1}{2} \sum_{j \in [m]} \sum_{S \in 2^{[n]}} \sum_{\pi \in \Pi_S} \Psi_j(\pi) \hat{X}_j^\pi \\
&\geq \frac{1}{2} \cdot (\beta - \delta) \cdot \text{OPT(MMS-LP)} \\
&\geq \frac{1}{2} \cdot (\beta - \delta) \cdot \text{OPT(MMS-IP)}.
\end{aligned}$$

Here, the first inequality follows from Lemma 5.4.3, the second inequality follows from Lemma 5.4.2. The final inequality follows since (MMS-LP) is the linear relaxation of (MMS-IP). Observing that (MMS-IP) is an integer programming formulation of the multi-machine sequencing problem concludes the proof.

5.4.5 Proof of Theorem 1.3.7

By Lemma 5.3.3, the incremental generalized assignment problem can be reformulated as a multi-machine sequencing problem, where each single-machine sequencing problem is a reformulation of a generalized incremental knapsack problem, and moreover, the family of single-machine sequencing problems one obtains is closed. Section 3.2 gives a $\frac{1}{2} - \epsilon$ approximated algorithm for the generalized incremental knapsack problem in polynomial time. Thus, Theorem 1.3.7 is a direct result of applying Theorem 5.3.1, together with the $\frac{1}{2} - \epsilon$ approximated algorithm for the generalized incremental knapsack problem.

5.5 Comparison with the approach by Fleischer et al.

As previously mentioned, our techniques to solve the multi-machine sequencing problem are inspired by [30]. In this section, we discuss the extent of the similarities, as well as the differences in the two problems and the techniques employed.

Separable assignment vs. multi-machine sequencing. [30] consider the separable assignment problem, where we are given n items and m machines. Assigning item i to machine j earns profit $p_{i,j}$. Each machine j has a feasibility set $\mathcal{F}_j \subseteq 2^{[n]}$ that is an independence system. The goal is to find a feasible item to machine assignment that earns maximum profit.

Notice that the generalized assignment problem can be formulated as a separable assignment problem. In turn, the separable assignment problem can be formulated as a multi-machine sequencing problem over the same set $[n]$ of items. In particular, given a set $S \subseteq [n]$ and $\pi \in \Pi_S$, let $\varphi_{j,\pi}(i) = p_{i,j}$ if $\{\ell : \pi(\ell) \leq \pi(i)\} \subseteq \mathcal{F}_j$. That is, if the set of all items that come before item

i, including item i itself, are feasible for machine j, we let $\varphi_{j,\pi}(i) = p_{i,j}$. Else, let $\varphi_{j,\pi}(i) = 0$. One easily checks that, since $\mathcal{F}_j$ is an independence system, the properties of inclusionwise non-increasing profit profile are satisfied. Furthermore, since the separable assignment problem is a packing problem, by Point 3 of Lemma 5.3.2, the family of single-machine sequencing problems obtained from separation assignment problems via this reduction is closed.

Let $S_1, \ldots, S_m$ be a feasible solution to the separable assignment problem, where S_j denotes the set of items assigned to machine j. Since $S_j \in \mathcal{F}_j$, for any permutation $\pi_j \in \Pi_{S_j}$, clearly $\Psi_j(\pi_j) = \sum_{i \in S_j} p_{i,j}$. On the other hand, let $\pi_1, \ldots, \pi_m$ be a feasible solution to the multi-machine sequencing problem. Removing all items i from S_j such that $\varphi_{j,\pi_j}(i) = 0$ gives a feasible solution to the separable assignment problem with equivalent profit.

The key difference between the separable assignment problem and the multi-machine sequencing problem lies in the fact that in the latter setting, the profit an item earns depends not only on the item and the machine it is assigned to, but also on other items assigned to the same machine as well as their internal permutation; whereas in the separable assignment problem, the profit an item earns is only dependent on the machine to which it is assigned (as long as the set of assigned item is feasible). Hence, the multi-machine sequencing problem is strictly more general than the separable assignment problem.

As such, to solve the multi-machine sequencing problem, we need to develop a new IP formulation and a new rounding procedure to obtain a feasible integer solution from a feasible fractional solution. The IP formulation in [30] consists of indicator variables X_j^S to indicate if machine j is assigned set S. Alternatively, in our IP formulation, we use indicator variables $X_j^{\pi_S}$ to denote if machine j is assigned set S with permutation π_S. With this added granularity of the permutation of the item set, the profit $\Psi_j(\pi_S)$ becomes well-defined and obtainable in polynomial time.

Comparison of the rounding procedures. It is also worthwhile to highlight the differences between the rounding procedures. Given a feasible LP solution $\hat{X}_j^S$, [30] obtain a feasible integral solution $\overline{X}_j^S$ as follows: first, it assigns each machine j set S with probability $\overline{X}_j^S$. Then, following

this rounding, if an item is assigned to multiple machines, the item is assigned to the most profitable machine. Similar to [30], in our rounding procedure, we assign a machine j permutation π_S with probability $\hat{X}_j^{\pi_S}$. As a second step, to address the same infeasibility issue of an item being assigned to multiple machines, we instead assign an item to one of these machines uniformly at random. To bound the expected profit of the feasible IP solution against the profit of the feasible LP solution, the proof proposed by [30] relies on the fact that for every item i, there is a well-defined ordering of machines in non-decreasing profit order, independent of other items that are assigned to that machine. Using this order, they explicitly calculate the probability an item i is assigned to machine j in their procedure and achieve a lower bound of $1 - \frac{1}{e}$. As previously mentioned, this ordering does not exist in our setting. Thus, it is unclear if similar techniques can be utilized in our setting to achieve a similar lower bound. Instead, we use our different rounding procedure that allows us to calculate the probability an item i is assigned to machine j, and achieve a worse lower bound of $\frac{1}{2}$.

Chapter 6: On the facets of the incremental knapsack polytope

6.1 Introduction

In this chapter, we present polyhedral results for the incremental knapsack polytope. In particular, we build upon the ideas of cover inequalities for the classical knapsack polytope and strengthen these inequalities to define valid inequalities for the incremental knapsack polytope. Our main contribution is the extension of the incremental knapsack polytope of one direction of an "if and only if" result by Balas [6] on facet-defining inequalities of the knapsack polytope with coefficients 0 and 1.

The remainder of the chapter is organized as follows. In Section 6.2, we give a brief summary of cover inequalities for the classical knapsack polytope and necessary and sufficient conditions for when they define a facet. In Section 6.3, we show how, starting with a classic knapsack cover inequality, we can strengthen it to obtain a stronger valid inequality for the incremental knapsack polytope. We dub this class of inequalities the *lift and push cover inequalities*. We show that, interestingly, under the same conditions in which cover inequalities define a facet for the classical knapsack polytope, the inequalities our procedure obtains define a facet for the incremental knapsack polytope. These conditions, however, are not necessary. In Section 6.4, we give separation algorithms for lift and push cover inequalities. We provide two algorithms, the first of which is an exact separation procedure that runs in pseudo-polynomial time. Finally, for any $\epsilon > 0$, we present a polynomial time, $(1 - \epsilon)$-approximate separation algorithm.

6.2 Cover inequalities for the classical knapsack polytope

In this section, we give an overview of $0/1$ facets of the knapsack polytope [6]. Given the standard integer programming formulation of the classical knapsack problem

$$\begin{aligned} \max \quad & \sum_{i\in[n]} p_i x_i \\ \text{s.t.} \quad & \sum_{i\in[n]} w_i x_i \le W \\ & x_i \in \{0,1\} \qquad \forall\, i \in [n], \end{aligned} \tag{KP}$$

the knapsack polytope is defined as the convex hull of the feasible points satisfying the constraints given above. Let $S \subseteq [n]$. Let $j_1, j_2, \ldots, j_{|S|}$ be an order of distinct indices such that $j_k \in S$ for each $k \in [|S|]$ and $w_{j_1} \ge w_{j_2} \ge \cdots \ge w_{j_{|S|}}$. We say a set S is a *cover* of the knapsack polytope if $\sum_{i\in S} w_i > W$. Let $E(S)$ denote the *extension* of S where $E(S) = S \cup \{i : i \in [n], w_i \ge w_{j_1}\}$, so $E(S)$ contains all items in the cover along with all items with weights greater than or equal to the maximal weighted item in the cover. We let the *cover inequality* be defined as

$$\sum_{i\in S} x_i \le |S| - 1 \tag{6.1}$$

and the *extended cover inequality* be defined as

$$\sum_{i\in E(S)} x_i \le |S| - 1. \tag{6.2}$$

It is straightforward to verify that both (6.1) and (6.2) are valid inequalities for the knapsack polytope.

We call S a *strong cover* if S is a cover and satisfies the following conditions:

C_1. For every $i \in S$, we have $\sum_{j\in S\setminus\{i\}} w_j \le W$.

C_2. For every $i \in [n] \setminus E(S)$, we have $\sum_{j\in S\setminus\{j_1\}} w_j + w_i \le W$.

Balas showed in [6] the following characterization of facet defining 0/1-inequalities of the knapsack polytope:

Theorem 6.2.1. *Let $Q \subseteq [n]$ and $k \geq 0$. An inequality of the form*

$$\sum_{i \in Q} x_i \leq k$$

is a facet-defining inequality of the knapsack polytope if and only if $Q = E(S)$ for some strong cover S, $k = |S| - 1$ and

$$\sum_{j \in S \setminus \{j_1, j_2\}} w_j + w_{\max} \leq W,$$

where $w_{\max} = \max\{w_i : i \in [n]\}$.

6.3 Lift and push cover inequalities for the incremental knapsack polytope

In this section, we show, starting with a cover inequality for the knapsack polytope presented in the last section, how to obtain the stronger lift and push cover inequality for the incremental knapsack polytope. This procedure is formalized in Section 6.3.1. In Section 6.3.2, we extend one direction of Balas's result in Theorem 6.2.1 and give sufficient conditions for when life and push cover inequalities define facets for the incremental knapsack polytope. Finally, we show that these conditions are not necessary.

6.3.1 The lift and push procedure

Recall that the generalized incremental knapsack problem has the following integer programming formulation:

$$\begin{aligned}
\max \quad & \sum_{i\in[n]} \sum_{t\in[T]} p_{i,t}(x_{i,t} - x_{i,t-1}) & \\
\text{s.t.} \quad & \sum_{i\in[n]} w_i x_{i,t} \leq W_t & \forall\, t \in [T] \\
& x_{i,t} \leq x_{i,t+1} & \forall\, i \in [n], t \in [T-1] \\
& x_{i,t} \in \{0,1\} & \forall\, i \in [n], t \in [T]
\end{aligned} \tag{GIK-IP}$$

Let the incremental knapsack polytope be the convex hull of the feasible points satisfying the constraints of (GIK-IP).

We say S is a *cover* of the incremental knapsack polytope at time t if for some $t \in [T]$, we have $\sum_{i\in S} x_i > W_t$. We say S is a *strong cover* of the incremental knapsack polytope at time t if S is a cover that satisfies,

C_1. For every $i \in S$, we have $\sum_{j\in S\setminus\{i\}} w_j \leq W_t$.

C_2. For every $i \in [n] \setminus E(S)$, we have $\sum_{j\in S\setminus\{j_1\}} w_j + w_i \leq W_t$.

We remark that these definitions are identical to those for the knapsack polytope, with the added specification of a time parameter t. It is easy to verify that, analogous to their knapsack counterparts given in (6.1) and (6.2), the cover inequality and extended cover inequality at time t, given respectively in (6.3) and (6.4) below, are both valid inequalities for the knapsack polytope.

$$\sum_{i\in S} w_i x_{i,t} \leq |S| - 1 \tag{6.3}$$

$$\sum_{i\in E(S)} w_i x_{i,t} \leq |S| - 1. \tag{6.4}$$

We can further strengthen the above inequalities as follows. Starting with a cover S in some

time $t \in [T]$, The lift and push cover inequality is defined as

$$\sum_{i \in E(S)} x_{i,t_{i,S}} \leq |S| - 1, \tag{6.5}$$

where for $i \in S$

$$t_{i,S} = t_S = \max \left\{ t : \sum_{j \in S} w_j > W_t \right\}, \tag{6.6}$$

and for $i \notin S$

$$t_{i,S} = \max \left\{ t : \sum_{j \in S \setminus \{j_1\}} w_j + w_i > W_t \right\}. \tag{6.7}$$

Note that every cover S uniquely determines a lift and push cover inequality. Therefore, for each cover S, we call the associated lift and push cover inequality the *LPCI of* S. Note that $t_{i,S} = t_{j,S}$ for any $i, j \in S$. Therefore, we denote $t_S = t_{i,S}$ for every $i \in S$. When it is clear from context what the cover S is in the definition of (6.7), we simply denote $t_{i,S}$ as t_i. The next lemma shows that this procedure gives a valid inequality.

Lemma 6.3.1. *Given a cover S for some time $t \in [T]$, the lift and push cover inequality of S as given by* (6.5) *is valid.*

Proof. If $E(S) = S$, validity of (6.5) is trivial since S is a cover in time t_S. Thus we assume $E(S) \setminus S$ is nonempty. Suppose (6.5) is not valid, then there exists $\bar{x}$ feasible for (GIK-IP) that violates (6.5). Let $K_1 = \{i \in S : \bar{x}_{i,t_S} = 1\}$ and $K_2 = \{i \in E(S) \setminus S : \bar{x}_{i,t_i} = 1\}$. Since $\bar{x}$ violates (6.5), we know that $|K_1| + |K_2| \geq |S|$. Since $\bar{x}$ satisfies the constraints of (GIK-IP) and S is a cover in t_S, we have $|K_1| \leq |S| - 1$. Therefore, $|K_2| \geq 1$. Let $k' \in K_2$ so that $t'_{k'} = \max\{t_i : i \in K_2\}$, then

$$\begin{aligned}
\sum_{i \in [n]} w_i \bar{x}_{i,t'_{k'}} &\geq \sum_{i \in K_1} w_i + \sum_{i \in K_2} w_i \\
&\geq \sum_{i \in K_1} w_i + (|K_2| - 1) w_{j_1} + w_{k'} \\
&\geq \sum_{i \in S \setminus \{j_1\}} w_i + w_{k'}
\end{aligned}$$

$$> \quad W_{t'_{k'}},$$

where the first inequality is by definition of K_1 and K_2, the second inequality is by noting for each $i \in K_2$, $w_i \geq w_{j_1}$, the third inequality is by noting $|K_1| + |K_2| - 1 \geq |S| - 1$ by assumption, and the final inequality is by (6.7). We have reached a contradiction since $\bar{x}$ satisfies $\sum_{i\in[n]} w_i \bar{x}_{i,t'_{k'}} \leq W_{t'_{k'}}$ by feasibility of $\bar{x}$. □

6.3.2 Facet defining lift and push cover inequalities

In this section, we give sufficient conditions for when the lift and push procedure from Section 6.3.1 defines a facet for the incremental knapsack polytope. The main result is given in the following theorem.

Theorem 6.3.2. *The inequality*

$$\sum_{i\in S} x_{i,t_S} + \sum_{i\in E(S)\setminus S} x_{i,t_i} \leq |S| - 1, \tag{6.8}$$

with t_S and t_i defined by (6.6) *and* (6.7) *respectively, defines a facet of the incremental knapsack polytope if S is a strong cover at time t_S, $|S| \geq 2$, and*

$$\sum_{i\in S\setminus\{j_1,j_2\}} w_i + w_{\max} \leq W_{t_S}.$$

Before proceeding with the proof, we remark that the conditions in Theorem 6.3.2 are identical to Balas's result in Theorem 6.2.1. However, while Balas's result holds as an if and only if, the converse of Theorem 6.3.2 is not true. In particular, as shown in the example below, there exists $0/1$ facet-defining inequalities of the incremental knapsack polytope that can be lifted from non-strong covers.

Example 6.3.3. *Consider the following constraints of an instance of the incremental knapsack*

problem with $n = 5$ and $T = 2$,

$$\begin{aligned} & 4x_{1,1} + 3x_{2,1} + 2x_{3,1} + 2x_{4,1} + 2x_{5,1} \leq 6 \\ & 4x_{1,2} + 3x_{2,2} + 2x_{3,2} + 2x_{4,2} + 2x_{5,2} \leq 9 \\ & x_{i,1} \leq x_{i,2} \qquad \forall\, i \in [5] \\ & x_{i,t} \in \{0,1\} \qquad \forall\, i \in [5], t \in [2]. \end{aligned} \tag{6.9}$$

We show in Appendix F.1.1 that the following inequality defines a facet for the corresponding incremental knapsack polytope:

$$x_{1,2} + x_{2,1} + x_{3,1} + x_{4,1} + x_{5,2} \leq 3 \tag{6.10}$$

Notice that this inequality can be obtained starting with the cover $S = \{2,3,4,5\}$ in time 1 with the corresponding extended cover inequality of S,

$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} + x_{5,1} \leq 3.$$

However, $\{2,3,4,5\}$ is not a strong cover in time 1, as it violates Condition C_1. In addition, in (6.10), the definition of t_5 does not match the definition given in (6.6).

Proof of Theorem 6.3.2. For ease of notation, let P denote the incremental knapsack polytope. Let $D = \{(i,t) : w_i \leq W_t, i \in [n], t \in [T]\}$. It is easy to see that $\dim(P) \leq |D|$. To show that (6.8) is a facet, we will define $|D|$ linearly independent feasible solutions of P that satisfy (6.8) at equality.

To this end, we partition $[n]$ into 3 sets: S, $S_1 = E(S) \setminus S$, and $S_2 = [n] \setminus E(S)$. We assume without loss of generality that $S = \{1, 2, \ldots, |S|\}$. For each $i \in [n]$, we let t'_i denote the first time t in which w_i is feasible, i.e., $t'_i = \min\{t : w_i \leq W_t\}$. For each $(\hat{\imath}, \hat{t}) \in D$, we will define a *solution generated by* $(\hat{\imath}, \hat{t})$. In the solution generated by $(\hat{\imath}, \hat{t})$, we let $x_{\hat{\imath},\tau} = 1$ for all $\tau \geq \hat{t}$ and $x_{\hat{\imath},\tau} = 0$ for all $\tau < \hat{t}$. For any $j \in [n]$ and $j \neq \hat{\imath}$, $x_{j,t}$ will be defined differently for different cases in what follows.

First, assume $\hat{\imath} \in S$ and $t'_{\hat{\imath}} \leq \hat{t} \leq t_S$. For $\hat{\imath} \neq |S|$ and $j \neq \hat{\imath}$, define:

$$x_{j,t} = \begin{cases} 1, & \text{if } t_S \leq t \leq T, j \in S \setminus \{\hat{\imath}, \hat{\imath}+1\} \\ 0, & \text{otherwise} \end{cases} \tag{6.11}$$

For $\hat{\imath} = |S|$ and $j \neq \hat{\imath}$, define:

$$x_{j,t} = \begin{cases} 1, & \text{if } t_S \leq t \leq T, j \in S \setminus \{\hat{\imath}, 1\} \\ 0, & \text{otherwise} \end{cases} \tag{6.12}$$

Definitions (6.11) and (6.12) together generate $\sum_{i \in S}(t_S - t'_i + 1)$ solution assignments. The next claim shows that the solutions as defined above are indeed feasible and satisfy (6.8) at equality. For readability, its straightforward proof is provided in Appendix F.1.2.

Claim 6.3.4. *The solutions generated by $(\hat{\imath}, \hat{t})$ for $\hat{\imath} \in S$ and $t'_{\hat{\imath}} \leq \hat{t} \leq t_S$ are feasible and satisfy* (6.8) *at equality.*

Again assume $\hat{\imath} \in S$, and fix $\hat{t} > t_S$, for any $j \neq \hat{\imath}$, define:

$$x_{j,t} = \begin{cases} 1, & \text{if } t_S \leq t \leq T, j \in S \setminus \{\hat{\imath}\} \\ 0, & \text{otherwise} \end{cases} \tag{6.13}$$

Definition (6.13) generates the remaining $|S| \cdot (T - t_S)$ solution assignments for $\hat{\imath} \in S$. The next claim shows that the solutions are feasible and satisfy (6.8) at equality; its proof is deferred to Appendix F.1.3.

Claim 6.3.5. *The solutions generated by $(\hat{\imath}, \hat{t})$ for $\hat{\imath} \in S$ and $t_S < \hat{t} \leq T$ are feasible and satisfy* (6.8) *at equality.*

Now assume $\hat{\imath} \in S_1$, fix any $t'_{\hat{\imath}} \leq \hat{t} \leq t_{\hat{\imath}}$, where $t_{\hat{\imath}}$ is defined by (6.7). For any $j \neq \hat{\imath}$, define:

$$x_{j,t} = \begin{cases} 1, & \text{if } t_S \leq t \leq T, j \in S \setminus \{j_1, j_2\} \\ 0, & \text{otherwise} \end{cases} \tag{6.14}$$

For $\hat{\imath} \in S_1$, fix any $\hat{t} > t_{\hat{\imath}}$. For any $j \neq \hat{\imath}$, define:

$$x_{j,t} = \begin{cases} 1, & \text{if } t_S \leq t \leq T, j \in S \setminus \{j_1\} \\ 0, & \text{otherwise} \end{cases} \tag{6.15}$$

Definitions (6.14) and (6.15) together generate $\sum_{i \in S_1}(T - t'_i + 1)$ solution assignments. The following two claims show the solutions as defined above are feasible and satisfy (6.8) at equality. Their proofs can be found in Appendices F.1.4 and F.1.5 respectively.

Claim 6.3.6. *The solutions generated by* $(\hat{\imath}, \hat{t})$ *for* $\hat{\imath} \in S_1$ *and* $t'_{\hat{\imath}} \leq \hat{t} \leq t_{\hat{\imath}}$ *are feasible and satisfy* (6.8) *at equality.*

Claim 6.3.7. *The solutions generated by* $(\hat{\imath}, \hat{t})$ *for* $\hat{\imath} \in S_1$ *and* $\hat{t} > t_{\hat{\imath}}$ *are feasible and satisfy* (6.8) *at equality.*

Finally, for $\hat{\imath} \in S_2$, fix any $\hat{t} \geq t'_{\hat{\imath}}$, for any $j \neq \hat{\imath}$, define:

$$x_{j,t} = \begin{cases} 1, & \text{if } t_S \leq t \leq T, j \in S \setminus \{j_1\} \\ 0, & \text{otherwise} \end{cases} \tag{6.16}$$

Definition (6.16) generates the remaining $\sum_{i \in S_2}(T - t'_i + 1)$ solution assignments. The following claim shows that the solutions as defined above are feasible and satisfy (6.8) at equality. Its proof is provided in Appendix F.1.6.

Claim 6.3.8. *The solutions generated by* $(\hat{\imath}, \hat{t})$ *for* $\hat{\imath} \in S_2$ *and* $\hat{t} \geq t'_{\hat{\imath}}$ *are feasible and satisfy* (6.8) *at equality.*

Thus far, we have given $|D|$ feasible solutions that satisfy (6.8) at equality. Finally, in the next claim, we show that all $|D|$ solutions are linearly independent, which concludes the proof that (6.8) defines a facet.

Claim 6.3.9. *All solution assignments generated by* $(\hat{\imath}, \hat{t}) \in D$ *are linearly independent.*

Proof. To show that all solution assignments are linearly independent, consider a $|D| \times nT$ matrix where each row is a solution assignment. We want to show the matrix is full rank. If there exists a column with only 1 nonzero element, we can eliminate the row that contains the unique nonzero element for that column, since it cannot be linearly dependent to other rows. For any $i \in S_1 \cup S_2$ and any $t \in [T]$, $x_{i,t} = 1$ for solutions generated by $(i, \hat{t})$ for $\hat{t} \leq t$, and $x_{i,t} = 0$ for all other solutions. In particular, for any $i \in S_1 \cup S_2$, $x_{i,1} = 1$ only for the solution generated by $(i, 1)$, allowing us to eliminate the row corresponding to the solution generated by $(i, 1)$. After this elimination, now $x_{i,2} = 1$ only for the solution generated by $(i, 2)$, allowing us to eliminate the corresponding row as well. We can do this sequentially followed by $t = 3, \ldots, T$ until we have eliminated all rows corresponding to solutions generated by (i, t) for any $i \in S_1 \cup S_2$ and any $t \geq t'_i$.

Similarly, for any $i \in S$ and $t < t_S$, $x_{i,t} = 1$ only for solutions generated by $(i, \hat{t})$ for $\hat{t} \leq t$. Hence, we can do the same elimination as above for rows corresponding to the solutions generated by (i, t) sequentially for $t = 1, 2, \ldots, t_S - 1$.

We are left with solution assignments generated by (i, t) for $i \in S$ and $t \geq t_S$ and the columns corresponding to variables $x_{i,t}$ for $i \in S$ and $t \geq t_S$. We denote the corresponding sub-matrix as X. If X is full rank, the entire solution matrix is full rank. For all $i \in S$ and $i \geq 2$, $x_{i,T} = 0$ only for the solution assignment generated by $(i - 1, t_S)$. If $i \in S$ and $i = 1$, $x_{i,T} = 0$ only for the solution assignment generated by $(|S|, t_S)$. Thus, we can eliminate rows corresponding to the solutions generated by (i, t_S) for any $i \in S$. With these rows eliminated, for any $i \in S$ and $t_S \leq t < T$, $x_{i,t} = 0$ only for the solution assignment generated by $(i, \hat{t})$ for $\hat{t} > t$. Thus, for $i \in S$, we can eliminate rows corresponding to the solutions generated by (i, t) sequentially for $t = T - 1, T - 2, \ldots, t_S$ and conclude X is full rank.

To justify the above operation, in a binary matrix, columns with only 1 zero element cannot

be linearly dependent to any other column as long as exchanging 0's and 1's does not change the rank of X; or equivalently, since X does not contain the 0 vector, the vectors of X are affinely independent implies that vectors of $1-X$ are affinely independent. Suppose vectors of $1-X$ are not affinely independent. Let $\vec{x}_1, \ldots, \vec{x}_r$ be the columns of X. There exists α such that $\sum_{i=1}^{r} \alpha_i(1-\vec{x}_i) = 0$, $\sum_{i=1}^{r} \alpha_i = 0$, $\alpha_i \neq 0$ for some i. Therefore,

$$\sum_{i=1}^{r} \alpha_i(1-\vec{x}_i) = \sum_{i=1}^{r} \alpha_i - \sum_{i=1}^{r} \alpha_i\vec{x}_i = 0 - \sum_{i=1}^{r} \alpha_i\vec{x}_i = 0.$$

Thus, we have $\sum_{i=1}^{r} \alpha_i\vec{x}_i = 0$, hence vectors of X are affinely independent. Hence the sub-matrix X is full rank. □

□

6.4 Separation algorithms

In this section, we propose separation algorithms for lift and push cover inequalities. These ideas build upon cover inequality separation algorithms for the classical knapsack polytope given in [45]. In Section 6.4.1, we give an exact separation algorithm in pseudo-polynomial time. In Section 6.4.2, we give an approximate separation algorithm in polynomial time.

6.4.1 Exact separation

Consider a fractional point $\bar{x} \in [0,1]^{nT}$ that satisfies the constraints of the linear relaxation of (GIK-IP). The lift and push cover inequality separation problem is to either find a lift and push cover inequality that $\bar{x}$ violates, or to conclude that $\bar{x}$ satisfies all such inequalities. In the remainder of this section, we provide the exact separation algorithm that proves the following theorem.

Theorem 1.3.8. *Given a fractional point $\bar{x} \in [0,1]^{nT}$ that satisfies the linear relaxation of* (GIK-IP)*, the lift and push cover inequality separation problem can be solved in time $O(Tn^4||w||_\infty)$.*

Notice that since the running time depends on the input parameter w, it is pseudo-polynomial in nature. Even in the knapsack setting, both cover inequality separation and extended cover inequality separation are NP-hard [48, 20]. Thus, a polynomial time separation algorithm is not possible unless P = NP. To prove this result, we first give a reformulation of the separation problem in Section 6.4.1.1. Using this reformulation, we develop a dynamic programming approach in Section 6.4.1.2 that gives the result of Theorem 1.3.8.

6.4.1.1 Separation problem reformulation

In this section, we give a reformulation for the lift and push cover inequalities separation problem. Let $\bar{x}$ be a fractional point that satisfies the constraints of (GIK-IP). Suppose items are sorted in non-decreasing weight order so that $w_1 \leq \cdots \leq w_n$, with ties broken arbitrarily. For each $t \in [T]$, $k \in [n]$ and $r = 0, \ldots, \sum_{i\in[n]} w_i$, we define:

$$f_t(k,r) = \min\left\{\sum_{j=1}^{k}(1-\bar{x}_{j,t})y_{j,t} : \sum_{j=1}^{k} w_j y_{j,t} = r, y_{k,t} = 1, y \in \{0,1\}^k\right\}. \tag{6.17}$$

For each $k \in [n]$, $r = 0, \ldots, \sum_{i\in[n]} w_i$ and $i \in \{k+1, \ldots, n\}$, we define:

$$t_{(k,r)}(i) = \max\{t : r - w_k + w_i > W_t\}. \tag{6.18}$$

The next lemma shows that in order to solve the lift and push cover inequality separation problem, it is sufficient to solve for $f_t(k,r)$ for all possible values of t, k and r.

Lemma 6.4.1. *$\bar{x}$ can be separated by a lift and push cover inequality given in* (6.5) *if and only if there exist $t \in [T]$, $k \in [n]$ and $r \geq W_t + 1$ such that*

$$f_t(k,r) - \sum_{j=k+1}^{n} \bar{x}_{j,t_{(k,r)}(j)} < 1, \tag{6.19}$$

where $t_{(k,r)}(j)$ is as defined in (6.18).

In particular, if (6.19) *is satisfied, let y^* be the solution that gives $f_t(k,r)$. Then $\bar{x}$ violates the*

LPCI of S^* where $S^* = \{i \in [k] : y^*_{i,t} = 1\}$.

Proof. First, we prove the forward direction. Suppose there exists LPCI of S for some cover S that separates $\bar{x}$. Let $k = \max\{i : i \in S\}$. First, we will show we can assume that $E(S) \setminus S = \{k+1, \ldots, n\}$. Clearly $E(S) \setminus S \supseteq \{k+1, \ldots, n\}$. Suppose $E(S) \setminus S \neq \{k+1, \ldots, n\}$, then there must exists $j \notin S$ such that $w_j = w_k$ and $j < k$, let $\bar{S} = S \setminus \{k\} \cup \{j\}$. We claim that if $\bar{x}$ violates the LPCI of S, it also violates the LPCI of $\bar{S}$. Notice that by definition, we have $E(S) = E(\bar{S})$. It is also straightforward to verify through the definitions of (6.6) and (6.7) that for each $i \in E(S)$, we have $t_{i,S} = t_{i,\bar{S}}$.

Therefore,

$$\sum_{i \in E(\bar{S})} \bar{x}_{i,t_{i,\bar{S}}} = \sum_{i \in E(S)} \bar{x}_{i,t_{i,S}} > |S| - 1 = |\bar{S}| - 1.$$

Thus, $\bar{S}$ also separates $\bar{x}$. We can repeat the above procedure until no such element j exists.

Hence, we let S denote a cover where the LPCI of S separates $\bar{x}$ and $E(S) \setminus S = \{k+1, \ldots, n\}$. Additionally, we let $t = t_S$ and $r = w(S)$. Note that r indeed satisfies $r \geq W_t + 1$ since S is a cover at time t. Furthermore, for every $i \in \{k+1, \ldots, n\}$, we have:

$$t_{(k,r)}(i) = \max\{t : w(S) - w_k + w_i > W_t\} = t_{i,S}, \tag{6.20}$$

where $t_{i,S}$ is defined by (6.7).

Let $\bar{y}_{j,t} = 1$ if $j \in S$, $\bar{y}_{j,t} = 0$ otherwise. Notice that $\sum_{j=1}^{k} w_j \bar{y}_{j,t} = w(S) = r$ and since $k \in S$, we have $\bar{y}_{k,t} = 1$. Thus, $\bar{y}$ satisfies the constraints of (6.17). Given these definitions, we have:

$$\begin{aligned} f_t(k,r) - \sum_{j=k+1}^{n} \bar{x}_{j,t_{(k,r)}(j)} &\leq \sum_{j=1}^{k} (1 - \bar{x}_{j,t}) \bar{y}_{j,t} - \sum_{j=k+1}^{n} \bar{x}_{j,t_{(k,r)}(j)} \\ &= |S| - \sum_{j \in S} \bar{x}_{j,t} - \sum_{j \in E(S) \setminus S} \bar{x}_{j,t_j} \\ &< |S| - (|S| - 1) \\ &= 1, \end{aligned}$$

where the first inequality follows since as explained above, $\bar{y}$ satisfies the constraints of (6.17). The first equality follows by definition of $\bar{y}$ and by (6.20). The final inequality follows since we know $\bar{x}$ violates the LPCI of S.

For the reverse direction, let y^* be the minimizer of (6.17) that gives $f_t(k, r)$ for some $t \in [T]$, $k \in [n]$ and $r \geq W_t + 1$ such that (6.19) is satisfied. Let $S^* = \{i \in [k] : y^*_{i,t} = 1\}$. We will show that $\bar{x}$ violates the LPCI of S^*. Notice that since $t_{S^*} = \max\{\tau : \sum_{j \in S} w_j > W_\tau\}$ where $\sum_{j \in S} w_j = \sum_{j=1}^{k} w_j y^*_{j,t} = r \geq W_t + 1$, we necessarily have $t_{S^*} \geq t$. Furthermore, by definition, we have $E(S^*) \setminus S^* \supseteq \{k+1, \ldots, n\}$ and for every $j \in \{k+1, \ldots, n\}$, we have

$$\begin{aligned} t_j &= \max\{\tau : \sum_{i \in S} w_i - w_k + w_j > W_\tau\} \\ &= \max\{\tau : r - w_k + w_j > W_\tau\} \\ &= t_{(k,r)}(j). \end{aligned}$$

It follows that

$$\begin{aligned} |S^*| - \sum_{j \in S^*} \bar{x}_{j,t_{S^*}} - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j} &\leq |S^*| - \sum_{j \in S^*} \bar{x}_{j,t} - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j} \\ &= \sum_{j=1}^{k} (1 - \bar{x}_{j,t}) y^*_{j,t} - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j} \\ &\leq f_t(k, r) - \sum_{j=k+1}^{n} \bar{x}_{j,t_{(k,r)}(j)} \\ &< 1, \end{aligned}$$

where the first inequality follows since by feasibility of $\bar{x}$, we have $\bar{x}_{j,t_{S^*}} \geq \bar{x}_{j,t}$ for all $j \in [n]$, $t_{S^*} \geq t$. The first equality follows by definition of S^*. The second inequality follows by definition of y^* and by noting that for $j \in \{k+1, \ldots, n\} \subseteq E(S) \setminus S$, we have $t_j = t_{(k,r)}(j)$ as explained above. The final inequality by assumption.

Rearranging the inequality, we have $\sum_{j \in S} \bar{x}_{j,t_{S^*}} + \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j} > |S| - 1$. We conclude that $\bar{x}$ can be separated by the LPCI of S^*. □

6.4.1.2 *Dynamic programming Approach*

Given Lemma 6.4.1, to find a lift and push cover equality that separates $\bar{x}$, it suffices to obtain $f_t(k, r)$ for every $t \in [T]$, $k \in [n]$ and $r = 0, \ldots, \sum_{i\in[n]} w_i$ and to find such $f_t(k, r)$ where (6.19) is satisfied. If no such $f_t(k, r)$ exists, we can conclude that $\bar{x}$ satisfies all lift and push cover inequalities. In the remainder of the section, we give a dynamic programming approach to solve for $f_t(k, r)$.

Recursive equations. For all $t \in [T]$, we initialize $f_t(0, 0) = 0$. In cases where $w_k > r$, no point y satisfies the constraints of (6.17), in which case we set $f_t(k, r) = \infty$. By noting the constraint of $y_{k,t} = 1$ in (6.17), we develop the following recursive equations to solve for $f_t(k, r)$:

$$\begin{aligned} f_t(k,r) &= \min\left\{\sum_{j=1}^{k}(1-\bar{x}_{j,t})y_{j,t} : \sum_{j=1}^{k} w_j y_{j,t} = r, y_{k,t} = 1, y \in \{0,1\}^k\right\} \\ &= \min\left\{\sum_{j=1}^{k-1}(1-\bar{x}_{j,t})y_{j,t} + (1-\bar{x}_{k,t}) : \sum_{j=1}^{k-1} w_j y_{j,t} = r - w_k, y \in \{0,1\}^{k-1}\right\} \\ &= \min_{\ell<k} f_t(\ell, r - w_k) + (1-\bar{x}_{k,t}). \end{aligned}$$

Here, the first equality is by definition of $f_t(k, r)$, the second equality follows by noting we hold $y_{k,t} = 1$, the final equality follows by definition of $f_t(\ell, r - w_k)$.

From dynamic programming to separation. Notice that in (6.19), the term $\sum_{j=k+1}^{n} \bar{x}_{j,t_{(k,r)}(j)}$ can be calculated in linear time, independent of the value of $f_t(k, r)$. Thus, after computing $f_t(k, r)$ for all $t \in [T]$, $k \in [n]$ and $r \in [\sum_{i\in[n]} w_i]_0$, it is straight forward to enumerate through all states and find ones that satisfy (6.19). Let $f_t(k, r)$ be such a value, let y^* be the solution that achieves $f_t(k, r)$ and let $S^* = \{i : y^*_{i,t} = 1\}$. By Lemma 6.4.1, the lift and push cover inequality of S^* separates $\bar{x}$. On the other hand, if no such state satisfies (6.19), we can conclude, again by Lemma 6.4.1, that $\bar{x}$ satisfies all lift and push cover inequalities of (6.5).

Running time. Given the recursive equation of $f_t(k, r)$, each state takes $O(n)$ time to solve. We incur another $O(n)$ time to check if (6.19) is satisfied. There are $O(Tn^2||w||_\infty)$ number of states, giving a total running time of $O(Tn^4||w||_\infty)$, concluding Theorem 1.3.8.

6.4.2 Approximate separation

As discussed in the previous section, we can solve the separation problem in pseudo-polynomial time. In this section, we show that we can improve the running time to be polynomial in the problem's input size by relaxing the definition of the separation problem.

Theorem 1.3.9. *Given a fractional point $\bar{x} \in [0,1]^{nT}$ that satisfies the constraints of the linear relaxation of* (GIK-IP)*. For any $\epsilon > 0$, there exists an algorithm that, in time $O(T\frac{n^7}{\epsilon})$, gives a lift and push cover inequality that $\bar{x}$ violates, or concludes that $(1-\epsilon)\bar{x}$ satisfies all such inequalities.*

For ease of notation, for every $j \in [n]$ and $t \in [T]$, we let $\bar{p}_{j,t} = 1 - \bar{x}_{j,t}$. Notice that since $0 \leq \bar{x}_{j,t} \leq 1$, we have $\bar{p}_{j,t} \geq 0$. Without loss of generality we assume $\bar{p}$ is scaled so that for each $j \in [n]$ and $t \in [T]$, $\bar{p}_{j,t}$ is a non-negative integer. As is the case in Section 6.4.1, we assume items are sorted in non-decreasing weight order so that $w_1 \leq \cdots \leq w_n$. We define

$$g_t(k, q) = \max\left\{\sum_{j=1}^{k} w_j y_{j,t} : \sum_{j=1}^{k} \bar{p}_{j,t} y_{j,t} = q, y_{k,t} = 1, y \in \{0,1\}^k\right\}.$$

For each $t \in [T]$, let $v^t_1, v^t_2, \ldots, v^t_n$ be the ordering of distinct indices such that $\bar{p}_{v^t_1,t} \leq \bar{p}_{v^t_2,t} \leq \cdots \leq \bar{p}_{v^t_n,t}$. Let $v^t(j) = \{\ell : v^t_\ell = j\}$. For all $i \in [n]$, we define subproblems:

$$g_{t,i}(k, q) = \max\left\{\sum_{j=1}^{k} w_j y_{j,t} : \sum_{j=1}^{k} \bar{p}_{j,t} y_{j,t} = q, y_{k,t} = 1, y \in \{0,1\}^k, y_{j,t} = 0 \,\forall\, v^t(j) > i\right\}. \quad (6.21)$$

Recursive equations. Similar to our approach to solve for $f_t(k, r)$ in the previous section, we give the following recursive equations to solve for $g_{t,i}(k, q)$. We initialize $g_{t,i}(0, 0) = 0$ for all $t \in [T]$ and $i \in [n]$. If $v^t(k) > i$ or if $q < \bar{p}_{k,t}$, then no point satisfies the constraints of (6.21) and we let $g_{t,i}(k, q) = -\infty$. Else, by noting that $y_{k,t} = 1$ for each $g_{t,i}(k, q)$, for $v^t(k) \leq i$, we rewrite

$g_{t,i}(k,q)$ as follows:

$$g_{t,i}(k,q) = \max\left\{\sum_{j=1}^{k-1} w_j y_{j,t} + w_k : \sum_{j=1}^{k-1} \bar{p}_{j,t} y_{j,t} = q - \bar{p}_{k,t}, \right.$$
$$\left. y \in \{0,1\}^{k-1}, y_{j,t} = 0 \,\forall\, v^t(j) > i \right\}.$$

Thus, we obtain,

$$g_{t,i}(k,q) = \begin{cases} \max\limits_{\ell<k} g_{t,i}(\ell, q - \bar{p}_{k,t}) + w_k, & \text{if } v^t(k) \le i \text{ and } q \ge \bar{p}_{k,t} \\ -\infty, & \text{otherwise.} \end{cases} \tag{6.22}$$

6.4.2.1 Approximate dynamic programming

If we were to solve every value of $g_{t,i}(k,q)$ for $t \in [T]$, $i \in [n]$, $k \in [n]$ and $q = \sum_{j=1}^{k} \bar{p}_{j,t}$ given the recursive equations above, the running time is still pseudo-polynomial due to the number of possible values of q. We limit the state space of q by rounding down $\bar{p}$ as follows:
Let $\tilde{\epsilon} = \frac{\epsilon}{n}$. For each $t \in [T]$, and $i \in [n]$:

- Let $c_i = \frac{\tilde{\epsilon} \bar{p}_{v^t_i,t}}{i}$.
- Let $\tilde{p}_{j,t} = \lfloor \frac{\bar{p}_{j,t}}{c_i} \rfloor$ for all $v^t(j) \le i$.

For each $t \in [T]$, $i \in [n]$, $k \in [n]$ and $q = 0, \ldots, \sum_{j \in [k]: v^t(j) \le i} \tilde{p}_{j,t}$. Let $\tilde{g}_{t,i}(k,q)$ be defined as

$$\tilde{g}_{t,i}(k,q) = \max\left\{\sum_{j=1}^{k} w_j y_{j,t} : \sum_{j=1}^{k} \tilde{p}_{j,t} y_{j,t} = q, y_{k,t} = 1, y \in \{0,1\}^k, y_{j,t} = 0 \,\forall\, v^t(j) > i \right\}.$$

Note that this definition is identical to that of $\tilde{g}_{t,i}(k,q)$, except with profits $\tilde{p}$ instead of $\bar{p}$. We also remark that limiting the values of q to non-negative integers between 0 and $\sum_{j \in [k]: v^t(j) \le i} \tilde{p}_{j,t}$ is without loss of generality since we have the constraint $y_{j,t} = 0$ for all j such that $v^t(j) > i$. The recursion to solve for $\tilde{g}$ is therefore identical to that of g, except with $\tilde{p}$ replacing $\bar{p}$.

6.4.2.2 *Proof of Theorem 1.3.9*

In this section, we give a proof of Theorem 1.3.9. We first discuss how evaluating $\tilde{g}_{t,i}(k,q)$ for all possible states could give a lift and push cover inequality that $\bar{x}$ violates, if one exists. Then, we prove that if our prescribed procedure fails to return a lift and push cover inequality, then $(1-\epsilon)\bar{x}$ satisfies all lift and push cover inequalities (see Lemma 6.4.2). Finally, we conclude the section by discussing the overall running time.

From approximate dynamic programming to approximate separation. First, for any cover S such that $|S| = 1$, the inequality of (6.5) reduces to $\sum_{i\in E(S)} \bar{x}_{i,t_i} \leq 0$. If there exists such an inequality that separates $\bar{x}$, we can find it in polynomial time through enumeration, or conclude that $\bar{x}$ satisfies all such inequalities. Thus, going forward, we assume $\bar{x}$ satisfies all lift an push cover inequalities for covers of cardinality 1.

Let $\bar{y}$ be the solution that gives $\tilde{g}_{t,i}(k,q)$. If $\tilde{g}_{t,i}(k,q) \geq W_t + 1$, let $\bar{S} = \{i \in [k] : \bar{y}_{i,t} = 1\}$. By definition, $w(\bar{S}) = \tilde{g}_t(k,q)$ so $\bar{S}$ is a cover at time t. Given $\bar{S}$, we can obtain the LPCI of $\bar{S}$ given by (6.5).

If $\sum_{j\in E(\bar{S})} \bar{x}_{j,t_j} > |\bar{S}| - 1$, we have found a lift and push cover inequality that separates $\bar{x}$. Otherwise, suppose no such inequality exists among all states $t \in [T]$, $i \in [n]$, $k \in [n]$ and $q \in [\sum_{j\in[k]:v^t(j)\leq i} \tilde{p}_{j,t}]_0$. In this case, in the lemma that follows, we show that $(1-\epsilon)\bar{x}$ satisfies all lift and push cover inequalities of S for $|S| \geq 2$.

Lemma 6.4.2. *If for all states $t \in [T]$, $i,k \in [n]$, $q \in \{0,\ldots,\sum_{j=1}^{n} \tilde{p}_{j,t}\}$, with associated cover $\bar{S}$ of $\tilde{g}_{t,i}(k,q)$ as defined above, we have $\sum_{j\in E(\bar{S})} \bar{x}_{j,t_j} \leq |\bar{S}| - 1$, then for all covers $S \subseteq 2^{[n]}$ where $|S| \geq 2$, $\bar{x}$ satisfies*

$$(1-\epsilon) \sum_{j\in E(S)} \bar{x}_{j,t_j} \leq |S| - 1,$$

as defined in (6.5).

Proof. Hold fixed some integer s where $2 \leq s \leq n$. For any cover $S \subseteq 2^{[n]}$ such that $|S| = s$, let $y_{j,t_S} = 1$ for all $j \in S$, $y_{j,t_S} = 0$ otherwise. Among all covers S and y as defined above, let S^* and

y^* be one that minimizes the expression below,

$$\sum_{j=1}^{n}(1-\bar{x}_{j,t_S})y_{j,t_S} - \sum_{j\in E(S)\setminus S}\bar{x}_{j,t_j}. \tag{6.23}$$

We first show that S^* satisfies the property given in Claim 6.4.3. Then we show that to prove that $(1-\epsilon)\bar{x}$ satisfies all lift and push cover inequalities of cardinality s, it is sufficient to prove that $(1-\epsilon)\bar{x}$ satisfies the LPCI of S^*. Their proofs can be found in Appendices F.2.1 and F.2.2 respectively.

Claim 6.4.3. *There exists S^* that minimizes* (6.23) *such that $E(S^*)\setminus S^* = \{k^*+1,\dots,n\}$, where $k^* = \max\{i : i \in S^*\}$.*

Claim 6.4.4. *If $(1-\epsilon)\sum_{j\in E(S^*)}\bar{x}_{j,t_j} \le |S^*|-1$, then for all other covers S such that $|S| = |S^*|$, we also have*

$$(1-\epsilon)\sum_{j\in E(S)}\bar{x}_{j,t_j} \le |S|-1.$$

Let $k^* = \max\{i : i \in S^*\}$; let $t^* = t_{|S^*|}$; let $i^* = \max\{i : y^*_{v_i^{t^*},t^*} = 1\}$; let $q^* = \sum_{i\in S^*}\tilde{p}_{i,t^*}$. Notice that by definition of i^*, for all j such that $v^t(j) > i$, we have $y^*_{j,t} = 0$. Therefore, y^* is a feasible solution for $\tilde{g}_{t^*,i^*}(k^*,q^*)$. Let $\bar{y}$ be the solution that achieves $\tilde{g}_{t^*,i^*}(k^*,q^*)$. The next claim gives an upper bound for $\sum_{j=1}^{k^*}\bar{p}_{j,t^*}\bar{y}_{j,t^*}$ in terms of $\sum_{j=1}^{k^*}\bar{p}_{j,t^*}y^*_{j,t^*}$. Its proof is provided in Appendix F.2.3.

Claim 6.4.5.

$$\sum_{j=1}^{k^*}\bar{p}_{j,t^*}\bar{y}_{j,t^*} \le (1+\tilde{\epsilon})\sum_{j=1}^{k^*}\bar{p}_{j,t^*}y^*_{j,t^*}.$$

Recall that $S^* = \{i : y^*_{i,t^*} = 1\}$. Similarly, we let $\bar{S} = \{i : \bar{y}_{i,t^*} = 1\}$. Since $\bar{y}$ is the solution that achieves $\tilde{g}_{t^*,i^*}(k^*,q^*)$ where y^* is feasible, we have that $w(\bar{S}) = \sum_{j=1}^{k^*}w_j\bar{y}_{j,t^*} \ge \sum_{j=1}^{k^*}w_jy^*_{j,t^*} = w(S^*)$. Therefore, clearly $\bar{S}$ is a cover. Furthermore, following the definition given in (6.7), for every $i \in (E(S^*)\setminus S^*)\cap(E(\bar{S})\setminus\bar{S})$, we have $t_{i,S^*} \le t_{i,\bar{S}}$.

By Claim 6.4.3 we have $E(S^*)\setminus S^* = \{k^*+1,\dots,n\}$; whereas by definition, we have $E(\bar{S})\setminus\bar{S} \supseteq \{k^*+1,\dots,n\}$. Thus, $E(S^*)\setminus S^* \subseteq E(\bar{S})\setminus\bar{S}$. Additionally, by definition, $k^* = \max\{i : i \in S^*\} = \max\{i : i \in \bar{S}\}$. Therefore, by (6.7), for all $j \in E(S^*)\setminus S^*$, we have $t_{j,S^*} \le t_{j,\bar{S}}$, which implies

$\bar{x}_{j,t_{j,S^*}} \leq \bar{x}_{j,t_{j,\bar{S}}}$. We conclude

$$\sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_{j,S^*}} \leq \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_{j,\bar{S}}} \leq \sum_{j \in E(\bar{S}) \setminus \bar{S}} \bar{x}_{j,t_{j,\bar{S}}} \tag{6.24}$$

Recall that for all $j \in [n]$ and $t \in [T]$, we have $\bar{p}_{j,t} = 1 - \bar{x}_{j,t}$. Therefore, combining Claim 6.4.5 and (6.24), we conclude

$$\begin{aligned} \sum_{j=1}^{k^*} (1 - \bar{x}_{j,t^*}) \bar{y}_{j,t^*} - \sum_{j \in E(\bar{S}) \setminus \bar{S}} \bar{x}_{j,t_{j,\bar{S}}} &\leq (1 + \tilde{\epsilon}) \sum_{j=1}^{k^*} (1 - \bar{x}_{j,t^*}) y^*_{j,t^*} - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_{j,S^*}} \\ &= (1 + \tilde{\epsilon})(|S^*| - \sum_{j \in S^*} \bar{x}_{j,t^*}) - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_{j,S^*}} \end{aligned} \tag{6.25}$$

Taking the left hand side of the above inequality, we have

$$\begin{aligned} \sum_{j=1}^{k^*} (1 - \bar{x}_{j,t^*}) \bar{y}_{j,t^*} - \sum_{j \in E(\bar{S}) \setminus \bar{S}} \bar{x}_{j,t_{j,\bar{S}}} &= |\bar{S}| - \sum_{j \in \bar{S}} \bar{x}_{j,t^*} - \sum_{j \in E(\bar{S}) \setminus \bar{S}} \bar{x}_{j,t_j} \\ &\geq |\bar{S}| - \sum_{j \in \bar{S}} \bar{x}_{j,t_{\bar{S}}} - \sum_{j \in E(\bar{S}) \setminus \bar{S}} \bar{x}_{j,t_j} \\ &\geq 1, \end{aligned}$$

where the equality follows by definition of $\bar{S}$, the first inequality follows by again noting $t_{\bar{S}} \geq t^*$ since $w(\bar{S}) \geq w(S^*)$, and the final inequality follows since we know that $\bar{x}$ satisfies the LPCI of $\bar{S}$. Combining the above inequality with the right hand side of (6.25), we have

$$1 \leq (1 + \tilde{\epsilon})(|S^*| - \sum_{j \in S^*} \bar{x}_{j,t^*}) - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j}.$$

Using this inequality, the next claim concludes the lemma. We present the algebraic details in Appendix F.2.4.

Claim 6.4.6. *For any S^* such that*

$$1 \leq (1+\tilde{\epsilon})(|S^*| - \sum_{j \in S^*} \bar{x}_{j,t^*}) - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j},$$

we have

$$(1-\epsilon)(\sum_{j \in S^*} \bar{x}_{j,t^*} + \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j}) \leq |S^*| - 1.$$

□

Running time. Finally, we conclude the proof of Theorem 1.3.9 by analyzing the running time of the procedure above. For each $t \in [T]$ and $i, k \in [n]$, for $v^t(j) \leq i$, we have

$$\tilde{p}_{j,t} \leq \tilde{p}_{v_i^t,t} \leq \frac{\bar{p}_{v_i^t,t}}{c_i} = \frac{i}{\tilde{\epsilon}} \leq \frac{n}{\tilde{\epsilon}} = \frac{n^2}{\epsilon}.$$

The number of possible states is therefore $O(T \cdot n^2 \cdot \frac{n^3}{\epsilon})$. The value function of $\tilde{g}$ can be solved in $O(n)$ time for each state. From the optimal solution of each value function, the associated LPCI can be computed in $O(n)$ time. Overall, the total running time is $O(T\frac{n^7}{\epsilon})$.

Conclusion

In this thesis, we present and study many incremental packing problems. We believe this thesis has made significant contributions to address these problems, including algorithmic ideas that could have impacts beyond the problems we discuss in this thesis. The results of this thesis also naturally lead to more open questions related to the problems and techniques explored here, which we will discuss below.

Single-time policies: extensions and limits. In Chapter 2, we present single-time policies that can easily be adapted to other incremental packing problems beyond the generalized incremental knapsack problem. Algorithms of this nature are tractable in practice as long as the time-underlying classical packing problem can be solved efficiently. However, many theoretical questions remain.

When do single-time policies lead to constant-factor approximations for incremental packing problems? The proof techniques we use for the generalized incremental knapsack problem exploits explicitly the fact that the classical knapsack problem admits an FTPAS. In particular, given that the knapsack problem can be approximated to a factor of $1-\epsilon$, using the c-flexible single-time algorithm, the generalized incremental knapsack problem can be approximated to a factor of $f(c, T\epsilon)$. We are then able to give a factor dependent on only c and ϵ by scaling down ϵ by a function of T. However, if the time-underlying packing problem only admits a PTAS or if it is already APX-hard (for example, in the case of the incremental generalized assignment problem)

and only admits an α-approximation for some fixed $\alpha \in (0, 1)$, it is an open question if we can still obtain a constant factor approximation ratio independent of T using single-time policies.

Limits of single-time policies? Because of the challenges described above, presumably, there are cases where a classical packing problem can be approximated by a constant factor efficiently but single-time policies would still not give a constant factor approximation for the incremental version of the problem. If this is indeed true, what are the limits to single-time policies?

Generalized incremental knapsack problem. In Chapter 3, we give a $(\frac{1}{2} - \epsilon)$-approximation for the generalized incremental knapsack problem and show that it admits a QPTAS. This leads to a number of natural open questions, which we elaborate upon below.

Improved constant-factor approximation? A natural direction for future research would be to investigate whether the $(\frac{1}{2} - \epsilon)$-approximation we obtained in Section 3.2 can be improved. One possible approach to achieve such improvements lies in proposing an efficient way to combine k-light and k-heavy items within a single solution, rather than constructing separate solutions that compete against each of these contributions by themselves. Our efforts along these lines have not been successful to date, perhaps since significantly different methods appear to be required.

Obstacles toward obtaining a PTAS? A particularly challenging direction to pursue is whether the quasi-PTAS we devised in Section 3.4 can be enhanced to admit a truly polynomial running time. In fact, we are unaware of any inapproximability result that rules out the existence of a PTAS for generalized incremental knapsack. To this end, the first bottleneck resides in the guessing step for bounded weight ratio instances in Section 3.3.2. The second bottleneck emerges from our $O(\frac{\log M}{\epsilon})$ bound on the number of items crossing each cluster in a near-optimal permutation, formally established in Lemma 3.4.3. Bypassing these two sources for quasi-polynomial running time would result in a polynomial-time implementation of our overall approach.

Monotone submodular all-or-nothing incremental knapsack problem. In Chapter 4, we give a PTAS for `IK-AoN` by showing that it is no harder than its linearized variant, `IK`. This settles the hardness of the problem as strongly NP-hard, making it substantially different from many other

submodular maximization problems, which are all known to be APX-hard. To bridge the gap between IK−AoN and submodular function maximization, we raise the following open questions.

Good approximations for more general monotone submodular functions? IK−AoN has a very specific profit assumption. Namely, the inclusion of each item in a set either earns the full profit or no profit at all. This is a profit structure that we exploit throughout Section 4.2 in order to reduce it to IK. Suppose we were to relax the profit structure, and say that we associate with each item i some profit set $\mathcal{P}_i$ such that $|\mathcal{P}_i|$ is bounded. The inclusion of each item in a set earns some $p_i \in \mathcal{P}_i$. In a monotone submodular function of this nature, under the incremental knapsack setting, we do not yet have any non-trivial approximation results. If this question can be answered in the positive, the next natural step is to extend the problem to monotone submodular functions in an incremental knapsack setting, without any additional mitigating assumptions on the profit function.

When does the problem become APX-hard? We know that IK−AoN is strongly NP-hard. On the other hand, if the profit function is a monotone submodular function, even if $T = 1$, we know that the problem cannot be approximated better than a factor of $1 - \frac{1}{e}$ unless P = NP [28]. Under the setting we mention in the previous paragraph, where an item may take on only a bounded number of profit, the question remains whether the problem is strongly NP-hard or APX-hard.

Single-machine algorithms. In Chapter 5, we showed that if the single-machine sequencing problem has a polynomial time β-approximation, then for any $\delta > 0$, the multi-machine sequencing problem has, in expectation, a polynomial time $\frac{1}{2}(\beta - \delta)$-approximation.

Is the $\frac{1}{2}(\beta - \delta)$-approximation tight? In [30], Fleischer et al. gave a $(1 - \frac{1}{e})\beta$-approximation for the separable assignment problem if the single-bin problem has a β-approximation. They also showed that the $1 - \frac{1}{e}$ lower bound is tight unless NP $\subseteq$ DTIME($n^{O(\log \log n)}$). In Section 5.5, we discuss the extent to which our approach is inspired by [30]. We also explain why the multi-machine sequencing problem is more general than the separable assignment problem that [30] solved and why a new rounding procedure is required. The question remains if the $\frac{1}{2}$

lower bound is tight for our model, or, if with with a different approach, a better approximation ratio can be achieved.

On the facets of the incremental knapsack polytope. In Chapter 6, we provide a class of facet defining inequalities for the incremental knapsack polytope, which we call the lift and push cover inequalities. Given that there are exponentially many of them, we also give separation algorithms for these inequalties so that they can be generated efficiently.

Complete characterization of facets with $0/1$*-coefficients?* We show in Theorem 6.3.2 that only one direction of Balas's result is extendable to the incremental knapsack polytope. Specifically, we show that extended cover inequalities of strong covers define facets of the incremental knapsack polytope through the lift and push technique. However, we also show in Example 6.3.3 that there exists facets with $0/1$-coefficients that cannot be lifted from strong covers. Due to these results, clearly, a complete classification of facets with $0/1$-coefficients for the incremental knapsack polytope will require further research, possibly needing different lifting techniques.

Integrality gap of the generalized incremental knapsack problem with lift and push cover inequalities added? Bienstock et al. showed that even with an `IK` objective, the IP formulation of (GIK-IP) has an unbounded integrality gap compared to its LP relaxation [7]. In particular, the integrality gap goes to infinity as T goes to infinity. This suggests that without strengthening (GIK-IP) with valid inequalities, classical LP rounding techniques, such as the one we use in Section 4.3.1, cannot produce a constant factor approximation when we let T be arbitrarily large. We remark that the instance with unbounded integrality gap shown in [7] has an integrality gap of 1 if we add all lift and push cover inequalities defined in Section 6.3. Thus, it remains an open question of what is the integrality gap when all lift and push cover inequalities are added. If there is a bounded integrality gap, can the lift and push cover inequalities and the approximate separation algorithm given in Section 6.4.2, combined with LP-rounding techniques, give an algorithm that either leads to a PTAS or improve upon the $(\frac{1}{2} - \epsilon)$-approximation we give in Section 3.2?

References

[1] D. Adjiashvili, S. Bosio, R. Weismantel, and R. Zenklusen, "Time-expanded packings," in *International Colloquium of Automata, Languages and Programming*, 2014, pp. 64–76.

[2] E. C. Akrida, J. Czyzowicz, L. Gasieniec, L. Kuszner, and P. G. Spirakis, "Temporal flows in temporal networks," *Journal of Computer and System Sciences*, vol. 103, pp. 46–60, 2019.

[3] A. Anagnostopoulos, F. Grandoni, S. Leonardi, and A. Wiese, "A mazing $2 + \epsilon$ approximation for unsplittable flow on a path," *ACM Transactions on Algorithms*, vol. 14, no. 4, 2018.

[4] A. Aouad and D. Segev, "Technical note-an approximate dynamic programming approach to the incremental knapsack problem," *Operations Research*, 2022.

[5] B. S. Baker, "Approximation algorithms for NP-complete problems on planar graphs," *Journal of the ACM*, vol. 41, no. 1, pp. 153–180, 1994.

[6] E. Balas, "Facets of the knapsack polytope," *Mathematical Programming*, vol. 8, pp. 146–164, 1975.

[7] D. Bienstock, J. Sethuraman, and C. Ye, *Approximation algorithms for the incremental knapsack problem via disjunctive programming*, arXiv preprint arXiv:1311.4563, 2013.

[8] N. Boland, T. Kalinowski, H. Waterer, and L. Zheng, "Scheduling arc maintenance jobs in a network to maximize total flow over time," *Discrete Applied Mathematics*, vol. 163, no. 1, pp. 34–52, 2014.

[9] P. Bonsma, J. Schulz, and A. Wiese, "A constant-factor approximation algorithm for unsplittable flow on paths," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 767–799, 2014.

[10] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz, "A tight linear time $(1/2)$-approximation for unconstrained submodular maximization," *SIAM Journal on Computing*, vol. 44, no. 5, pp. 1384–1402, 2015.

[11] N. Buchbinder, M. Feldman, J. S. Naor, and R. Schwartz, "Submodular maximization with cardinality constraints," ser. SODA '14, Portland, Oregon: Society for Industrial and Applied Mathematics, 2014, 1433–1452, ISBN: 9781611973389.

[12] G. Calinescu, A. Chakrabarti, H. Karloff, and Y. Rabani, "An improved approximation algorithm for resource allocation," *ACM Transactions on Algorithms*, vol. 7, no. 4, 2011.

[13] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[14] A. Caprara, "Packing 2-dimensional bins in harmony," in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, 2002, pp. 490–499.

[15] V. T. Chakaravarthy, A. R. Choudhury, S. Gupta, S. Roy, and Y. Sabharwal, "Improved algorithms for resource allocation under varying capacity," in *Proceedings for the 22nd Annual European Symposium on Algorithms*, 2014, pp. 222–234.

[16] C. Chekuri and S. Khanna, "A polynomial time approximation scheme for the multiple knapsack problem," *SIAM Journal on Computing*, vol. 35, no. 3, pp. 713–728, 2005.

[17] R. Cohen, L. Katzir, and D. Raz, "An efficient approximation for the generalized assignment problem," *Information Processing Letters*, vol. 100, no. 4, pp. 162–166, 2006.

[18] W. Cook, W. Cunningham, W. Pulleyblank, and A. Schrijver, *Combinatorial Optimization*, ser. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011, ISBN: 9781118031391.

[19] H. Crowder, E. L. Johnson, and M. Padberg, "Solving large-scale zero-one linear programming problems," *Operations Research*, vol. 31, no. 5, pp. 803–834, 1983.

[20] A. Del Pia, J. Linderoth, and H. Zhu, "On the complexity of separation from the knapsack polytope," in *Integer Programming and Combinatorial Optimization*, 2022, 168–180.

[21] F. Della Croce, U. Pferschy, and R. Scatamacchia, "Approximating the 3-period incremental knapsack problem," *Journal of Discrete Algorithms*, vol. 52, pp. 55–69, 2018.

[22] ——, "On approximating the incremental knapsack problem," *Discrete Applied Mathematics*, vol. 264, pp. 26–42, 2019.

[23] K. A. Dowsland and W. B. Dowsland, "Packing problems," *European Journal of Operational Research*, vol. 56, no. 1, pp. 2–14, 1992.

[24] L. Epstein, "On bin packing with clustering and bin packing with delays," *Discrete Optimization*, vol. 41, Article 100647, 2021.

[25] Y. Faenza and I. Malinovic, "A PTAS for the time-invariant incremental knapsack problem," in *Proceedings of the 5th International Symposium on Combinatorial Optimization*, 2018, pp. 157–169.

[26] Y. Faenza, D. Segev, and L. Zhang, "Approximation algorithms for the generalized incremental knapsack problem," *Mathematical Programming*, 2022.

[27] R. Z. Farahani, Z. Drezner, and N. Asgari, "Single facility location and relocation problem with time dependent weights and discrete planning horizon," *Annals of Operations Research*, vol. 167, pp. 353–368, 2009.

[28] U. Feige, "A threshold of ln n for approximating set cover," *J. ACM*, vol. 45, no. 4, 634–652, 1998.

[29] U. Feige and J. Vondrák, "The submodular welfare problem with demand queries," *Theory of Computing*, vol. 6, pp. 247–290, 2010.

[30] L. Fleischer, M. X. Goemans, V. S. Mirrokni, and M. Sviridenko, "Tight approximation algorithms for maximum separable assignment problems," *Mathematics of Operations Research*, vol. 36, no. 3, pp. 416–431, 2011.

[31] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956.

[32] A. M. Frieze, "A cost function property for plant location problems," *Mathematical Programming*, vol. 7, pp. 245–248, 1974.

[33] L. Graf, T. Harks, and L. Sering, "Dynamic flows with adaptive route choice," *Mathematical Programming*, vol. 183, 309–335, 2020.

[34] F. Grandoni, S. Ingala, and S. Uniyal, "Improved approximation algorithms for unsplittable flow on a path with time windows," in *Proceedings of the 13th International Workshop on Approximation and Online Algorithms*, 2015, pp. 13–24.

[35] F. Grandoni, T. Mömke, and A. Wiese, "A ptas for unsplittable flow on a path," in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, ser. STOC 2022, 2022, 289–302.

[36] F. Grandoni, T. Mömke, A. Wiese, and H. Zhou, "A $(5/3+\epsilon)$-approximation for unsplittable flow on a path: Placing small tasks into boxes," in *Proceedings of the 50th Annual ACM Symposium on Theory of Computing*, 2018, pp. 607–619.

[37] M. Groß, J.-P. W. Kappmeier, D. R. Schmidt, and M. Schmidt, "Approximating earliest arrival flows in arbitrary networks," in *Proceedings of the 20th Annual European Symposium on Algorithms*, 2012, pp. 551–562.

[38] M. Grotschel, L. Lovász, and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*. Springer, 1993.

[39] Z. Gu, G. L. Nemhauser, and M. W. P. Savelsbergh, "Lifted cover inequalities for 0-1 integer programs: Computation," *INFORMS Journal on Computing*, vol. 10, no. 4, pp. 427–437, 1998.

[40] J. R. K. Hartline, "Incremental optimization," Ph.D. dissertation, Department of Computer Science, Cornell University, 2008.

[41] D. S. Hochbaum and W. Maass, "Approximation schemes for covering and packing problems in image processing and VLSI," *Journal of the ACM*, vol. 32, no. 1, pp. 130–136, 1985.

[42] C. Hojny *et al.*, "Knapsack polytopes: A survey," *Annals of Operations Research*, vol. 292, pp. 469–517, 2020.

[43] A. Ismaili, "Routing games over time with FIFO policy," in *Proceedings of the 13th Conference on Web and Internet Economics*, 2017, pp. 266–280.

[44] C. Jin, "An improved FPTAS for 0-1 knapsack," in *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming*, 2019, 76:1–76:14.

[45] K. Kaparis and A. N. Letchford, "Separation algorithms for 0-1 knapsack polytopes," *Mathematical Programming*, vol. 124, pp. 69–91, 2010.

[46] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03, Association for Computing Machinery, 2003, 137–146, ISBN: 1581137370.

[47] G. Kim, E. P. Xing, L. Fei-Fei, and T. Kanade, "Distributed cosegmentation via submodular optimization on anisotropic diffusion," in *2011 International Conference on Computer Vision*, 2011, pp. 169–176.

[48] D Klabjan, G. Nemhauser, and C Tovey, "The complexity of cover inequality separation," *Operations Research Letters*, vol. 23, no. 1, pp. 35–40, 1998.

[49] F. Kojima, "Recent developments in matching theory and their practical applications," in *Advances in Economics and Econometrics: Eleventh World Congress*, ser. Econometric Society Monographs. Cambridge University Press, 2017, vol. 1, 138–175.

[50] A. Krause and D. Golovin, "Submodular function maximization.," *Tractability*, vol. 3, pp. 71–104, 2014.

[51] J. Lee, V. S. Mirrokni, V. Nagarajan, and M. Sviridenko, "Maximizing nonmonotone submodular functions under matroid or knapsack constraints," *SIAM Journal on Discrete Mathematics*, vol. 23, no. 4, pp. 2053–2078, 2010.

[52] M. Lin and P. Jaillet, "On the quickest flow problem in dynamic networks - A parametric min-cost flow approach," in *Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2015, pp. 1343–1356.

[53] S. Liu, "A review for submodular optimization on machine scheduling problems," in *Complexity and Approximation*, D.-Z. Du and J. Wang, Eds. Springer International Publishing, 2020, pp. 252–267, ISBN: 978-3-030-41672-0.

[54] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," *European Journal of Operational Research*, vol. 141, no. 2, pp. 241–252, 2002.

[55] S. Nickel and F. Saldanha-da Gama, "Multi-period facility location," in *Location Science*, G. Laporte, S. Nickel, and F. Saldanha da Gama, Eds., Springer International Publishing, 2019, pp. 303–326.

[56] Z. Nutov, I. Beniaminy, and R. Yuster, "A $(1 - 1/e)$-approximation algorithm for the generalized assignment problem," *Operations Research Letters*, vol. 34, no. 3, pp. 283–288, 2006.

[57] T. Öncan, "A survey of the generalized assignment problem and its applications," *INFOR: Information Systems and Operational Research*, vol. 45, no. 3, pp. 123–141, 2007.

[58] M. Padberg, "(1,k)-configurations and facets for packing problems," *Mathematical Programming*, vol. 18, pp. 94–99, 1980.

[59] A. M. Sharp, "Incremental algorithms: Solving problems in a changing world," Ph.D. dissertation, Department of Computer Science, Cornell University, 2007.

[60] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem," *Mathematical Programming*, vol. 62, pp. 461–474, 1993.

[61] M. Skutella, "An introduction to network flows over time," in *Research Trends in Combinatorial Optimization*, W. Cook, L. Lovász, and J. Vygen, Eds., Springer, 2009, pp. 451–482.

[62] M. Sviridenko, "A note on maximizing a submodular set function subject to a knapsack constraint," *Operations Research Letters*, vol. 32, no. 1, pp. 41–43, 2004.

[63] T. J. Van Roy and L. A. Wolsey, "Solving mixed integer programming problems using automatic reformulation," *Operations Research*, vol. 35, no. 1, pp. 45–57, 1987.

[64] C. Ye, "On the trade-offs between modeling power and algorithmic complexity," Ph.D. dissertation, Columbia University, 2016.

[65] E. Zemel, "Easily computable facets of the knapsack polytope," *Mathematics of Operations Research*, vol. 14, no. 4, pp. 760–764, 1989.

Appendix A: Incremental packing problems

A.1 Reduction to Unsplittable Flow on a Path with Bag Constraints

As a first attempt of reducing the generalized incremental knapsack problem to unsplittable flow on a path (see Section 1.4.4), we could construct a path over the sequence of vertices $0, \ldots, T$, where each edge $(t, t+1)$ has a capacity of W_t. Then, for each item-time pair (i, t), one could create a corresponding task to capture the decision of inserting item i at time period t; this task would extend across the subpath $\langle t, \ldots, T \rangle$, with a profit of p_{it} and a demand of w_i. However, as each item i may be inserted into the knapsack only once, we are lacking an additional constraint, stating that at most one of the tasks corresponding to $\{(i, t)\}_{t \in [T]}$ can be picked, which makes the resulting problem fundamentally different.

To capture this additional constraint, consider a generalization of the unsplittable flow on a path problem with "bag constraints", first studied by Chakaravarthy et al. [15]. In this setting, tasks are further partitioned into a set of bags $\mathcal{B}_1, \ldots, \mathcal{B}_k$, and we are allowed to pick at most one task from each bag. This way, we can model the generalized incremental knapsack problem by having a separate bag $\mathcal{B}_i$ for each item $i \in [n]$, which contains all tasks corresponding to $\{(i, t)\}_{t \in [T]}$, thereby capturing the extra feature that each item may be inserted only once. For this generalization, Grandoni et al. [34] proposed an $O(\frac{\log \log n}{\log n})$-approximation through an LP-based rounding approach, which is currently the best known performance guarantee.

Appendix B: Single-time policies for the generalized incremental knapsack problem

B.1 Proof of Lemma 2.1.1

Given an instance $\mathcal{I}$ of the generalized incremental knapsack problem, we construct another instance $\mathcal{I}'$ defined over the same set of items with profit p' as follows: for every $i \in [n]$ and $t \in [T]$, let $p'_{i,t} = \max_{\tau \in [T], \tau \geq t} p_{i,\tau}$. Capacity and weight parameters remain unchanged. The profit of any feasible chain of $\mathcal{S}'$ in $\mathcal{I}'$ is therefore given by $\Phi'(\mathcal{S}') = \sum_{t \in [T]} \sum_{i \in S_t \setminus S_{t-1}} p'_{i,t}$. Clearly profit p' satisfies the property that $p'_{i,t} \geq p'_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$.

Let $\mathcal{S}$ be a feasible chain of $\mathcal{I}$. Clearly $\mathcal{S}$ is feasible in $\mathcal{I}'$. To see $\Phi'(\mathcal{S}) \geq \Phi(\mathcal{S})$, note that

$$\begin{aligned} \Phi'(\mathcal{S}) &= \sum_{t \in [T]} \sum_{i \in S_t \setminus S_{t-1}} p'_{i,t} \\ &\geq \sum_{t \in [T]} \sum_{i \in S_t \setminus S_{t-1}} p_{i,t} \\ &= \Phi(\mathcal{S}). \end{aligned}$$

In the formulas above, the only inequality follows since $p'_{i,t} = \max_{\tau \in [T], \tau \geq t} p_{i,\tau}$.

Conversely, take any feasible chain $\mathcal{S}'$ of $\mathcal{I}'$. For any $i \in S'_T$, let $t'(i)$ denote the insertion time of item i with respect to the chain $\mathcal{S}'$. For any $i \in S'_T$, we construct $\mathcal{S}$ by setting the insertion time of i to be $t(i) \in \operatorname{argmax}_{\tau \in [T], \tau \geq t'(i)} p_{i,\tau}$.

In the above construction, $S_t \subseteq S'_t$ for every $t \in [T]$. Therefore

$$w(S_t) \leq w(S'_t) \leq W_t.$$

Hence $\mathcal{S}$ is feasible in $\mathcal{I}$. By construction, for every item $i \in S'_T$, item i earns the exact same

profit in $\mathcal{I}'$ and $\mathcal{I}$, therefore $\Phi(\mathcal{S}) = \Phi'(\mathcal{S}')$.

B.2 The fully rigid algorithm may output a solution with an arbitrarily bad approximation ratio

Consider a generalized incremental knapsack instance with $T = 2$ and $n = 2$. Let $W_1 = 1$ and $W_2 = 2$. For item $i = 1$ and for $t \in [T]$, let $p_{i,t} = 1$. For item $i = 2$ and for $t \in [T]$, let $p_{i,t} = M$ for an arbitrarily large integer M. Let $w_1 = 1$ and $w_2 = 2$. It is easy to see the optimal solution outputs $\mathcal{S}^* = (\emptyset, \{2\})$, earning a total profit of M.

In $t = 1$, the fully rigid algorithm sets $S_1 = \{1\}$, since item 1 is the only item that is feasible. In $t = 2$, the fully rigid algorithm solves the knapsack problem with item 2 and capacity $W_2 - w(S_1) = 1$. Since item 2 is infeasible, it returns back solution $Q_2 = \emptyset$ and outputs final solution $\mathcal{S} = (\{1\}, \{1\})$, with a profit of 1, giving an arbitrarily bad approximation ratio.

B.3 The fully flexible algorithm may output an $O(\frac{1}{T})$-approximated solution

Consider a genearlized incremental knapsack instance with T times and $n = 2T$ items. Let $W_t = \sum_{\tau=1}^{t} T^\tau + T\epsilon$ for all $t \in [T]$. For $i \in [T]$, item i has profit $p_{i,t} = 1$ if $t \leq i$ and profit $p_{i,t} = 0$ otherwise. Each item $i \in [T]$ has weight $w_i = T^i + \epsilon$. For $i \in [T]$, item $T+i$ has profit $p_{T+i,t} = 1 + i\epsilon$ if $t \leq i$ and profit $p_{T+i,t} = 0$ otherwise. Each item $T + i$ has weight $w_{T+i} = W_i$.

A feasible chain to this instance is $\mathcal{S}^* = (S_1^*, \ldots, S_T^*)$ where $S_t^* = \{i : i \leq t\}$ for all $t \in [T]$, with an optimal profit of T. We will show by induction on $t \in [T]$ that, at the end of round t, the fully flexible algorithm gives the solution $\mathcal{S} = (S_1, \ldots, S_T)$ where $S_\tau = \emptyset$ for all $\tau \leq t - 1$ and $S_\tau = \{T + t\}$ for all $t \leq \tau \leq T$. That is, in any round t, the algorithm inserts item $T + t$ starting in time t and inserts no item otherwise. This statement implies that at the end of the algorithm, we get the solution $\mathcal{S} = (S_1, \ldots, S_T)$ where $S_t = \emptyset$ for $t \leq T - 1$ and $S_T = \{2T\}$, with a total profit of $1 + T\epsilon$. Taking ϵ sufficiently small, this gives an $O(\frac{1}{T})$-approximated solution.

For the base case $t = 1$, notice that only items 1 and $T + 1$ are (individually) feasible in the knapsack problem. Since item $T+1$ has strictly more profit than item 1, the fully flexible algorithm

may take item $T + 1$.

Now for the general case $t \geq 2$, assume by inductive hypothesis that at the end of round $t - 1$, the algorithm gives the chain $(S_1, \ldots, S_T)$ where $S_\tau = \emptyset$ for $\tau \leq t - 2$ and $S_\tau = \{T + t - 1\}$ for $\tau \geq t - 1$. Thus, in the knapsack problem in round t, for any $i \in [t - 1]$, item i has profit 0; for any $i \in [t - 2]$, item $T + i$ has profit 0; item $T + t - 1$ has profit $1 + (t - 1)\epsilon$, item t has profit 1 and item $T + t$ has profit $1 + t\epsilon$. All other items are infeasible. Thus, the optimal solution to the knapsack problem must be a subset of items $\{t, T + t - 1, T + t\}$. It is easy to check that no subset of cardinality 2 is feasible. Thus, since item $T + t$ is the most profitable item in the set, the optimal is to simply take item $T + t$. Therefore, at the end of round t, we have $S_\tau = \emptyset$ for $\tau \leq t - 1$ and $S_\tau = \{T + t\}$ for $\tau \geq t$. This concludes the proof.

B.4 Additional proofs from Chapter 2

B.4.1 Proof of Lemma 2.3.1

We will prove the statement by induction on t. For the base case $t = 1$, from Step 1c, notice that $\mathrm{ALG}_\tau^{(1)} = \mathrm{ALG}_{\tau+1}^{(1)}$ for all $\tau \in [T - 1]$. Thus, $\mathrm{ALG}^{(1)}$ is a chain. Furthermore, by the capacity constraint of $\Pi^{(t)}$, $w(S) \leq W_1 \leq W_\tau$ for every $\tau \in [T]$. Thus, $\mathrm{ALG}^{(1)}$ is feasible.

For the general case of $t \geq 2$, we can assume without loss of generality $p'(S) \geq p'(\mathrm{ALG}_{t-1}^{(t-1)})$, else $\mathrm{ALG}^{(t)} = \mathrm{ALG}^{(t-1)}$ and the statement is true by inductive hypothesis. In this case $\mathrm{ALG}^{(t)} = (\mathrm{ALG}_1^{(t-1)} \cap S, \ldots, \mathrm{ALG}_{t-1}^{(t-1)} \cap S, \underbrace{S, \ldots, S}_{T-t+1 \text{ times}})$ and

$$(\mathrm{ALG}_1^{(t-1)} \cap S) \subseteq \cdots \subseteq (\mathrm{ALG}_{t-1}^{(t-1)} \cap S) \subseteq \underbrace{S \subseteq \cdots \subseteq S}_{T-t+1 \text{ times}},$$

where the first $t - 1$ inclusions follow from the inductive hypothesis, and the remaining inclusions follow by construction. Thus $\mathrm{ALG}^{(t)}$ is a chain. To prove feasibility, for $\tau \in [t - 1]$, notice that $w(\mathrm{ALG}_\tau^{(t-1)} \cap S) \leq w(\mathrm{ALG}_\tau^{(t-1)}) \leq W_\tau$, where the second inequality follows from inductive hypothesis. For $\tau \geq t$, feasibility follows since $w(S) \leq W_t \leq W_\tau$. Thus, $\mathrm{ALG}^{(t)}$ is feasible.

B.4.2 Proof of Lemma 2.4.4

Notice that, for any $t \in [T]$ with $t \geq 2$ and $j = 0$, since $\text{ALG}_j^{(t)} = \emptyset$, the statement is trivial. For $j \in [t-1]$, this follows immediately from Step 1d. Notice that either $\text{ALG}_j^{(t)} = \text{ALG}_j^{(t-1)}$, or $\text{ALG}_j^{(t)} = \text{ALG}_j^{(t-1)} \cap S$, where S is the solution to $\Pi^{(t)}$. Then clearly $\text{ALG}_j^{(t)} \subseteq \text{ALG}_j^{(t-1)}$.

B.4.3 Proof of Lemma 2.4.5

For any $j, t \in [T]$ and $t \geq 2$, let $i \in R_j^{(t)}$. We will show that $i \in R_{j+1}^{(t)}$. If $j + 1 \geq t$, the proof is trivial since $R_{j+1}^{(t)} = R_j^{(t)}$ by definition. Thus, we consider $j + 1 < t$.

By definition, $i \in \text{ALG}_j^{(t-1)} \setminus \text{ALG}_j^{(t)}$. Recall that $\text{ALG}^{(t-1)}$ is a chain by Lemma 2.3.1. Then $i \in \text{ALG}_j^{(t-1)}$ implies $i \in \text{ALG}_{j+1}^{(t-1)}$. If $i \notin \text{ALG}_{j+1}^{(t)}$, then $i \in \text{ALG}_{j+1}^{(t-1)} \setminus \text{ALG}_{j+1}^{(t)} = R_{j+1}^{(t)}$ and we are done.

Thus, assume $i \in \text{ALG}_{j+1}^{(t)}$. We show that this case leads to a contradiction, concluding the proof. We first claim that $i \notin \text{ALG}_j^{(j+1)}$. By Lemma 2.4.4, since $j + 1 < t$, we have $\text{ALG}_{j+1}^{(t)} \subseteq \text{ALG}_{j+1}^{(t-1)} \subseteq \cdots \subseteq \text{ALG}_{j+1}^{(j+1)}$. Hence, $i \in \text{ALG}_{j+1}^{(t)}$ implies $i \in \text{ALG}_{j+1}^{(j+1)}$. Furthermore, $i \notin \text{ALG}_j^{(t)}$. Thus, if $i \in \text{ALG}_j^{(j+1)}$, this means $i \notin \text{ALG}_j^{(\tau)}$ for at some round $j + 1 < \tau \leq t$, so by Step 1d of the algorithm, $i \notin \text{ALG}_{j+1}^{(\tau)}$ as well. Thus $i \notin \text{ALG}_{j+1}^{(j+1)}$, a contradiction.

Hence, $i \notin \text{ALG}_j^{(j+1)}$. Again, by Lemma 2.4.4, since $\text{ALG}_j^{(t-1)} \subseteq \cdots \subseteq \text{ALG}_j^{(j+1)}$, $i \notin \text{ALG}_j^{(j+1)}$ implies $i \notin \text{ALG}_j^{(t-1)}$. We have arrived at a contradiction since by definition $i \in \text{ALG}_j^{(t-1)}$.

B.4.4 Auxiliary lemmas

Before proceeding with the remaining proofs, we provide some useful lemmas, which will be used throughout the proofs in the remainder of this section. The following lemma gives a formula for the profit $\mathcal{R}^{(t)}$ earns at every time as a function of the profit earned by the solutions produced by the algorithm.

Lemma B.4.1. *For every $t \in [T]$, $t \geq 2$ and $j \in [t-1]$,*

$$p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}) = p_j(\mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}) - p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}).$$

Proof. First consider the case where $t \in [T], t \geq 2$, $j = 1$. Recalling that $R_0^{(t)} = \mathrm{ALG}_0^{(t-1)} = \mathrm{ALG}_0^{(t)} = \emptyset$ by definition, the statement reduces to $p_1(R_1^{(t)}) = p_1(\mathrm{ALG}_1^{(t-1)}) - p_1(\mathrm{ALG}_1^{(t)})$. To see this equality holds, note

$$p_1(R_1^{(t)}) = p_1(\mathrm{ALG}_1^{(t-1)} \setminus \mathrm{ALG}_1^{(t)}) = p_1(\mathrm{ALG}_1^{(t-1)}) - p_1(\mathrm{ALG}_1^{(t)}),$$

where the first equality holds by definition and the second equality holds by noting that $\mathrm{ALG}_1^{(t)} \subseteq \mathrm{ALG}_1^{(t-1)}$ by Lemma 2.4.4.

Now, assume $2 \leq j \leq t-1$. Let

$$A = \mathrm{ALG}_j^{(t-1)}, \quad B = \mathrm{ALG}_{j-1}^{(t-1)}, \quad C = \mathrm{ALG}_j^{(t)}, \quad D = \mathrm{ALG}_{j-1}^{(t)},$$

we have, by definition:

$$R_j^{(t)} \setminus R_{j-1}^{(t)} = (A \setminus C) \setminus (B \setminus D), \quad ALG_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)} = A \setminus B, \quad \mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)} = C \setminus D.$$

$D \subseteq B$ follows by Lemma 2.4.4 since $j \leq t-1$. Let $i \in \mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}$. Since $j \leq t-1$, following Step 1d of the algorithm, we have $\mathrm{ALG}_j^{(t)} = \mathrm{ALG}_j^{(t-1)} \cap S$ and $\mathrm{ALG}_{j-1}^{(t)} = \mathrm{ALG}_{j-1}^{(t-1)} \cap S$ for some $S \subseteq [n]$. Hence,

$$C \setminus D = \mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)} = (\mathrm{ALG}_j^{(t-1)} \cap S) \setminus (\mathrm{ALG}_{j-1}^{(t-1)} \cap S) \subseteq \mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)} = A \setminus B. \quad \text{(B.1)}$$

Moreover, since by Lemma 2.3.1, $\mathrm{ALG}^{(t)}$ is a chain, we have

$$A \cap D = \mathrm{ALG}_j^{(t-1)} \cap \mathrm{ALG}_{j-1}^{(t)} \subseteq \mathrm{ALG}_j^{(t-1)} \cap \mathrm{ALG}_j^{(t)} \subseteq \mathrm{ALG}_j^{(t)} = C. \quad \text{(B.2)}$$

We deduce:

$$\begin{aligned}(A \setminus C) \setminus (B \setminus D) &= (A \setminus C) \setminus B \\ &= (A \setminus B) \setminus C \\ &= (A \setminus B) \setminus (C \setminus D),\end{aligned}$$

where the first equality follows from (B.2) and the last equality follows from $D \subseteq B$. We conclude

$$p_j((A \setminus C) \setminus (B \setminus D)) = p_j((A \setminus B) \setminus (C \setminus D)) = p_j(A \setminus B) - p_j(C \setminus D),$$

where the last equality follows from (B.1). □

The next lemma bounds the differences $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})$ in terms of the profits of $\mathrm{ALG}^{(t)}$, OPT, and $\mathcal{R}^{(t)}$.

Lemma B.4.2. *For $t \in [T]$, $t \geq 2$, we have $\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)}) \leq -a_t + b_t + d_t + \ell_t$, where $\ell_t = p_t(\mathrm{OPT}_t \setminus (\mathrm{OPT}_{t-1} \cup \mathrm{ALG}_{t-1}^{(t-1)}))$.*

Proof. Fix $t \in [T], t \geq 2$. By defining, for the appropriate indices j, $\gamma_j, \beta_j, \alpha_j, \lambda_j$ as in the equation below, using the definition of $\mathrm{HB}^{(t-1)}$ and $\mathrm{HB}^{(t)}$ we have:

$$\begin{aligned}&\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)}) \\ &\quad= \sum_{j=1}^{t-1} \underbrace{p_j(\mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)})}_{\gamma_j} + \sum_{j=t}^{T} \underbrace{p_j(\mathrm{OPT}_j \setminus (\mathrm{OPT}_{j-1} \cup \mathrm{ALG}_{t-1}^{(t-1)}))}_{\beta_j} \\ &\quad\quad- \left(\sum_{j=1}^{t} \underbrace{p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)})}_{\alpha_j} + \sum_{j=t+1}^{T} \underbrace{p_j(\mathrm{OPT}_j \setminus (\mathrm{OPT}_{j-1} \cup \mathrm{ALG}_t^{(t)}))}_{\lambda_j} \right) \qquad \text{(B.3)}\end{aligned}$$

We now relate the right-hand side of (B.3) to a_t, b_t, d_t, ℓ_t. First note that $\alpha_t = a_t$ and that by Lemma B.4.1, for $j \in [t-1]$, we have $\gamma_j - \alpha_j = p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})$. Hence,

$$\sum_{j=1}^{t-1} \gamma_j - \sum_{j=1}^{t} \alpha_j = \sum_{j=1}^{t-1} (\gamma_j - \alpha_j) - \alpha_t = \sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}) - a_t = b_t - a_t. \qquad \text{(B.4)}$$

Further note that

$$\begin{aligned} \mathrm{ALG}_{t-1}^{(t-1)} \setminus \mathrm{ALG}_{t}^{(t)} &= (\mathrm{ALG}_{t-1}^{(t-1)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \setminus (\mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \\ &= R_{t-1}^{(t)}, \end{aligned} \tag{B.5}$$

where the first equality follows by recalling that $\mathrm{ALG}^{(t)}$ is a chain, and the second equality follows by definition of $R_{t-1}^{(t)}$ and by noting that $(\mathrm{ALG}_{t-1}^{(t-1)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \cap (\mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}) = \emptyset$ from Step 1d of the algorithm.

On the other hand, observe that

$$\begin{aligned} \mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t-1)} &= (\mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \setminus (\mathrm{ALG}_{t-1}^{(t-1)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \\ &= \mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}, \end{aligned} \tag{B.6}$$

where the first equality follows by observing that by Lemma 2.4.4, $\mathrm{ALG}_{t-1}^{(t)} \subseteq \mathrm{ALG}_{t-1}^{(t-1)}$. The second equality follows again since $(\mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \cap (\mathrm{ALG}_{t-1}^{(t-1)} \setminus \mathrm{ALG}_{t-1}^{(t)}) = \emptyset$.

Therefore, for $j \in \{t+1, \ldots, T\}$, defining e_j and f_j as below, we have

$$\begin{aligned} \beta_j - \lambda_j &= p_j(\mathrm{OPT}_j \setminus (\mathrm{OPT}_{j-1} \cup \mathrm{ALG}_{t-1}^{(t-1)})) - p_j(\mathrm{OPT}_j \setminus (\mathrm{OPT}_{j-1} \cup \mathrm{ALG}_{t}^{(t)})) \\ &= -p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap \mathrm{ALG}_{t-1}^{(t-1)}) + p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap \mathrm{ALG}_{t}^{(t)}) \\ &= -p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_{t-1}^{(t-1)} \setminus \mathrm{ALG}_{t}^{(t)})) \\ &\quad + p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t-1)})) \\ &= -\underbrace{p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap R_{t-1}^{(t)})}_{e_j} + \underbrace{p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_{t}^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}))}_{f_j}, \end{aligned}$$

where the first two equalities follow from set arithmetics, and the third equality follows by applying (B.5) and (B.6). Therefore, for $j \in \{t+1, \ldots, T\}$, we have $\beta_j - \lambda_j = -e_j + f_j$. By observing

$\beta_t = \ell_t$, we have

$$\sum_{j=t}^{T} \beta_j - \sum_{j=t+1}^{T} \lambda_j = \ell_t + \sum_{j=t+1}^{T} (-e_j + f_j). \tag{B.7}$$

Lastly, observe

$$\sum_{j=t+1}^{T} f_j = d_t. \tag{B.8}$$

The lemma then follows by plugging (B.4), (B.7), (B.8) into (B.3) and dropping $-\sum_{j=t+1}^{T} e_j \leq 0$. □

Finally, we provide the following equivalent interpretation of δ_t:

Lemma B.4.3. *For all $t \in [T]$, $t \geq 2$, we have:*

$$\delta_t = a_t + c \sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}) - c \sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}).$$

Proof. Recall, that, in the t-th iteration of Step 1d, the algorithm decides whether to set $\mathrm{ALG}^{(t)} = \mathrm{ALG}^{(t-1)}$, or to update $\mathrm{ALG}^{(t)}$ with the solution to $\Pi^{(t)}$. If the former holds, then both the left- and right-hand side of the equality from the thesis of the lemma are equal to 0. So suppose this is not the case. With respect to the objective function of $\Pi^{(t)}$, the profit of $\mathrm{ALG}_{t-1}^{(t-1)}$ is

$$\begin{aligned} \sum_{i \in \mathrm{ALG}_t^{(t-1)} \setminus \mathrm{ALG}_{t-1}^{(t-1)}} p_{i,t} + c \sum_{j=1}^{t-1} \sum_{i \in \mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}} p_{i,j} &= c \sum_{j=1}^{t-1} \sum_{i \in \mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}} p_{i,j} \\ &= c \sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}), \end{aligned} \tag{B.9}$$

where the first equality is due to that $\mathrm{ALG}_t^{(t-1)} \setminus \mathrm{ALG}_{t-1}^{(t-1)} = \emptyset$ by construction, and the second follows by the definition. Similarly, with respect to the objective function of $\Pi^{(t)}$, the profit of

$\mathrm{ALG}_t^{(t)}$ is

$$\sum_{i\in \mathrm{ALG}_t^{(t)}\setminus \mathrm{ALG}_{t-1}^{(t)}} p_{i,t} + c\sum_{j=1}^{t-1}\sum_{i\in \mathrm{ALG}_j^{(t)}\setminus \mathrm{ALG}_{j-1}^{(t)}} p_{i,j} = \underbrace{p_t(\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})}_{a_t} + c\sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}). \tag{B.10}$$

Taking the difference between (B.10) and (B.9) and applying Lemma B.4.1,

$$a_t + c\sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}) - c\sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}) = a_t - c\underbrace{\sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})}_{b_t} = \delta_t.$$

□

From Lemma B.4.3 and Step 1d of the algorithm, for $t \in [T]$, $t \geq 2$, we have that δ_t is the maximum of 0 and the difference between the profits of $\mathrm{ALG}_t^{(t)}$ and $\mathrm{ALG}_{t-1}^{(t-1)}$ in $\Pi^{(t)}$. Recalling that $\delta_1 = a_1 - cb_1 = a_1$, we deducing the following.

Observation B.4.4. *For all $t \in [T]$, we have $\delta_t \geq 0$.*

B.4.5 Proof of Lemma 2.4.6

Recall that by definition $\mathrm{OPT}_0 = \mathrm{ALG}_0^{(1)} = \emptyset$, we have:

$$\begin{aligned}
\Phi(\mathrm{HB}^{(0)}) - \Phi(\mathrm{HB}^{(1)}) &= \sum_{j=1}^{T} p_j(\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) - p_1(\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)}) \\
&\quad - \sum_{j=2}^{T} p_j(\mathrm{OPT}_j \setminus (\mathrm{OPT}_{j-1} \cup \mathrm{ALG}_1^{(1)})) \\
&= p_1(\mathrm{OPT}_1) - p_1(\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)}) \\
&\quad + \sum_{j=2}^{T} p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)})) \\
&\leq (1+\epsilon)p_1(\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)}) - p_1(\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)}) \\
&\quad + \sum_{j=2}^{T} p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)}))
\end{aligned}$$

$$= \epsilon \underbrace{p_1(\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)})}_{a_1} + \underbrace{\sum_{j=2}^{T} p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)}))}_{d_1},$$

where the first equality follows by definition and the other two by simple algebraic calculations. The inequality follows since OPT_1 is clearly feasible in $\Pi^{(1)}$, and since $\Pi^{(1)}$ is solved to $(1 - \frac{\epsilon}{1+\epsilon})$-optimality, we have $p_1(\mathrm{OPT}_1) \leq (1+\epsilon)p_1(\mathrm{ALG}_1^{(1)}) = (1+\epsilon)p_1(\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)})$.

B.4.6 Proof of Lemma 2.4.7

$$\begin{aligned}
\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)}) &\leq b_t - a_t + d_t + \underbrace{p_t(\mathrm{OPT}_t \setminus (\mathrm{OPT}_{t-1} \cup \mathrm{ALG}_{t-1}^{(t-1)}))}_{\ell_t} \\
&\leq b_t - a_t + d_t + (1+\epsilon)(a_t - cb_t) \\
&= \epsilon a_t - (c-1)b_t + d_t - \epsilon c b_t \\
&\leq \epsilon a_t - (c-1)b_t + d_t,
\end{aligned}$$

where the first inequality follows from Lemma B.4.2 and the last from $b_t \geq 0$. As for the second, notice that, since $t \notin \mathcal{B}$, $t \geq 2$, we have:

$$\begin{aligned}
\ell_t &= p_t(\mathrm{OPT}_t \setminus (\mathrm{OPT}_{t-1} \cup \mathrm{ALG}_{t-1}^{(t-1)})) \\
&\leq p_t(\mathrm{OPT}_t \setminus \mathrm{OPT}_{t-1}) \\
&\leq (1+\epsilon)\delta_t \\
&= (1+\epsilon)(a_t - cb_t).
\end{aligned}$$

B.4.7 Proof of Lemma 2.4.8

To prove Lemma 2.4.8, we first need the following auxiliary lemma. Roughly speaking, the lemma gives an upper bound to the profit of certain items first inserted by OPT at time $t_i \in \mathcal{B}$ (the term ℓ_{t_i} below) in terms of the profit of items inserted or removed by the algorithm in certain previous times.

Lemma B.4.5. *For all $i \in [k]$, $t_i \in \mathcal{B}$, we have:*

$$(1+\epsilon)(c\sum_{j=t_{i-1}}^{t_i-1} a_j + a_{t_i}) + c\epsilon\Phi(\mathrm{ALG}^{(t_i)}) - cb_{t_i} \geq \underbrace{p_{t_i}(\mathrm{OPT}_{t_i} \setminus (\mathrm{OPT}_{t_i-1} \cup \mathrm{ALG}_{t_i-1}^{(t_i-1)}))}_{\ell_{t_i}}.$$

Proof. We first investigate the case $i \geq 2$. Hence, fix such an i and abbreviate $t' = t_{i-1}$. Since $t' \in \mathcal{B}$, by Observation B.4.4 we have

$$p_{t'}(\mathrm{OPT}_{t'} \setminus \mathrm{OPT}_{t'-1}) > (1+\epsilon)\delta_{t'} \geq 0.$$

Hence $w(\mathrm{OPT}_{t'}) > W_{t'-1}$, else $\mathrm{OPT}_{t'}$ is feasible in time $t'-1$, a contradiction since OPT is inclusionwise maximal among the optimal solutions. Then by feasibility of ALG, we have $w(\mathrm{ALG}_{t'-1}^{(t'-1)}) \leq W_{t'-1} < w(\mathrm{OPT}_{t'})$. Then by defining A as in the inequality below, we have:

$$\begin{aligned} w(\underbrace{\mathrm{ALG}_{t'-1}^{(t'-1)} \cup (\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1})}_{A}) &\leq w(\mathrm{ALG}_{t'-1}^{(t'-1)}) + w(\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1}) \\ &< w(\mathrm{OPT}_{t'}) + w(\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1}) \\ &\leq w(\mathrm{OPT}_{t_i-1}) + w(\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1}) \\ &\leq W_{t_i}, \end{aligned}$$

by feasibility of OPT and $t' = t_{i-1} \leq t_i - 1$. Said otherwise, the weight of A satisfies the knapsack constraint at time t_i. Since $t_i - 1 \geq t' \geq t_1 \geq 2$, we can repeatedly apply Lemma 2.4.4 and deduce

$\mathrm{ALG}_j^{(t_i-1)} \subseteq \mathrm{ALG}_j^{(t'-1)}$ for all $j \in [t'-1]$. Therefore, defining B as in the inequality below,

$$w(\underbrace{\mathrm{ALG}_{t'-1}^{(t_i-1)} \cup (\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1})}_{B}) \leq w(\underbrace{\mathrm{ALG}_{t'-1}^{(t'-1)} \cup (\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1})}_{A}) < W_{t_i},$$

hence B also satisfies the knapsack constraint at time t_i. Therefore, B is a feasible solution for algorithm $\Pi^{(t_i)}$. Recalling that $\Pi^{(t_i)}$ chooses solution $\mathrm{ALG}^{(t_i)}$ and that the profit of $\mathrm{ALG}^{(t_i)}$ in $\Pi^{(t_i)}$ is

$$D = c\sum_{j=1}^{t_i-1} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)}) + p_{t_i}(\mathrm{ALG}_{t_i}^{(t_i)} \setminus \mathrm{ALG}_{t_i-1}^{(t_i)}),$$

we define F as below and deduce

$$\begin{aligned}
(1+\epsilon)D &\geq p(B) \\
&= c\sum_{j=1}^{t'-1} p_j(\mathrm{ALG}_j^{(t_i-1)} \setminus \mathrm{ALG}_{j-1}^{(t_i-1)}) \\
&\quad + c\sum_{j=t'}^{t_i-1} p_j((\mathrm{ALG}_j^{(t_i-1)} \setminus \mathrm{ALG}_{j-1}^{(t_i-1)}) \cap (\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1})) \\
&\quad + p_{t_i}(\mathrm{OPT}_{t_i} \setminus (\mathrm{OPT}_{t_i-1} \cup \mathrm{ALG}_{t_i-1}^{(t_i-1)})) \\
&\geq \underbrace{c\sum_{j=1}^{t'-1} p_j(\mathrm{ALG}_j^{(t_i-1)} \setminus \mathrm{ALG}_{j-1}^{(t_i-1)})}_{F} + \underbrace{p_{t_i}(\mathrm{OPT}_{t_i} \setminus (\mathrm{OPT}_{t_i-1} \cup \mathrm{ALG}_{t_i-1}^{(t_i-1)}))}_{\ell_{t_i}}, \qquad \text{(B.11)}
\end{aligned}$$

where the first inequality follows since we solve $\Pi^{(t_i)}$ with an FPTAS for knapsack, and the equality holds by recalling that $\Pi^{(t_i)}$ multiplies by a factor c items that are in $\mathrm{ALG}_j^{(t_i-1)}$ for any $j \in [t_i-1]$. In this case, those are the items first introduced by $\mathrm{ALG}^{(t_i-1)}$ in times $1, \ldots, t'-1$, plus items introduced by $\mathrm{ALG}^{(t_i-1)}$ in times $t', \ldots, t_i-1$ that are also in $\mathrm{OPT}_{t_i} \setminus \mathrm{OPT}_{t_i-1}$.

Using Lemma B.4.1 and defining G and H,

$$\underbrace{\sum_{j=1}^{t'-1} p_j(\mathrm{ALG}_j^{(t_i-1)} \setminus \mathrm{ALG}_{j-1}^{(t_i-1)})}_{F} = \underbrace{\sum_{j=1}^{t'-1} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)})}_{G} + \underbrace{\sum_{j=1}^{t'-1} p_j(R_j^{(t_i)} \setminus R_{j-1}^{(t_i)})}_{H}. \qquad \text{(B.12)}$$

Subtracting cF from the left-hand side of (B.11), using (B.12), and defining I as below gives:

$$\begin{aligned}
(1+\epsilon)D - cF &= (1+\epsilon)(c\sum_{j=1}^{t_i-1} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)}) + p_{t_i}(\mathrm{ALG}_{t_i}^{(t_i)} \setminus \mathrm{ALG}_{t_i-1}^{(t_i)})) - cF \\
&= (1+\epsilon)c\underbrace{\sum_{j=1}^{t'-1} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)})}_{G} + (1+\epsilon)c\underbrace{\sum_{j=t'}^{t_i-1} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)})}_{I} \\
&\quad + (1+\epsilon)\underbrace{p_{t_i}(\mathrm{ALG}_{t_i}^{(t_i)} \setminus \mathrm{ALG}_{t_i-1}^{(t_i)})}_{a_{t_i}} - cG - cH \\
&= c\epsilon G + (1+\epsilon)cI + (1+\epsilon)a_{t_i} - cH,
\end{aligned}$$

while subtracting cF from the right-hand side of (B.11) gives $cF + \ell_{t_i} - cF = \ell_{t_i}$. Hence,

$$c\epsilon G + (1+\epsilon)cI + (1+\epsilon)a_{t_i} - cH \geq \ell_{t_i}. \tag{B.13}$$

We need some more intermediate steps to manipulate the left-hand side of (B.13). By repeatedly applying Lemma B.4.1 for $t' \leq j \leq t_i - 1$, we have:

$$\begin{aligned}
p_j(\mathrm{ALG}_j^{(j)} \setminus \mathrm{ALG}_{j-1}^{(j)}) &= p_j(\mathrm{ALG}_j^{(j+1)} \setminus \mathrm{ALG}_{j-1}^{(j+1)}) + p_j(R_j^{(j+1)} \setminus R_{j-1}^{(j+1)}) \\
&= p_j(\mathrm{ALG}_j^{(j+2)} \setminus \mathrm{ALG}_{j-1}^{(j+2)}) + p_j(R_j^{(j+2)} \setminus R_{j-1}^{(j+2)}) + p_j(R_j^{(j+1)} \setminus R_{j-1}^{(j+1)}) \\
&\vdots \\
&= p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)}) + \sum_{t=j+1}^{t_i} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}).
\end{aligned} \tag{B.14}$$

(B.14) allows us to write (introducing N and J as defined below)

$$\begin{aligned}
(1+\epsilon)cI - cH &= (1+\epsilon)c\sum_{j=t'}^{t_i-1} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)}) - c\sum_{j=1}^{t'-1} p_j(R_j^{(t_i)} \setminus R_{j-1}^{(t_i)}) \\
&= (1+\epsilon)\Big(c\sum_{j=t'}^{t_i-1} \underbrace{p_j(\mathrm{ALG}_j^{(j)} \setminus \mathrm{ALG}_{j-1}^{(j)})}_{a_j} - c\sum_{j=t'}^{t_i-1}\sum_{t=j+1}^{t_i} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})\Big)
\end{aligned}$$

$$
\begin{aligned}
& -c\sum_{j=1}^{t'-1} p_j(R_j^{(t_i)} \setminus R_{j-1}^{(t_i)}) \\
= \quad & (1+\epsilon)c\sum_{j=t'}^{t_i-1} a_j - \epsilon c \underbrace{\sum_{j=t'}^{t_i-1}\sum_{t=j+1}^{t_i} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})}_{N} \\
& -c\Big(\underbrace{\sum_{j=t'}^{t_i-1}\sum_{t=j+1}^{t_i} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}) + \sum_{j=1}^{t'-1} p_j(R_j^{(t_i)} \setminus R_{j-1}^{(t_i)})}_{J}\Big).
\end{aligned}
$$

Observing that

$$
J \geq \sum_{j=t'}^{t_i-1} p_j(R_j^{(t_i)} \setminus R_{j-1}^{(t_i)}) + \sum_{j=1}^{t'-1} p_j(R_j^{(t_i)} \setminus R_{j-1}^{(t_i)}) = \underbrace{\sum_{j=1}^{t_i-1} p_j(R_j^{(t_i)} \setminus R_{j-1}^{(t_i)})}_{b_{t_i}},
$$

(where the inequality comes from holding $t = t_i$ in the internal summation of the first term), we deduce

$$
(1+\epsilon)cI - cH = (1+\epsilon)c\sum_{j=t'}^{t_i-1} a_j - \epsilon cN - cJ \leq (1+\epsilon)c\sum_{j=t'}^{t_i-1} a_j - \epsilon cN - cb_{t_i} \leq (1+\epsilon)c\sum_{j=t'}^{t_i-1} a_j - cb_{t_i}. \tag{B.15}
$$

Plugging (B.15) into (B.13), we have:

$$
c\epsilon G + (1+\epsilon)c\sum_{j=t'}^{t_i-1} a_j - cb_{t_i} + (1+\epsilon)a_{t_i} \geq \ell_{t_i}
$$

and the thesis follows since

$$
G = \sum_{j=1}^{t'-1} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)}) \leq \sum_{j=1}^{t_i} p_j(\mathrm{ALG}_j^{(t_i)} \setminus \mathrm{ALG}_{j-1}^{(t_i)}) = \Phi(\mathrm{ALG}^{(t_i)}).
$$

To settle the statement for t_1, observe that $t_1 \geq 2$ by definition of $\mathcal{B}$ and recall that we set $t_0 = 1$.

Then

$$w(\underbrace{\mathrm{OPT}_{t_1} \setminus \mathrm{OPT}_{t_1-1}}_{B'}) \le w(\mathrm{OPT}_{t_1}) \le W_{t_1},$$

by feasibility of OPT. Thus B' is a feasible solution for algorithm $\Pi^{(t_1)}$. Similarly to the case for t_i with $i \ge 2$, $\Pi^{(t_1)}$ chooses $\mathrm{ALG}^{(t_1)}$, whose profit in $\Pi^{(t_1)}$ is

$$D' = c\sum_{j=1}^{t_1-1} p_j(\mathrm{ALG}_j^{(t_1)} \setminus \mathrm{ALG}_{j-1}^{(t_1)}) + p_{t_1}(\mathrm{ALG}_{t_1}^{(t_1)} \setminus \mathrm{ALG}_{t_1-1}^{(t_1)}),$$

we have

$$\begin{aligned}
(1+\epsilon)D' &\ge p(B') \\
&= c\sum_{j=1}^{t_1-1} p_j((\mathrm{ALG}_j^{(t_1-1)} \setminus \mathrm{ALG}_{j-1}^{(t_1-1)}) \cap (\mathrm{OPT}_{t_1} \setminus \mathrm{OPT}_{t_1-1})) \\
&\quad + \underbrace{p_{t_1}(\mathrm{OPT}_{t_1} \setminus (\mathrm{OPT}_{t_1-1} \cup \mathrm{ALG}_{t_1-1}^{(t_1-1)}))}_{\ell_{t_1}} \\
&\ge \ell_{t_1}. \qquad \text{(B.16)}
\end{aligned}$$

Note that (B.14) holds for all $j \in [t_1 - 1]$, therefore we have:

$$\begin{aligned}
(1+\epsilon)D' &= (1+\epsilon)(c\sum_{j=1}^{t_1-1} \underbrace{p_j(\mathrm{ALG}_j^{(j)} \setminus \mathrm{ALG}_{j-1}^{(j)})}_{a_j} - c\sum_{j=1}^{t_1-1}\sum_{t=j+1}^{t_1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}) \\
&\quad + \underbrace{p_{t_1}(\mathrm{ALG}_{t_1}^{(t_1)} \setminus \mathrm{ALG}_{t_1-1}^{(t_1)})}_{a_{t_1}}) \\
&\le (1+\epsilon)(c\sum_{j=1}^{t_1-1} a_j - c\underbrace{\sum_{j=1}^{t_1-1} p_j(R_j^{(t_1)} \setminus R_{j-1}^{(t_1)})}_{b_{t_1}} + a_{t_1}) \\
&\le (1+\epsilon)(c\sum_{j=1}^{t_1-1} a_j + a_{t_1}) - cb_{t_1}
\end{aligned}$$

$$\leq \quad (1+\epsilon)(c\sum_{j=1}^{t_1-1} a_j + a_{t_1}) + c\epsilon\Phi(\mathrm{ALG}^{(t_1)}) - cb_{t_1},$$

where the first equality uses (B.14) and the first inequality comes from holding $t = t_1$ in the internal summation of the second term. The second and third inequality follow since b_{t_1} and $\Phi(\mathrm{ALG}^{(t_1)})$ are both non-negative. Finally, combining the above inequality with (B.16) gives the result. □

Proof of Lemma 2.4.8. Recall that by Lemma B.4.5 we have for $t_i \in \mathcal{B}$,

$$\ell_{t_i} \leq (1+\epsilon)(c\sum_{j=t_{i-1}}^{t_i-1} a_j + a_{t_i}) + c\epsilon\Phi(\mathrm{ALG}^{(t_i)}) - cb_{t_i},$$

Plugging the above in Lemma B.4.2, we have

$$\begin{aligned}
\Phi(\mathrm{HB}^{(t_i-1)}) - \Phi(\mathrm{HB}^{(t_i)}) &\leq b_{t_i} - a_{t_i} + \ell_{t_i} + d_{t_i} \\
&\leq b_{t_i} - a_{t_i} + (1+\epsilon)(c\sum_{j=t_{i-1}}^{t_i-1} a_j + a_{t_i}) + c\epsilon\Phi(\mathrm{ALG}^{(t_i)}) - cb_{t_i} + d_{t_i} \\
&= (1+\epsilon)c\sum_{j=t_{i-1}}^{t_i-1} a_j + \epsilon a_{t_i} - (c-1)b_{t_i} + c\epsilon\Phi(\mathrm{ALG}^{(t_i)}) + d_{t_i}.
\end{aligned}$$

□

B.4.8 Proof of Lemma 2.4.9

Let $t \in [T]$, $t \geq 2$. From Observation B.4.4 and the definition of δ_t, we have:

$$\underbrace{p_t(\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})}_{a_t} \geq c\underbrace{\sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})}_{b_t},$$

and therefore

$$\begin{aligned}\underbrace{\sum_{j=1}^{t} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)})}_{\Phi(\mathrm{ALG}^{(t)})} &= \underbrace{p_t(\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})}_{a_t} + \sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}) \\ &\geq c \underbrace{\sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})}_{b_t} + \sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}).\end{aligned}$$

Repeatedly applying the above inequality and Lemma B.4.1, we derive:

$$\begin{aligned}\underbrace{\sum_{j=1}^{T} p_j(\mathrm{ALG}_j^{(T)} \setminus \mathrm{ALG}_{j-1}^{(T)})}_{\Phi(\mathrm{ALG})=\Phi(\mathrm{ALG}^{(T)})} &\geq c \sum_{j=1}^{T-1} p_j(R_j^{(T)} \setminus R_{j-1}^{(T)}) + \sum_{j=1}^{T-1} p_j(\mathrm{ALG}_j^{(T)} \setminus \mathrm{ALG}_{j-1}^{(T)}) \\ &= (c-1)\sum_{j=1}^{T-1} p_j(R_j^{(T)} \setminus R_{j-1}^{(T)}) + \underbrace{\sum_{j=1}^{T-1} p_j(\mathrm{ALG}_j^{(T-1)} \setminus \mathrm{ALG}_{j-1}^{(T-1)})}_{\Phi(\mathrm{ALG}^{(T-1)})} \\ &\geq (c-1)\sum_{j=1}^{T-1} p_j(R_j^{(T)} \setminus R_{j-1}^{(T)}) + c\sum_{j=1}^{T-2} p_j(R_j^{(T-1)} \setminus R_{j-1}^{(T-1)}) \\ &\quad + \sum_{j=1}^{T-2} p_j(\mathrm{ALG}_j^{(T-1)} \setminus \mathrm{ALG}_{j-1}^{(T-1)}) \\ &= (c-1)\sum_{j=1}^{T-1} p_j(R_j^{(T)} \setminus R_{j-1}^{(T)}) + (c-1)\sum_{j=1}^{T-2} p_j(R_j^{(T-1)} \setminus R_{j-1}^{(T-1)}) \\ &\quad + \underbrace{\sum_{j=1}^{T-2} p_j(\mathrm{ALG}_j^{(T-2)} \setminus \mathrm{ALG}_{j-1}^{(T-2)})}_{\Phi(\mathrm{ALG}^{(T-2)})} \\ &\quad \vdots \\ &\geq (c-1)\sum_{t=2}^{T}\sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}) + p_1(\mathrm{ALG}_1^{(1)} \setminus \mathrm{ALG}_0^{(1)})\end{aligned}$$

$$\geq (c-1)\sum_{t=2}^{T}\underbrace{\sum_{j=1}^{t-1} p_j(R_j^{(t)}\backslash R_{j-1}^{(t)})}_{b_t}.$$

Thus,

$$\sum_{t=2}^{T} b_t \leq \frac{1}{c-1}\Phi(\mathrm{ALG}).$$

B.4.9 Proof of Claim 2.4.10

By Lemma 2.4.6, Lemma 2.4.7 and Lemma 2.4.8, summing differences $\Phi(\mathrm{HB}_{t-1}) - \Phi(\mathrm{HB}_t)$ over all $t \in [T]$,

$$\begin{aligned}
\sum_{t=1}^{T}(\Phi(\mathrm{HB}^{(t-1)}) &- \Phi(\mathrm{HB}^{(t)})) \\
&\leq \epsilon\sum_{t=1}^{T} a_t - (c-1)\sum_{t=2}^{T} b_t + \sum_{t=1}^{T} d_t + (1+\epsilon)(c\sum_{t=1}^{T} a_t) + c\epsilon\sum_{t=1}^{T}\Phi(\mathrm{ALG}^{(t)}) \\
&= (c+(c+1)\epsilon)\sum_{t=1}^{T} a_t - (c-1)\sum_{t=2}^{T} b_t + \sum_{t=1}^{T} d_t + c\epsilon\sum_{t=1}^{T}\Phi(\mathrm{ALG}^{(t)})
\end{aligned} \tag{B.17}$$

We now bound the right-hand side of (B.17). By Lemma B.4.1, for $t \in [T]$, $t \geq 2$, $\sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t)} \setminus \mathrm{ALG}_{j-1}^{(t)}) = \sum_{j=1}^{t-1} p_j(\mathrm{ALG}_j^{(t-1)} \setminus \mathrm{ALG}_{j-1}^{(t-1)}) - \sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})$. Thus, by repeatedly applying Lemma B.4.1,

$$\begin{aligned}
\underbrace{\sum_{t=1}^{T} p_t(\mathrm{ALG}_t^{(T)} \setminus \mathrm{ALG}_{t-1}^{(T)})}_{\Phi(\mathrm{ALG})} &= p_T(\mathrm{ALG}_T^{(T)} \setminus \mathrm{ALG}_{T-1}^{(T)}) + \sum_{j=1}^{T-1} p_j(\mathrm{ALG}_j^{(T)} \setminus \mathrm{ALG}_{j-1}^{(T)}) \\
&= p_T(\mathrm{ALG}_T^{(T)} \setminus \mathrm{ALG}_{T-1}^{(T)}) + \sum_{j=1}^{T-1} p_j(\mathrm{ALG}_j^{(T-1)} \setminus \mathrm{ALG}_{j-1}^{(T-1)})
\end{aligned}$$

$$
\begin{aligned}
& - \sum_{j=1}^{T-1} p_j(R_j^{(T)} \setminus R_{j-1}^{(T)}) \\
= \;& p_T(\mathrm{ALG}_T^{(T)} \setminus \mathrm{ALG}_{T-1}^{(T)}) + p_{T-1}(\mathrm{ALG}_{T-1}^{(T-1)} \setminus \mathrm{ALG}_{T-2}^{(T-1)}) \\
& + \sum_{j=1}^{T-2} p_j(\mathrm{ALG}_j^{(T-1)} \setminus \mathrm{ALG}_{j-1}^{(T-1)}) - \sum_{j=1}^{T-1} p_j(R_j^{(T)} \setminus R_{j-1}^{(T)}) \\
= \;& \sum_{t=T-1}^{T} p_t(\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}) + \sum_{j=1}^{T-2} p_j(\mathrm{ALG}_j^{(T-2)} \setminus \mathrm{ALG}_{j-1}^{(T-2)}) \\
& - \sum_{t=T-1}^{T} \sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)}) \\
& \vdots \\
= \;& \sum_{t=1}^{T} \underbrace{p_t(\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})}_{a_t} - \sum_{t=2}^{T} \underbrace{\sum_{j=1}^{t-1} p_j(R_j^{(t)} \setminus R_{j-1}^{(t)})}_{b_t}
\end{aligned}
$$

Therefore,

$$
\sum_{t=1}^{T} a_t = \Phi(\mathrm{ALG}) + \sum_{t=2}^{T} b_t. \tag{B.18}
$$

Moreover, for every $t \in [T]$,

$$
\begin{aligned}
d_t &= \sum_{j=t+1}^{T} p_j((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})) \\
&\leq \sum_{j=t+1}^{T} p_t((\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}) \cap (\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})) \\
&= p_t((\cup_{j=t+1}^{T}(\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1})) \cap (\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)})) \\
&\leq p_t(\mathrm{ALG}_t^{(t)} \setminus \mathrm{ALG}_{t-1}^{(t)}) \\
&= a_t,
\end{aligned} \tag{B.19}
$$

where the first inequality follows since the function p_t is non-increasing in t, and the second equality follows since $\mathrm{OPT}_j \setminus \mathrm{OPT}_{j-1}$ are disjoint sets for all $j \in [T]$. Plugging (B.18) and (B.19)

into (B.17), we deduce:

$$\begin{aligned}
\sum_{t=1}^{T}(\Phi(\mathrm{HB}^{(t-1)}) - \Phi(\mathrm{HB}^{(t)})) \quad &\leq \quad (c + (c+1)\epsilon)(\Phi(\mathrm{ALG}) + \sum_{t=2}^{T} b_t) - (c-1)\sum_{t=2}^{T} b_t + \sum_{t=1}^{T} d_t \\
&\quad + c\epsilon \sum_{t=1}^{T} \Phi(\mathrm{ALG}^{(t)}) \\
&\leq \quad (c + (c+1)\epsilon)(\Phi(\mathrm{ALG}) + \sum_{t=2}^{T} b_t) - (c-1)\sum_{t=2}^{T} b_t \\
&\quad + c\epsilon \sum_{t=1}^{T} \Phi(\mathrm{ALG}^{(t)}) + \Phi(\mathrm{ALG}) + \sum_{t=2}^{T} b_t \\
&= \quad (c + 1 + (c+1)\epsilon)\Phi(\mathrm{ALG}) + (2 + (c+1)\epsilon)\sum_{t=2}^{T} b_t \\
&\quad + c\epsilon \sum_{t=1}^{T} \Phi(\mathrm{ALG}^{(t)}).
\end{aligned}$$

B.4.10 Proof of Claim 2.4.11

$$\begin{aligned}
U &= (c + 1 + (c+1)\epsilon)\ \Phi(\mathrm{ALG}) + (2 + (c+1)\epsilon)\sum_{t=2}^{T} b_t + c\epsilon \sum_{t=1}^{T} \Phi(\mathrm{ALG}^{(t)}) \\
&\leq (c + 1 + (c+1)\epsilon)\ \Phi(\mathrm{ALG}) + (2 + (c+1)\epsilon)(\frac{1}{c-1})\ \Phi(\mathrm{ALG}) + c\epsilon T\ \Phi(\mathrm{ALG}) \\
&= (c + 1 + \frac{2}{c-1})\ \Phi(\mathrm{ALG}) + (c + 1 + \frac{c+1}{c-1} + cT)\epsilon\ \Phi(\mathrm{ALG}) \\
&= (c + 1 + \frac{2}{c-1} + \epsilon')\ \Phi(\mathrm{ALG})
\end{aligned}$$

where the first inequality follows from plugging in (2.2) and Lemma 2.4.9, and the final equality follows by the definition of ϵ'.

B.4.11 Proof of Theorem 2.4.2

Consider the following instance of the generalized incremental knapsack problem where $n = 2T - 1$. Here T is taken to be a large enough integer, and $\epsilon > 0$ a small enough real number. There

are 2 sets of items: $i \in [T]$ and $i \in [2T-1] \backslash [T]$. For the first set,

$$p_{i,t} = \begin{cases} c^t & \text{if } t \leq i \\ 0 & \text{otherwise} \end{cases}, \quad w_i = T^i + \epsilon.$$

For the second set,

$$p_{T+i,t} = \begin{cases} c^t + (c^t - 1)\epsilon & \text{if } t \leq i \\ 0 & \text{otherwise} \end{cases}, \quad w_{T+i} = \sum_{t=1}^{i} T^t + T\epsilon.$$

Additionally, let $W_t = \sum_{\tau=1}^{t} T^\tau + T\epsilon$.

A feasible solution is to insert item t (hence, from the first set) in time t. The profit of this solution is:

$$\sum_{i=1}^{T} c^i = \frac{c^{T+1} - c}{c - 1}.$$

In $\Pi^{(1)}$, only items 1 and $T+1$ are feasible, thus the algorithm will insert item $T+1$ since it earns $(c-1)\epsilon$ more profit.

We claim that, at each round $2 \leq t \leq T-1$, the solution constructed by $\Pi^{(t-1)}$ contains exactly item $T+t-1$, and $\Pi^{(t)}$ removes item $T+t-1$ and adds item $T+t$. We prove this claim by induction. For the basic case, first observe that, in $\Pi^{(2)}$, items $T+i$ and i for any $i > 2$ are not feasible. Also observe that inserting item 1 at this time gives a profit of 0. Neither item 2 nor $T+2$ are feasible without removing item $T+1$. So all choices of the algorithm are dominated by the following options:

a) Remove item $T+1$ to insert item 2, leading to a profit of c^2;

b) Removing item $T+1$ to insert item $T+2$, leading to a profit of $c^2 + (c^2 - 1)\epsilon$;

c) Keeping item $T+1$, leading to a profit of

$$c(c + (c-1)\epsilon) = c^2 + (c^2 - c)\epsilon < c^2 + (c^2 - 1)\epsilon.$$

Thus $\Pi^{(2)}$ removes item $T+1$ to insert item $T+2$, as required.

For the inductive step, suppose that the solution constructed by $\Pi^{(t)}$ is given by item $T+t$ only. In $\Pi^{(t+1)}$, the knapsack capacity increases by T^{t+1}. Thus again neither item $t+1$ nor $T+t+1$ is feasible without removing item $T+t$. Moreover, by construction, none of the items j and $T+j$ with $1 \leq j \leq t$ gives any profit, while none of the items j and $T+j$ with $t+2 \leq j \leq T$ can be inserted because of each of them alone violates the capacity constraint. Hence, all choices of the algorithm are dominated by the following options:

a) Remove item $T+t$ to insert item $t+1$, leading to a profit of c^{t+1};

b) Removing item $T+t$ to insert item $T+t+1$, leading to a profit of $c^{t+1}+(c^{t+1}-1)\epsilon$;

c) Keeping item $T+t$, leading to a profit of

$$c(c^t+(c^t-1)\epsilon) = c^{t+1}+(c^{t+1}-c)\epsilon < c^{t+1}+(c^{t+1}-1)\epsilon.$$

Thus $\Pi^{(t+1)}$ removes item $T+t$ to insert item $T+t+1$ and the claim follows.

It is easy to check that $\Pi^{(T)}$ will keep item $T-1$ and not add any item; hence, ALG earns profit $c^{T-1}+(c^{T-1}-1)\epsilon$.

We finally compare the profit of the optimal chain OPT against the profit of the chain produced by the fully flexible algorithm ALG:

$$\frac{\Phi(\mathrm{ALG})}{\Phi(\mathrm{OPT})} \leq \frac{(c-1)(c^{T-1}+(c^{T-1}-1)\epsilon)}{c^{T+1}-c} \leq (1+\epsilon)\frac{c-1}{c^2-\frac{1}{c^{T-2}}} \xrightarrow[T\to\infty]{} (1+\epsilon)\frac{c-1}{c^2},$$

and the thesis follows by taking ϵ arbitrarily small.

Appendix C: Algorithms for the generalized incremental knapsack problem through a sequencing reformulation

C.1 Additional proofs from Section 3.2

C.1.1 Proof of Claim 3.2.5

We first show that $(\tilde{S}^+, \tilde{\pi}^+)$ is indeed a bulky pair. For this purpose, since $(\tilde{S}, \tilde{\pi})$ is bulky, it suffices to explain why each item $i \in Q$ is necessarily k_i-heavy, where k_i is the unique index for which $C_{\tilde{\pi}^+}(i) \in \mathcal{I}_{k_i}$. This claim follows by noting that, for such items, the way we construct $(\tilde{S}^+, \tilde{\pi}^+)$ leads to a completion time of

$$\begin{aligned} C_{\tilde{\pi}^+}(i) &= w(\tilde{S}) + \sum_{j \in Q : \pi(j) \leq \pi(i)} w_j \\ &< w(\hat{S}) + \sum_{j \in Q : \pi(j) \leq \pi(i)} w_j \\ &= C_\pi(i) \ . \end{aligned} \tag{C.1}$$

Recalling that $Q = \{i \in S : C_\pi(i) \in \mathcal{I}_k\}$, we have just shown that $k_i \leq k$, and since item i is k-heavy due to the bulkiness of (S, π), it is k_i-heavy as well.

We proceed by showing that $(\tilde{S}^+, \tilde{\pi}^+)$ satisfies conditions 1-3:

1. *Top index*: $\text{top}(\tilde{S}^+, \tilde{\pi}^+) \leq k$. To verify this property, note that when $Q = \emptyset$, we clearly have $w(\tilde{S}^+) = w(\tilde{S}) < w(\hat{S}) = w(S)$, and therefore, $\text{top}(\tilde{S}^+, \tilde{\pi}^+) \leq \text{top}(S, \pi) \leq k$. In the opposite case, where $Q \neq \emptyset$, the makespans of both $\tilde{S}^+$ and S are attained by the respective completion times of precisely the same item in Q. However, by inequality (C.1), we have $C_{\tilde{\pi}^+}(i) \leq C_\pi(i)$ for every $i \in Q$, and it follows that $\text{top}(\tilde{S}^+, \tilde{\pi}^+) \leq \text{top}(S, \pi) \leq k$.

2. *Total profit*: $\Psi(\tilde{\pi}^+) \geq \psi_k$. Along the same lines, since $C_{\tilde{\pi}^+}(i) \leq C_\pi(i)$ for every $i \in Q$, it

follows that $\varphi_{\tilde{\pi}^+}(i) \geq \varphi_\pi(i)$ for such items. Thus,

$$\begin{aligned}
\Psi\left(\tilde{\pi}^+\right) &= \sum_{i\in\tilde{S}} \varphi_{\tilde{\pi}^+}(i) + \sum_{i\in Q} \varphi_{\tilde{\pi}^+}(i) \\
&= \sum_{i\in\tilde{S}} \varphi_{\tilde{\pi}}(i) + \sum_{i\in Q} \varphi_{\tilde{\pi}^+}(i) \\
&\geq \psi_{k-1} + \sum_{i\in Q} \varphi_\pi(i) \\
&= \left[\psi_k - \sum_{i\in Q} \varphi_\pi(i)\right]^+ + \sum_{i\in Q} \varphi_\pi(i) \\
&\geq \psi_k \ .
\end{aligned}$$

Here, the second equality holds since the permutations $\tilde{\pi}^+$ and $\tilde{\pi}$ are identical when restricted to items in $\tilde{S}$. The first inequality follows by recalling that $(\tilde{S}, \tilde{\pi}) \in \text{Bulky}(k-1, \psi_{k-1}, Q_{k-1})$, meaning in particular that $\sum_{i\in\tilde{S}} \varphi_{\tilde{\pi}}(i) = \Psi(\tilde{\pi}) \geq \psi_{k-1}$.

3. *Core*: $\text{core}(\tilde{S}^+) = Q_k$. One can easily verify that, for any pair of disjoint sets of items, S_1 and S_2, we have $\text{core}(S_1 \cup S_2) = \text{core}(\text{core}(S_1) \cup \text{core}(S_2))$. Therefore,

$$\begin{aligned}
\text{core}(\tilde{S}^+) &= \text{core}(\tilde{S} \cup Q) \\
&= \text{core}(\text{core}(\tilde{S}) \cup \text{core}(Q)) \\
&= \text{core}(\text{core}(S \setminus Q) \cup \text{core}(Q)) \\
&= \text{core}(S) \\
&= Q_k \ ,
\end{aligned}$$

where the second equality follows by noting that $\tilde{S}$ and Q are disjoint, and similarly, the fourth equality holds since $S \setminus Q$ and Q are clearly disjoint.

C.1.2 Proof of Lemma 3.2.6

Let us consider the sequence of states traversed by the dynamic program F, as it arrives to the optimal state (K, ψ_K^*, Q_K^*); the latter is "optimal" in the sense that $\psi_K^* = \psi^*$ and $F(K, \psi_K^*, Q_K^*) < \infty$. This sequence, along with the specific parameters and the bulky pair corresponding to each state will be designated by:

$$\begin{array}{c}(0,\psi_0^*,Q_0^*)\\(S_0^*,\pi_{S_0^*})\end{array} \xrightarrow[Q_1^*,\pi_{Q_1^*}]{} \begin{array}{c}(1,\psi_1^*,Q_1^*)\\(S_1^*,\pi_{S_1^*})\end{array} \xrightarrow[Q_2^*,\pi_{Q_2^*}]{} \begin{array}{c}(2,\psi_2^*,Q_2^*)\\(S_2^*,\pi_{S_2^*})\end{array} \xrightarrow[\cdots\cdots]{} \cdots \xrightarrow[Q_k^*,\pi_{Q_k^*}]{} \begin{array}{c}(k,\psi_k^*,Q_k^*)\\(S_k^*,\pi_{S_k^*})\end{array}$$
$$\xrightarrow[\cdots\cdots]{} \cdots \xrightarrow[Q_K^*,\pi_{Q_K^*}]{} \begin{array}{c}(K,\psi_K^*,Q_K^*)\\(S_K^*,\pi_{S_K^*})\end{array}.$$

To better understand this illustration, we note that for every $k \in [K]$, the collection of items Q_k^* and their internal permutation $\pi_{Q_k^*}$ are precisely those by which the dynamic program F transitions from state $(k-1, \psi_{k-1}^*, Q_{k-1}^*)$ to state (k, ψ_k^*, Q_k^*). Consequently, the resulting item set is $S_k^* = S_{k-1}^* \uplus Q_k^*$, whereas the resulting permutation $\pi_{S_k^*}$ is obtained by appending $\pi_{Q_k^*}$ to $\pi_{S_{k-1}^*}$. In addition, for the starting state, we have $\psi_0^* = 0$ and $Q_0^* = \emptyset$.

To prove the desired claim, we argue that one feasible sequence of states that can be traversed by the approximate program $\tilde{F}$ is obtained when each profit parameter ψ_k^* is substituted by $\tilde{\psi}_k = \lceil \psi_k^* - \min\{k, |S_k^*|\} \cdot \frac{\epsilon p_{\max}}{n} \rceil_{\mathcal{D}_\psi}$. Here, the operator $\lceil \cdot \rceil_{\mathcal{D}_\psi}$ rounds its argument up to the nearest value in $\mathcal{D}_\psi$. In other words, as shown in Claim C.1.1 below, we prove that

$$\begin{array}{c}(0,\tilde{\psi}_0,Q_0^*)\\(S_0^*,\pi_{S_0^*})\end{array} \xrightarrow[Q_1^*,\pi_{Q_1^*}]{} \begin{array}{c}(1,\tilde{\psi}_1,Q_1^*)\\(S_1^*,\pi_{S_1^*})\end{array} \xrightarrow[Q_2^*,\pi_{Q_2^*}]{} \begin{array}{c}(2,\tilde{\psi}_2,Q_2^*)\\(S_2^*,\pi_{S_2^*})\end{array} \xrightarrow[\cdots\cdots]{} \cdots \xrightarrow[Q_k^*,\pi_{Q_k^*}]{} \begin{array}{c}(k,\tilde{\psi}_k,Q_k^*)\\(S_k^*,\pi_{S_k^*})\end{array}$$
$$\xrightarrow[\cdots\cdots]{} \cdots \xrightarrow[Q_K^*,\pi_{Q_K^*}]{} \begin{array}{c}(K,\tilde{\psi}_K,Q_K^*)\\(S_K^*,\pi_{S_K^*})\end{array}$$

forms a feasible sequence of states, action parameters, and bulky pairs for $\tilde{F}$. That is, we have $(S_k^*, \pi_{S_k^*}) \in \widetilde{\text{Bulky}}(k, \tilde{\psi}_k, Q_k^*)$, for every $k \in [K]_0$. In light of this result, we conclude in particular

that $\tilde{F}(K, \tilde{\psi}_K, Q_K^*) < \infty$ with

$$\begin{aligned}\tilde{\psi}_K &= \left\lceil \psi_K^* - \min\{K, |S_K^*|\} \cdot \frac{\epsilon p_{\max}}{n} \right\rceil_{\mathcal{D}_\psi} \\ &\geq \psi^* - \epsilon p_{\max} \\ &\geq (1-\epsilon) \cdot \psi^* \ .\end{aligned}$$

Here, the first inequality holds since $\psi_K^* = \psi^*$ and $|S_K^*| \leq n$. To understand the second inequality, note that for every item $i \in [n]$, the pair that consists of introducing this item and nothing more is necessarily bulky. Indeed, as a result, the completion time of item i would fall within the interval $\mathcal{I}_{k_i}$, where k_i is the unique integer for which $(1+\epsilon)^{k_i - 1} < w_i \leq (1+\epsilon)^{k_i}$. However, since $w_i > (1+\epsilon)^{k_i - 1} \geq \epsilon^2 \cdot (1+\epsilon)^{k_i}$ for $\epsilon \leq \frac{1}{2}$, it follows that item i is k-heavy, implying in turn that the pair in question is bulky. Now, noting that this pair guarantees a profit of $\max\{p_{it} : t \in [T] \text{ and } w_i \leq W_t\}$, any such expression provides a lower bound on ψ^*, meaning that $\psi^* \geq \max\{p_{it} : i \in [n], t \in [T], \text{ and } w_i \leq W_t\} = p_{\max}$.

Claim C.1.1. $(S_k^*, \pi_{S_k^*}) \in \widetilde{\text{Bulky}}(k, \tilde{\psi}_k, Q_k^*)$*, for every* $k \in [K]_0$.

Proof. We first note that the parameter $\tilde{\psi}_k$ is indeed well-defined for all $k \in [K]_0$, since $\tilde{\psi}_k \leq \lceil \psi_K^* \rceil_{\mathcal{D}_\psi} \leq n p_{\max} = \max \mathcal{D}_\psi$. Given this observation, we proceed to prove the claim by induction on k.

In the base case of $k = 0$, the claim trivially holds since $\tilde{\psi}_0 = 0$, $Q_0^* = \emptyset$, $S_0^* = \emptyset$, and $\pi_{S_0^*}$ is the empty permutation. In the general case of $k \geq 1$, to argue that $(S_k^*, \pi_{S_k^*}) \in \tilde{B}(k, \tilde{\psi}_k, Q_k^*)$, we consider two scenarios, depending on whether Q_k^* is empty or not:

- *Case 1:* $Q_k^* = \emptyset$. We first observe that, since $S_k^* = S_{k-1}^* \cup Q_k^*$, we have $S_k^* = S_{k-1}^*$ by the case hypothesis, implying in turn that $\psi_k^* = \psi_{k-1}^*$ and $Q_k^* = Q_{k-1}^*$. Consequently,

$$\begin{aligned}\tilde{\psi}_k &= \left\lceil \psi_k^* - \min\{k, |S_k^*|\} \cdot \frac{\epsilon p_{\max}}{n} \right\rceil_{\mathcal{D}_\psi} \\ &\leq \left\lceil \psi_{k-1}^* - \min\{k-1, |S_{k-1}^*|\} \cdot \frac{\epsilon p_{\max}}{n} \right\rceil_{\mathcal{D}_\psi}\end{aligned}$$

$$= \tilde{\psi}_{k-1} \ .$$

and it follows that $\widetilde{\text{Bulky}}(k, \tilde{\psi}_k, Q_k^*) \supseteq \widetilde{\text{Bulky}}(k, \tilde{\psi}_{k-1}, Q_{k-1}^*) \supseteq \widetilde{\text{Bulky}}(k-1, \tilde{\psi}_{k-1}, Q_{k-1}^*)$, where the first inclusion holds since $\tilde{\psi}_k \leq \tilde{\psi}_{k-1}$ and $Q_k^* = Q_{k-1}^*$. Thus, $(S_k^*, \pi_{S_k^*}) = (S_{k-1}^*, \pi_{S_{k-1}^*}) \in \widetilde{\text{Bulky}}(k-1, \tilde{\psi}_{k-1}, Q_{k-1}^*) \subseteq \widetilde{\text{Bulky}}(k, \tilde{\psi}_k, Q_k^*)$, where the middle transition is precisely our induction hypothesis.

- *Case 2:* $Q_k^* \neq \emptyset$. In this case, $|S_k^*| = |S_{k-1}^*| + |Q_k^*| \geq |S_{k-1}^*| + 1$, as S_k^* is the disjoint union of S_{k-1}^* and Q_k^*. By the inductive hypothesis, $(S_{k-1}^*, \pi_{S_{k-1}^*}) \in \widetilde{\text{Bulky}}(k-1, \tilde{\psi}_{k-1}, Q_{k-1}^*)$, meaning that for the purpose of proving $(S_k^*, \pi_{S_k^*}) \in \widetilde{\text{Bulky}}(k, \tilde{\psi}_k, Q_k^*)$, it suffices to show that $\tilde{\psi}_{k-1} + \sum_{i \in Q_k^*} \varphi_{\pi_{S_k^*}}(i) \geq \tilde{\psi}_k$. We establish the latter inequality by noting that

$$\begin{aligned}
\tilde{\psi}_{k-1} + \sum_{i \in Q_k^*} \varphi_{\pi_{S_k^*}}(i) &= \tilde{\psi}_{k-1} + \psi_k^* - \psi_{k-1}^* \\
&\geq \left(\psi_{k-1}^* - \min\{k-1, |S_{k-1}^*|\} \cdot \frac{\epsilon p_{\max}}{n}\right) + \psi_k^* - \psi_{k-1}^* \\
&\geq \left\lceil \psi_k^* - \min\{k, |S_k^*|\} \cdot \frac{\epsilon p_{\max}}{n} \right\rceil_{\mathcal{D}_\psi} \\
&= \tilde{\psi}_k \ ,
\end{aligned}$$

where the first equality holds since $\psi_k^* = \psi_{k-1}^* + \sum_{i \in Q_k^*} \varphi_{\pi_{S_k^*}}(i)$, by the optimality of ψ_k^*.

□

C.1.3 Proof of Lemma 3.2.8

In order to construct the required permutation, for every $k \in [K-1]$, let π_k be an arbitrary permutation of the items that were assigned by x to bucket $\mathcal{B}_k$, i.e., $\{i \in [n] : x_{ik} = 1\}$. In addition, let π_- be an arbitrary permutation of the remaining items, i.e., those that were not to assigned to any bucket. The permutation π_x is now defined by concatenating these permutations in order of increasing index, with π_- appended at the end, namely, $\pi_x = \langle \pi_1, \ldots, \pi_{K-1}, \pi_- \rangle$. It is easy to verify that this construction can be implemented in $O(nK)$ time.

To obtain a lower bound of $\sum_{i\in[n]} \sum_{k\in[K-1]:i\in L_{k+1}} q_{ik}x_{ik}$ on the profit of this permutation, $\Psi(\pi_x) = \sum_{i\in[n]} \varphi_{\pi_x}(i)$, note that since each item is assigned to at most one bucket, it suffices to show that for every $i \in [n]$ and $k \in [K-1]$ with $x_{ik} = 1$, we necessarily have $\varphi_{\pi_x}(i) \geq q_{ik}$. For this purpose, we observe that

$$\begin{aligned}
\varphi_{\pi_x}(i) &= \max\left\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq C_{\pi_x}(i)\right\} \\
&\geq \max\left\{p_{i,t} : t \in [T+1] \text{ and } W_t \geq (1+\epsilon)^k\right\} \\
&= q_{ik} \,,
\end{aligned}$$

where the inequality above holds since $C_{\pi_x}(i) \leq (1+\epsilon)^k$. Indeed, this bound on the completion time of item i can be derived by observing that every item j that appears before i in the permutation π_x (i.e., $\pi_x(j) < \pi_x(i)$) was assigned by the solution x to one of the buckets $\mathcal{B}_1, \ldots, \mathcal{B}_k$, and therefore,

$$\begin{aligned}
C_{\pi_x}(i) &= \sum_{j\in[n]:\pi_x(j)\leq\pi_x(i)} w_j \\
&\leq \sum_{\kappa\in[k]} \sum_{j\in L_{\kappa+1}} w_j x_{j\kappa} \\
&\leq \sum_{\kappa\in[k]} \text{capacity}(\mathcal{B}_\kappa) \\
&= \sum_{\kappa\in[k]} \left((1+\epsilon)^\kappa - (1+\epsilon)^{\kappa-1}\right) \\
&\leq (1+\epsilon)^k \,,
\end{aligned}$$

where the second inequality follows from the second constraint of (GAP-IP).

C.2 Additional proofs from Section 3.3

C.2.1 Proof of Lemma 3.3.3

Clearly, $\mathcal{R} \cup \mathcal{G}$ is a chain for $\mathcal{I}$, as each of $\mathcal{R}$ and $\mathcal{G}$ is such a chain by itself. To verify the feasibility of $\mathcal{R} \cup \mathcal{G}$, note that for any time period $t \in [T]$, since $\mathcal{R}$ is feasible for $\mathcal{I}^{-\mathcal{G}}$ we have

$$\begin{aligned} w(R_t) &\leq W_t^{-\mathcal{G}} \\ &= \min_{t \leq \tau \leq T} (W_\tau - w(G_\tau)) \\ &\leq W_t - w(G_t) \ . \end{aligned}$$

By recalling that $G_1 \subseteq \cdots \subseteq G_T$ and $R_1 \subseteq \cdots \subseteq R_T \subseteq \mathcal{N}^{-\mathcal{G}} = \mathcal{N} \setminus G_T$, it follows in particular that G_t and R_t are disjoint, implying in turn that $w(R_t \cup G_t) = w(R_t) + w(G_t) \leq W_t$ as required.

Now, to account for the profit of $\mathcal{R} \cup \mathcal{G}$, we conclude that

$$\begin{aligned} \Phi(\mathcal{R} \cup \mathcal{G}) &= \sum_{t \in [T]} \sum_{i \in (R_t \cup G_t) \setminus (R_{t-1} \cup G_{t-1})} p_{it} \\ &= \sum_{t \in [T]} \left(\sum_{i \in R_t \setminus R_{t-1}} p_{it} + \sum_{i \in G_t \setminus G_{t-1}} p_{it} \right) \\ &= \Phi(\mathcal{R}) + \Phi(\mathcal{G}) \ . \end{aligned}$$

Here, the second equality holds again due to the observation above, since having both $G_1 \subseteq \cdots \subseteq G_T$ and $R_1 \subseteq \cdots \subseteq R_T \subseteq \mathcal{N} \setminus G_T$ means that $(R_t \cup G_t) \setminus (R_{t-1} \cup G_{t-1})$ can be written as the disjoint union of $R_t \setminus R_{t-1}$ and $G_t \setminus G_{t-1}$.

C.2.2 Proof of Lemma 3.3.4

For convenience, let us denote the chain in question by $\mathcal{R} = \mathcal{S}|_{\mathcal{N} \setminus G}$. By observing that $R_T = (S_T \cap (\mathcal{N} \setminus G)) = (S_T \setminus G_T) \subseteq \mathcal{N} \setminus G_T$, it follows that $\mathcal{R}$ is also a chain for $\mathcal{I}^{-\mathcal{G}}$. We proceed by

arguing that $\mathcal{R}$ is in fact feasible for the latter instance. To this end, note that for every $t \leq \tau$,

$$\begin{aligned} w(R_t) &\leq w(R_\tau) \\ &= w(S_\tau) - w(G_\tau) \\ &\leq W_\tau - w(G_\tau) \ , \end{aligned}$$

where the middle equality follows by recalling that S_t is the disjoint union of G_t and R_t, and the last inequality is implied by the feasibility of $\mathcal{S}$ for $\mathcal{I}$. As a result, $w(R_t) \leq \min_{t \leq \tau \leq T}(W_\tau - w(G_\tau)) = W_t^{-\mathcal{G}}$, which proves that $\mathcal{R}$ is a feasible chain for $\mathcal{I}^{-\mathcal{G}}$.

We now turn our attention to showing that $\Phi(\mathcal{R}) = \Phi(\mathcal{S}) - \Phi(\mathcal{G})$. Again, based on the observation that S_t is the disjoint union of G_t and R_t for every $t \in [T]$, we conclude that

$$\begin{aligned} \Phi(\mathcal{R}) + \Phi(\mathcal{G}) &= \sum_{t \in [T]} \left(\sum_{i \in R_t \setminus R_{t-1}} p_{it} + \sum_{i \in G_t \setminus G_{t-1}} p_{it} \right) \\ &= \sum_{t \in [T]} \sum_{i \in (R_t \cup G_t) \setminus (R_{t-1} \cup G_{t-1})} p_{it} \\ &= \sum_{t \in [T]} \sum_{i \in S_t \setminus S_{t-1}} p_{it} \\ &= \Phi(\mathcal{S}) \ . \end{aligned}$$

Finally, suppose that $\mathcal{S}$ is optimal for $\mathcal{I}$, but on the other hand, $\mathcal{R}$ is not optimal for $\mathcal{I}^{-\mathcal{G}}$, meaning that there exists a feasible chain $\mathcal{R}'$ for $\mathcal{I}^{-\mathcal{G}}$ with profit $\Phi(\mathcal{R}') > \Phi(\mathcal{R})$. Then, by Lemma 3.3.3, we infer that $\mathcal{R}' \cup \mathcal{G}$ is a feasible chain for $\mathcal{I}$, with profit $\Phi(\mathcal{R}' \cup \mathcal{G}) = \Phi(\mathcal{G}) + \Phi(\mathcal{R}') > \Phi(\mathcal{G}) + \Phi(\mathcal{R}) = \Phi(\mathcal{S})$, contradicting the optimality of $\mathcal{S}$.

C.2.3 Proof of Lemma 3.3.6

We say that an interval $\mathcal{I}_k$ is non-empty with respect to the permutation $\pi_{\mathcal{S}^*}$ if it contains the completion time of at least one item. Note that, since the latter completion time is within $[w_{\min}, nw_{\max}]$ and we assume that $w_{\min} = 3$ (see Section 3.2.2), the interval $\mathcal{I}_0 = [0, 1]$ is

clearly empty. Furthermore, any non-empty interval $\mathcal{I}_k = ((1+\epsilon)^{k-1}, (1+\epsilon)^k]$ necessarily has $\lfloor \log_{1+\epsilon}(w_{\min}) \rfloor \leq k \leq \lceil \log_{1+\epsilon}(nw_{\max}) \rceil$. Therefore, the number of non-empty intervals with respect to $\pi_{\mathcal{S}^*}$ is at most $\lceil \log_{1+\epsilon}(nw_{\max}) \rceil - \lfloor \log_{1+\epsilon}(w_{\min}) \rfloor + 1 \leq 2 \cdot \lceil \log_{1+\epsilon}(n\rho) \rceil$. Now, any such interval $\mathcal{I}_k$ is of length $(1+\epsilon)^k - (1+\epsilon)^{k-1}$, meaning that the number of k-heavy items with a completion time in this interval is at most $\frac{(1+\epsilon)^k - (1+\epsilon)^{k-1}}{\epsilon^2 \cdot (1+\epsilon)^k} \leq \frac{1}{\epsilon}$, as every k-heavy item has a weight of at least $\epsilon^2 \cdot (1+\epsilon)^k$. All in all, we have just shown that $|G^{*\text{heavy}}| \leq \frac{2 \cdot \lceil \log_{1+\epsilon}(n\rho) \rceil}{\epsilon} \leq \frac{3 \log(n\rho)}{\epsilon^2}$.

C.2.4 Proof of Lemma 3.3.7

For every item $i \in \mathcal{N}$, let t_i be its insertion time with respect to the optimal chain $\mathcal{S}^*$. By convention, for non-inserted items (i.e., those in $\mathcal{N} \setminus S^*_T$), we say that their "insertion time" is $T+1$, with a profit of $p_{i,T+1} = 0$. As explained during the proof of Lemma 3.2.1, our construction of the permutation $\pi_{\mathcal{S}^*}$ guarantees that $\varphi_{\pi_{\mathcal{S}^*}}(i) \geq p_{i,t_i}$ for every item $i \in \mathcal{N}$. While this inequality was established for any chain-to-permutation mapping, one can easily notice that, due to the optimality of $\mathcal{S}^*$, we actually have $\varphi_{\pi_{\mathcal{S}^*}}(i) = p_{i,t_i}$ for every $i \in \mathcal{N}$. Otherwise, there would have been at least one item with $\varphi_{\pi_{\mathcal{S}^*}}(i) > p_{i,t_i}$, implying that $\Psi(\pi_{\mathcal{S}^*}) > \Phi(\mathcal{S}^*)$. By Lemma 3.2.1, the permutation $\pi_{\mathcal{S}^*}$ can then be mapped to a feasible chain $\mathcal{S}$ with $\Phi(\mathcal{S}) = \Psi(\pi_{\mathcal{S}^*}) > \Phi(\mathcal{S}^*)$, contradicting the optimality of $\mathcal{S}^*$. Thus, $\Phi(\mathcal{H}^*) = \sum_{i \in G^{*\text{heavy}}} p_{i,t_i} = \sum_{i \in G^{*\text{heavy}}} \varphi_{\pi_{\mathcal{S}^*}}(i) = \Psi_{\text{heavy}}(\pi_{\mathcal{S}^*})$.

C.2.5 Proof of Lemma 3.3.10

We prove the lower bound $\alpha_r \geq \frac{r}{r+1} - r\delta$ by induction on r. For $r = 0$, we have $\alpha_0 = 0$ and the claim clearly holds. Now, for $r \geq 1$,

$$\begin{aligned}
\alpha_r &= \frac{1-\delta}{2-\alpha_{r-1}} \\
&\geq \frac{1-\delta}{2-(\frac{r-1}{r} - (r-1)\delta)} \\
&= \frac{r(1-\delta)}{r+1+r(r-1)\delta} \\
&\geq \frac{r(1-\delta)}{(r+1)(1+(r-1)\delta)}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{r}{r+1} \cdot \left(1 - \frac{r\delta}{1+(r-1)\delta}\right) \\
&\geq \frac{r}{r+1} - r\delta \ .
\end{aligned}
$$

C.3 Additional proofs from Section 3.4

C.3.1 Proof of Lemma 3.4.4

Sparse $(\mathcal{M}^-, \mathcal{M}^+)$-crossing. On the one hand, our construction guarantees that the last item in $C_{\mathcal{M}^-}$ appears in position $\pi(i_{\mathcal{M}^-,\mathcal{M}^+}) - 1 + |\mathcal{A}^-|$ of the permutation $\bar{\pi}$. On the other hand, every item in $C_{\mathcal{M}^+}$ that appears before this position necessarily belongs to $\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi)$. It follows that there are at most $|\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi)| = \frac{1}{\epsilon}$ such items, and therefore, $\text{cross}_{\mathcal{M}^-,\mathcal{M}^+}(\bar{\pi}) \leq \frac{1}{\epsilon}$.

Completion times. We establish this property by considering three cases, depending on whether the item in question appears before $i_{\mathcal{M}^-,\mathcal{M}^+}$, belongs to $\mathcal{A}^-$, or belongs to $\bar{\mathcal{A}}^-$.

- *Before $i_{\mathcal{M}^-,\mathcal{M}^+}$:* For every item $i \in \mathcal{N}$ with $\pi(i) \leq \pi(i_{\mathcal{M}^-,\mathcal{M}^+}) - 1$ we clearly have $C_{\bar{\pi}}(i) = C_{\pi}(i)$, since the permutations $\bar{\pi}$ and π are identical up to position $\pi(i_{\mathcal{M}^-,\mathcal{M}^+}) - 1$.

- *Items in $\mathcal{A}^-$:* For every item $i \in \mathcal{A}^-$, we have $C_{\bar{\pi}}(i) \leq C_{\pi}(i)$, since the collection of items appearing before i in $\bar{\pi}$ is a subset of those appearing before i in π.

- *Items in $\bar{\mathcal{A}}^-$:* For every item $i \in \bar{\mathcal{A}}^-$, the important observation is that the collection of items appearing before i in $\bar{\pi}$ consists of: (1) The same items appearing before i in π, except for the eliminated item $i_{\mathcal{M}^-,\mathcal{M}^+}$; as well as (2) All items in $\mathcal{A}^-$ appearing after i in π. Therefore,

$$
C_{\bar{\pi}}(i) \leq C_{\pi}(i) - w_{i_{\mathcal{M}^-,\mathcal{M}^+}} + w(\mathcal{A}^-) \leq C_{\pi}(i) \ .
$$

  To understand the last inequality, recall that $i_{\mathcal{M}^-,\mathcal{M}^+} \in \mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi)$, meaning in particular that this item resides within $C_{\mathcal{M}^+}$. Since $\mathcal{I} = (\mathcal{N}, W)$ is well-spaced, property 2 of such instances implies that $w_{i_{\mathcal{M}^-,\mathcal{M}^+}}$ is greater than the weight of any item in $C_{\mathcal{M}^-}$ by a multiplicative factor

of at least $n^{1+(\min \mathcal{M}^+ - \max \mathcal{M}^- - 1)/\epsilon} \geq n$, as $\max \mathcal{M}^- < \min \mathcal{M}^+$. Consequently, since all items in $\mathcal{A}^-$ reside within $C_{\mathcal{M}^-}$, we indeed have $w_{i_{\mathcal{M}^-,\mathcal{M}^+}} \geq n \cdot \max_{j \in C_{\mathcal{M}^-}} w_j \geq w(\mathcal{A}^-)$.

Difference. This property is straightforward, by construction of $\bar{\pi}$.

C.3.2 Proof of Claim 3.4.7

For simplicity of notation, let $\mathcal{D} = \{i_{\mathcal{M}^-,\mathcal{M}^+} : (\mathcal{M}^-, \mathcal{M}^+) \in \Omega, \mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+} \neq \emptyset\}$ be the collection of items that were removed throughout all recursive calls to our fixing procedure. Then, the profit of the resulting permutation π_{sparse} can be lower-bounded by observing that

$$\begin{aligned}
\Psi(\pi_{\text{sparse}}) &= \sum_{i \in \mathcal{N} \setminus \mathcal{D}} \varphi_{\pi_{\text{sparse}}}(i) \\
&\geq \sum_{i \in \mathcal{N} \setminus \mathcal{D}} \varphi_{\pi^*}(i) \\
&= \Psi(\pi^*) - \sum_{i \in \mathcal{D}} \varphi_{\pi^*}(i) \\
&\geq \Psi(\pi^*) - \epsilon \cdot \sum_{(\mathcal{M}^-,\mathcal{M}^+) \in \Omega} \varphi_{\pi^*}\left(\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}\left(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}\right)\right).
\end{aligned}$$

Here, the first inequality holds since, for any remaining item $i \in \mathcal{N} \setminus \mathcal{D}$, it is not difficult to verify (by induction on the recursion level) that property (P_2) of the fixing procedure implies $C_{\pi_{\text{sparse}}}(i) \leq C_{\pi^*}(i)$, and we therefore have $\varphi_{\pi_{\text{sparse}}}(i) \geq \varphi_{\pi^*}(i)$. The second inequality is obtained by recalling that any item $i_{\mathcal{M}^-,\mathcal{M}^+} \in \mathcal{D}$ was chosen as the least profitable item in $\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})$ with respect to π^*, thus

$$\begin{aligned}
\varphi_{\pi^*}(i_{\mathcal{M}^-,\mathcal{M}^+}) &\leq \frac{\varphi_{\pi^*}(\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}))}{|\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]})|} \\
&= \epsilon \cdot \varphi_{\pi^*}\left(\mathcal{X}_{\mathcal{M}^-,\mathcal{M}^+}\left(\pi_{[\min \mathcal{M}^-, \max \mathcal{M}^+]}\right)\right).
\end{aligned}$$

C.3.3 Proof of Claim 3.4.8

By definition, $\mathcal{X}_{\mathcal{M}_1^-,\mathcal{M}_1^+}(\pi_{[\min \mathcal{M}_1^-,\max \mathcal{M}_1^+]})$ and $\mathcal{X}_{\mathcal{M}_2^-,\mathcal{M}_2^+}(\pi_{[\min \mathcal{M}_2^-,\max \mathcal{M}_2^+]})$ contain only items in $\mathcal{M}_1^+$-indexed clusters and $\mathcal{M}_2^+$-indexed clusters, respectively. Thus, when $\mathcal{M}_1^+$ and $\mathcal{M}_2^+$ are disjoint, $\mathcal{X}_{\mathcal{M}_1^-,\mathcal{M}_1^+}(\pi_{[\min \mathcal{M}_1^-,\max \mathcal{M}_1^+]})$ and $\mathcal{X}_{\mathcal{M}_2^-,\mathcal{M}_2^+}(\pi_{[\min \mathcal{M}_2^-,\max \mathcal{M}_2^+]})$ must be disjoint as well. Hence, it remains to consider the scenario where $\mathcal{M}_1^+$ and $\mathcal{M}_2^+$ are not disjoint. In this case, the permutations $\pi_{[\min \mathcal{M}_1^-,\max \mathcal{M}_1^+]}$ and $\pi_{[\min \mathcal{M}_2^-,\max \mathcal{M}_2^+]}$ must have been created at different levels of the recursive construction; we assume without loss of generality that $\pi_{[\min \mathcal{M}_1^-,\max \mathcal{M}_1^+]}$ was created at a lower-index level. Therefore, $\mathcal{M}_2^+ \subseteq \mathcal{M}_1^+$, and $\mathcal{X}_{\mathcal{M}_2^-,\mathcal{M}_2^+}(\pi_{[\min \mathcal{M}_2^-,\max \mathcal{M}_2^+]})$ consists of only items in the right permutation, $\pi_{[\min \mathcal{M}_1^+,\max \mathcal{M}_1^+]}$. On the other hand, by construction, any item in $\mathcal{X}_{\mathcal{M}_1^-,\mathcal{M}_1^+}(\pi_{[\min \mathcal{M}_1^-,\max \mathcal{M}_1^+]})$ ends up in the left permutation, $\pi_{[\min \mathcal{M}_1^-,\max \mathcal{M}_1^-]}$, implying the disjointness of $\mathcal{X}_{\mathcal{M}_1^-,\mathcal{M}_1^+}(\pi_{[\min \mathcal{M}_1^-,\max \mathcal{M}_1^+]})$ and $\mathcal{X}_{\mathcal{M}_2^-,\mathcal{M}_2^+}(\pi_{[\min \mathcal{M}_2^-,\max \mathcal{M}_2^+]})$.

C.3.4 Proof of Lemma 3.4.9

We first observe that the pair $(\hat{S}, \hat{\pi})$ is indeed thin. To this end, note that since the permutation $\hat{\pi}$ is a prefix of π, for every $m \in [M]$ we clearly have $\text{cross}_m(\hat{\pi}) \leq \text{cross}_m(\pi) \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$, where the last inequality holds since (S, π) is thin. Next, we show that $(\hat{S}, \hat{\pi})$ satisfies conditions 1-3:

1. *Allowed items*: By construction, $\hat{S} = S \cap (C_{[1,m-1]} \uplus Q_{>m-1})$, implying that $\hat{S}$ forms a subset of $C_{[1,m-1]} \uplus Q_{>m-1}$.

2. *Required crossing items*: An additional implication of our definition of $\hat{S}$ is that $Q_{>m-1} \subseteq \hat{S}$, since $Q_{>m-1} \subseteq S$ by (3.8).

3. *Total profit*: To obtain a lower bound on the profit of $\hat{\pi}$, we observe that

$$\begin{aligned}
\Psi(\hat{\pi}) &= \sum_{i \in \hat{S}} \varphi_{\hat{\pi}}(i) \\
&= \sum_{i \in \hat{S}} \varphi_{\pi}(i) \\
&= \Psi(\pi) - \sum_{i \in S \setminus (C_{[1,m-1]} \uplus Q_{>m-1})} \varphi_{\pi}(i)
\end{aligned}$$

$$\geq \left[\psi_m - \sum_{i \in S \setminus (C_{[1,m-1]} \uplus Q_{>m-1})} \varphi_\pi(i)\right]^+$$
$$= \psi_{m-1} \,.$$

Here, the second equality holds since $\hat{\pi}$ is a prefix of π, as previously mentioned. The third equality follows by noting that $S \setminus \hat{S} = S \setminus (C_{[1,m-1]} \uplus Q_{>m-1})$. The inequality above is obtained by observing that its left-hand-side is non-negative, and by recalling that $(S, \pi) \in \mathrm{Thin}(m, \psi_m, Q_{>m})$, implying that $\Psi(\pi) \geq \psi_m$. The last equality is precisely the definition of ψ_{m-1}.

C.3.5 Proof of Lemma 3.4.10

By way of contradiction, suppose there exists a pair $(\tilde{S}, \tilde{\pi}) \in \mathrm{Thin}(m-1, \psi_{m-1}, Q_{>m-1})$ whose makespan is smaller than that of $\hat{S}$, namely, $w(\tilde{S}) < w(\hat{S})$. We begin by noticing that the item sets $S \setminus \hat{S}$ and $\tilde{S}$ are disjoint, since $S \setminus \hat{S} \subseteq (C_m \uplus Q_{>m}) \setminus Q_{>m-1} \subseteq C_{[m,M]} \setminus Q_{>m-1}$ whereas $\tilde{S} \subseteq C_{[1,m-1]} \uplus Q_{>m-1}$, as $(\tilde{S}, \tilde{\pi}) \in \mathrm{Thin}(m-1, \psi_{m-1}, Q_{>m-1})$. Taking advantage of this observation, we define a new pair $(\tilde{S}^+, \tilde{\pi}^+)$ as follows:

- The underlying set of items is given by $\tilde{S}^+ = \tilde{S} \uplus (S \setminus \hat{S})$.
- The permutation $\tilde{\pi}^+ : \tilde{S}^+ \to [|\tilde{S}^+|]$ is constructed by appending the items in $S \setminus \hat{S}$ to $\tilde{\pi}$, following their internal order in π.

The next claim shows that the resulting pair is a feasible solution to exactly the same subproblem for which (S, π) is optimal.

Claim C.3.1. $(\tilde{S}^+, \tilde{\pi}^+) \in \mathrm{Thin}(m, \psi_m, Q_{>m})$.

Proof. First, we show that $(\tilde{S}^+, \tilde{\pi}^+)$ is a thin pair. To this end, for every $\mu \in [M]$ with $C_\mu \cap \tilde{S}^+ \neq \emptyset$, let $i_\mu \in C_\mu$ be the item that appears last in $\tilde{\pi}^+$ out of this cluster, i.e., $i_\mu = \mathrm{argmax}_{i \in \tilde{S}^+ \cap C_\mu} \tilde{\pi}^+(i)$. We proceed by considering two cases:

- *Item i_μ appears in $\tilde{\pi}$*: By construction, $\tilde{\pi}$ is a prefix of $\tilde{\pi}^+$, and therefore $\text{cross}_\mu(\tilde{\pi}^+) = \text{cross}_\mu(\tilde{\pi}) \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$, where the last inequality holds since $(\tilde{S}, \tilde{\pi})$ is a thin pair.

- *Item i_μ does not appear in $\tilde{\pi}$*: In this case, $i_\mu \in S \setminus \hat{S} \subseteq C_{[m,M]} \setminus Q_{>m-1}$, implying that $\mu \geq m$. Thus, all items in clusters $C_{\mu+1}, \ldots, C_M$ that appear before i_μ in the permutation $\tilde{\pi}^+$ necessarily belong to $Q_{>m}$, and we conclude that $\text{cross}_\mu(\tilde{\pi}^+) \leq |Q_{>m}| \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$.

Next, we show that $(\tilde{S}^+, \tilde{\pi}^+)$ satisfies conditions 1-3:

1. *Allowed items*: First note that $\tilde{S} \subseteq C_{[1,m-1]} \uplus Q_{>m-1} \subseteq C_{[1,m]} \uplus Q_{>m}$, where the first inclusion holds since $(\tilde{S}, \tilde{\pi}) \in \text{Thin}(m-1, \psi_{m-1}, Q_{>m-1})$ and the second follows by definition of $Q_{>m-1}$ in (3.8). In addition, $S \subseteq C_{[1,m]} \uplus Q_{>m}$, since $(S, \pi) \in \text{Thin}(m, \psi_m, Q_{>m})$. Combining these two observations, we have $\tilde{S}^+ = \tilde{S} \uplus (S \setminus \hat{S}) \subseteq C_{[1,m]} \uplus Q_{>m}$ as required.

2. *Required crossing items*: To prove $Q_{>m} \subseteq \tilde{S}^+$, we observe that

$$\begin{aligned} Q_{>m} &\subseteq Q_{>m-1} \uplus (Q_{>m} \setminus Q_{>m-1}) \\ &\subseteq \tilde{S} \cup (S \setminus \hat{S}) \\ &= \tilde{S}^+ . \end{aligned}$$

   To better understand the second inclusion, note that $Q_{>m-1} \subseteq \tilde{S}$, since $(\tilde{S}, \tilde{\pi}) \in \text{Thin}(m-1, \psi_{m-1}, Q_{>m-1})$. In addition, $Q_{>m} \setminus Q_{>m-1} \subseteq S \setminus \hat{S}$, since $Q_{>m} \subseteq S$ due to having $(S, \pi) \in \text{Thin}(m, \psi_m, Q_{>m})$, and since $(Q_{>m} \setminus Q_{>m-1}) \cap \hat{S} = \emptyset$, due to having $Q_{>m} \subseteq C_{[m+1,M]}$ and $\hat{S} \subseteq C_{[1,m-1]} \uplus Q_{>m-1}$, where the latter inclusion holds since $\hat{S} \in \text{Thin}(m-1, \psi_{m-1}, Q_{>m-1})$.

3. *Total profit*: By construction, any item $i \in S \setminus \hat{S}$ appears in the permutation $\tilde{\pi}^+$ after all items in $\tilde{S}$, and moreover, the internal order between the items in $S \setminus \hat{S}$ is determined according to π. Hence, we can bound the completion time of any item $i \in S \setminus \hat{S}$ by noting that

$$C_{\tilde{\pi}^+}(i) = w(\tilde{S}) + \sum_{\substack{j \in S \setminus \hat{S}: \\ \pi(j) < \pi(i)}} w_j$$

$$
\begin{aligned}
&< \; w(\hat{S}) + \sum_{\substack{j \in S \setminus \hat{S}: \\ \pi(j) < \pi(i)}} w_j \\
&= \; C_\pi(i) \; ,
\end{aligned}
$$

where the inequality above follows from our initial assumption that $w(\tilde{S}) < w(\hat{S})$. Consequently, $\varphi_{\tilde{\pi}^+}(i) \geq \varphi_\pi(i)$ for such items, and we have

$$
\begin{aligned}
\Psi\left(\tilde{\pi}^+\right) &= \Psi\left(\tilde{\pi}\right) + \sum_{i \in S \setminus \hat{S}} \varphi_{\tilde{\pi}^+}(i) && \text{(C.2)} \\
&\geq \psi_{m-1} + \sum_{i \in S \setminus \hat{S}} \varphi_\pi(i) && \text{(C.3)} \\
&= \left[\psi_m - \sum_{i \in S \setminus (C_{[1,m-1]} \uplus Q_{>m-1})} \varphi_\pi(i) \right]^+ + \sum_{i \in S \setminus \hat{S}} \varphi_\pi(i) && \text{(C.4)} \\
&\geq \psi_m - \left(\sum_{i \in S \setminus (C_{[1,m-1]} \uplus Q_{>m-1})} \varphi_\pi(i) - \sum_{i \in S \setminus \hat{S}} \varphi_\pi(i) \right) \\
&= \psi_m \; . && \text{(C.5)}
\end{aligned}
$$

Here, equality (C.2) holds since $\tilde{\pi}$ is a prefix of $\tilde{\pi}^+$, with the items in $S \setminus \hat{S}$ forming the remaining suffix. Inequality (C.3) holds since $(\tilde{S}, \tilde{\pi}) \in \text{Thin}(m-1, \psi_{m-1}, Q_{m-1})$, meaning that $\Psi(\tilde{\pi}) \geq \psi_{m-1}$, and since $\varphi_{\tilde{\pi}^+}(i) \geq \varphi_\pi(i)$ for all $i \in S \setminus \hat{S}$, as shown above. Equality (C.4) follows from the definition of ψ_{m-1}. Equality (C.5) is obtained by noting that $S \setminus (C_{[1,m-1]} \uplus Q_{>m-1}) = S \setminus \hat{S}$.

□

Consequently, by combining our initial assumption that $w(\tilde{S}) < w(\hat{S})$ along with Claim C.3.1, we have just identified a pair $(\tilde{S}^+, \tilde{\pi}^+) \in \text{Thin}(m, \psi_m, Q_{>m})$ with a makespan of

$$
\begin{aligned}
w(\tilde{S}^+) &= w(\tilde{S}) + w(S \setminus \hat{S}) \\
&< w(\hat{S}) + w(S \setminus \hat{S}) \\
&= w(S) \; ,
\end{aligned}
$$

contradicting the fact that (S, π) minimizes $w(S)$ over the set $\text{Thin}(m, \psi_m, Q_{>m})$.

C.3.6 Proof of Lemma 3.4.11

Overview. Prior to delving into the nuts-and-bolts of our approach, we provide a high-level overview of its main ideas. For this purpose, to make sure condition 2 of Lemma 3.4.11 is satisfied, meaning that the item set $\hat{\mathcal{E}}$ we compute has a total weight of at most $F(m, \psi_m, Q_{>m}) - F(m - 1, \psi_{m-1}, Q_{>m-1})$, our algorithm relies on "knowing" the latter difference, which will be justified through binary search. With this limitation, restricting ourselves to the item set $(C_m \uplus Q_{>m}) \setminus Q_{>m-1}$, we aim to identify a feasible chain whose associated permutation (ϵ, Δ)-satisfies constraint 2. To this end, our algorithm "guesses" the insertion time of every item in $Q_{>m} \setminus Q_{>m-1}$ by enumerating over all feasible chains $\mathcal{G} = (G_1, \ldots, G_T)$ whose set of introduced items is $G_T = Q_{>m} \setminus Q_{>m-1}$. Since there are at most $\frac{\lceil \log_2 M \rceil}{\epsilon}$ such items, the number of required guesses is only $O(T^{O(\frac{\log M}{\epsilon})})$. For each guess, we construct the residual generalized incremental knapsack instance, as explained in Section 3.3.1, which will be solved to near-optimality via the approximation scheme proposed in Theorem 3.3.1.

Algorithm. For ease of presentation, on top of all input ingredients mentioned in Lemma 3.4.11, we feed into the upcoming algorithm an additional parameter $\omega \geq 0$, whose role will be explained later on. With this parameter, our algorithm operates as follows:

1. We define the generalized incremental knapsack instance $\hat{\mathcal{I}}^{\omega} = (\hat{\mathcal{N}}, \hat{W}^{\omega})$, where:
   - The set of items $\hat{\mathcal{N}}$ is comprised of those allowed by constraint 1, namely, $\hat{\mathcal{N}} = (C_m \uplus Q_{>m}) \setminus Q_{>m-1}$.
   - Additionally, we reduce the capacity W_t of each period $t \in [T]$ by Δ, while ensuring that the maximum resulting capacity does not exceed ω, meaning that $\hat{W}^{\omega}_t = \min\{[W_t - \Delta]^+, \omega\}$.
2. For every feasible chain $\mathcal{G} = (G_1, \ldots, G_T)$ for the instance $\hat{\mathcal{I}}^{\omega}$ with $G_T = Q_{>m} \setminus Q_{>m-1}$, we construct the residual instance $\hat{\mathcal{I}}^{\omega,-\mathcal{G}} = (\hat{\mathcal{N}}^{-\mathcal{G}}, \hat{W}^{\omega,-\mathcal{G}})$. The approximation scheme we

proposed in Section 3.3 is now applied to this instance, thereby obtaining a feasible chain $\mathcal{R}^{\mathcal{G}}$ whose profit is within factor $1 - \epsilon$ of the residual optimum (see Theorem 3.3.1). When there are no feasible chains with $G_T = Q_{>m} \setminus Q_{>m-1}$, we abort and report this finding.

3. Out of all chains $\mathcal{G}$ considered in step 2, let $\mathcal{G}^{\omega}$ be the one for which the sum of profits $\Phi(\mathcal{G}^{\omega}) + \Phi(\mathcal{R}^{\mathcal{G}^{\omega}})$ is maximized. The item set we return is $\mathcal{E}_{\omega} = R_T^{\mathcal{G}^{\omega}} \uplus (Q_{>m} \setminus Q_{>m-1})$, i.e., all items inserted by the chain $\mathcal{R}^{\mathcal{G}^{\omega}}$ along with those in $Q_{>m} \setminus Q_{>m-1}$. We define the corresponding permutation $\pi_{\mathcal{E}_{\omega}} : \mathcal{E}_{\omega} \to [|\mathcal{E}_{\omega}|]$ as the one constructed by Lemma 3.2.1 for the chain $\mathcal{G}^{\omega} \cup \mathcal{R}^{\mathcal{G}^{\omega}}$.

The binary search. We assume without loss of generality that all item weights take integer values. This property can easily be enforced by uniform scaling, which produces an equivalent instance whose input length is polynomial in that of the original instance. Now, knowing in advance that the total weight of any item set is an integer within $[0, w(\mathcal{N})]$, we employ our ω-parameterized algorithm to conduct a binary search over this interval, with the objective of identifying the smallest integer $\omega_{\min}$ such that:

- For $\omega_{\min}$, the algorithm returns a permutation $\pi_{\mathcal{E}_{\omega_{\min}}}$ that satisfies $\sum_{i \in \mathcal{E}_{\omega_{\min}}} \varphi^{+\Delta}_{\pi_{\mathcal{E}_{\omega_{\min}}}}(i) \geq (1 - \epsilon) \cdot (\psi_m - \psi_{m-1})$.

- In contrast, for $\omega_{\min} - 1/2$, the algorithm either aborts at step 2, or returns a permutation $\pi_{\mathcal{E}_{\omega_{\min}-1/2}}$ satisfying $\sum_{i \in \mathcal{E}_{\omega_{\min}-1/2}} \varphi^{+\Delta}_{\pi_{\mathcal{E}_{\omega_{\min}-1/2}}}(i) < (1 - \epsilon) \cdot (\psi_m - \psi_{m-1})$.

To verify that this search procedure is well-defined, let us examine the endpoints of $[0, w(\mathcal{N})]$. For $\omega = 0$, if we obtain a permutation $\pi_{\mathcal{E}_0}$ that satisfies $\sum_{i \in \mathcal{E}_0} \varphi^{+\Delta}_{\pi_{\mathcal{E}_0}}(i) \geq (1 - \epsilon) \cdot (\psi_m - \psi_{m-1})$, our immediate conclusion is that $\omega_{\min} = 0$. For $\omega = w(\mathcal{N})$, as shown in Lemma C.3.2 below, we are guaranteed to obtain a permutation $\pi_{\mathcal{E}_{w(\mathcal{N})}}$ that satisfies $\sum_{i \in \mathcal{E}_{w(\mathcal{N})}} \varphi^{+\Delta}_{\pi_{\mathcal{E}_{w(\mathcal{N})}}}(i) \geq (1-\epsilon) \cdot (\psi_m - \psi_{m-1})$.

Running time. Clearly, the number of binary search iterations we incur is linear in the input size. Now, within each iteration, since there are $O(T)$ guesses for the insertion time of every item

$i \in Q_{>m} \setminus Q_{>m-1}$ and since $|Q_{>m} \setminus Q_{>m-1}| \leq \frac{\lceil \log_2 M \rceil}{\epsilon}$, there are only $O(T^{O(\frac{\log M}{\epsilon})})$ chains $\mathcal{G}$ to consider in step 2. The crucial observation is that, for each such chain, the residual instance $\hat{\mathcal{I}}^{\omega,-\mathcal{G}}$ is defined over the set of items

$$\begin{aligned}
\hat{\mathcal{N}}^{-\mathcal{G}} &= \hat{\mathcal{N}} \setminus G_T \\
&= ((C_m \uplus Q_{>m}) \setminus Q_{>m-1}) \setminus (Q_{>m} \setminus Q_{>m-1}) \\
&\subseteq C_m \setminus Q_{>m-1} \\
&\subseteq C_m \ . \qquad \text{(C.6)}
\end{aligned}$$

Thus, $\hat{\mathcal{I}}^{\omega,-\mathcal{G}}$ is in fact a single-cluster instance, where the weights of any two items differ by a multiplicative factor of at most $n^{1/\epsilon}$, by property 1 of well-spaced instances (see Section 3.4.1). By Theorem 3.3.1, the running time of our approximation scheme for such instances is truly quasi-polynomial, being $O((nT)^{O(\frac{1}{\epsilon^6} \cdot \log n)} \cdot |\mathcal{I}|^{O(1)})$. All in all, we incur a running time of $O((nT)^{O(\frac{1}{\epsilon^6} \cdot (\log n + \log M))} \cdot |\mathcal{I}|^{O(1)})$, with room to spare.

Final solution and analysis. In the remainder of this section, we argue that the item set $\mathcal{E}_{\omega_{\min}}$ and its permutation $\pi_{\mathcal{E}_{\omega_{\min}}} : \mathcal{E}_{\omega_{\min}} \to [|\mathcal{E}_{\omega_{\min}}|]$ satisfy the properties required by Lemma 3.4.11. For this purpose, recalling that the latter lemma assumes $F(m, \psi_m, Q_{>m}) \leq W_T$ and $(m-1, \psi_{m-1}, Q_{>m-1}) = \text{Best}(m, \psi_m, Q_{>m})$, let $\mathcal{E}^*$ and $\pi^*_{\mathcal{E}^*} : \mathcal{E}^* \to [|\mathcal{E}^*|]$ be the item set and permutation attaining the minimum makespan $w(\mathcal{E}^*)$ over $\text{Extra}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}]$, noting that by definition,

$$F(m, \psi_m, Q_{>m}) = F(m-1, \psi_{m-1}, Q_{>m-1}) + w(\mathcal{E}^*) \ . \qquad \text{(C.7)}$$

At the heart of our analysis lies the following claim, showing that whenever the ω-parameterized algorithm is employed with $\omega \geq w(\mathcal{E}^*)$, we obtain a permutation whose Δ-shifted profit is at least $(1-\epsilon) \cdot (\psi_m - \psi_{m-1})$. We provide the proof in Appendix C.3.7.

Lemma C.3.2. *For any $\omega \geq w(\mathcal{E}^*)$, the ω-parameterized algorithm computes an item set $\mathcal{E}_\omega$ and a permutation $\pi_{\mathcal{E}_\omega} : \mathcal{E}_\omega \to [|\mathcal{E}_\omega|]$ that satisfy $\sum_{i \in \mathcal{E}_\omega} \varphi^{+\Delta}_{\pi_{\mathcal{E}_\omega}}(i) \geq (1-\epsilon) \cdot (\psi_m - \psi_{m-1})$.*

With this result in place, the properties required by Lemma 3.4.11 can easily be established, as we show next.

Lemma C.3.3. *The item set $\mathcal{E}_{\omega_{\min}}$ and permutation $\pi_{\mathcal{E}_{\omega_{\min}}}$ satisfy properties 1 and 2.*

Proof. We begin by explaining why $(\mathcal{E}_{\omega_{\min}}, \pi_{\mathcal{E}_{\omega_{\min}}}) \in \text{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$, as stated in property 1:

- *Constraint 1 is satisfied*: We first show that $\mathcal{E}_{\omega_{\min}} \subseteq (C_m \uplus Q_{>m}) \setminus Q_{>m-1}$ and $Q_{>m} \setminus Q_{>m-1} \subseteq \mathcal{E}_{\omega_{\min}}$. Since the item set in question is defined in step 3 as $\mathcal{E}_{\omega_{\min}} = R_T^{\mathcal{G}^{\omega_{\min}}} \uplus (Q_{>m} \setminus Q_{>m-1})$, it suffices to explain why $R_T^{\mathcal{G}^{\omega_{\min}}} \subseteq C_m \setminus Q_{>m-1}$. The latter inclusion follows by noting that $\mathcal{R}^{\mathcal{G}^{\omega_{\min}}}$ is a feasible chain for the instance $\hat{\mathcal{I}}^{\omega_{\min},-\mathcal{G}^{\omega_{\min}}}$, where the set of items is $\hat{\mathcal{N}}^{-\mathcal{G}^{\omega_{\min}}} \subseteq C_m \setminus Q_{>m-1}$, as shown in the first inclusion of (C.6).

- *Constraint 2 is (ϵ, Δ)-satisfied*: To argue that $\sum_{i\in\mathcal{E}_{\omega_{\min}}} \varphi^{+\Delta}_{\pi_{\mathcal{E}_{\omega_{\min}}}}(i) \geq (1-\epsilon)\cdot(\psi_m - \psi_{m-1})$, following Lemma C.3.2, there exists a value $\omega \leq w(\mathcal{N})$ for which $\sum_{i\in\mathcal{E}_\omega} \varphi^{+\Delta}_{\pi_{\mathcal{E}_\omega}}(i) \geq (1-\epsilon)\cdot(\psi_m - \psi_{m-1})$, and the desired claim is implied by the termination condition of our binary search.

We now turn our attention to proving that $w(\mathcal{E}_{\omega_{\min}}) \leq F(m, \psi_m, Q_{>m}) - F(m-1, \psi_{m-1}, Q_{>m-1})$, as stated in property 2. To this end, since $F(m, \psi_m, Q_{>m}) = F(m-1, \psi_{m-1}, Q_{>m-1}) + w(\mathcal{E}^*)$ by equation (C.7), it remains to argue that $w(\mathcal{E}_{\omega_{\min}}) \leq w(\mathcal{E}^*)$. To verify this relation, note that

$$\begin{aligned}
w\left(\mathcal{E}_{\omega_{\min}}\right) &= w\left(R_T^{\mathcal{G}^{\omega_{\min}}} \uplus (Q_{>m} \setminus Q_{>m-1})\right) \\
&= w\left(R_T^{\mathcal{G}^{\omega_{\min}}}\right) + w\left(G_T^{\omega_{\min}}\right) \\
&\leq \hat{W}_T \\
&= \min\left\{[W_T - \Delta]^+, \omega_{\min}\right\} \\
&\leq \omega_{\min} \\
&\leq w\left(\mathcal{E}^*\right) .
\end{aligned}$$

Here, the second equality holds since $G_T^{\omega_{\min}} = Q_{>m} \setminus Q_{>m-1}$, as stated in step 2. The first inequality follows by observing that the chain $\mathcal{R}^{\mathcal{G}^{\omega_{\min}}} \cup \mathcal{G}^{\omega_{\min}}$ is feasible for $\hat{\mathcal{I}}^{\omega_{\min}}$, due to Lemma 3.3.3, meaning in particular that for period T we have $w(R_T^{\mathcal{G}^{\omega_{\min}}}) + w(G_T^{\omega_{\min}}) \leq \hat{W}_T$. The final inequality is derived by combining Lemma C.3.2 and the termination condition of our binary search.

□

C.3.7 Proof of Lemma C.3.2

Constructing a feasible chain for $\hat{\mathcal{I}}^{\omega}$. With respect to the item set $\mathcal{E}^*$ and permutation $\pi^*_{\mathcal{E}^*}$, let us define a chain $\mathcal{S}_*$ for the instance $\hat{\mathcal{I}}^{\omega}$ as follows:

- The collection of inserted items is $S_{*T} = \mathcal{E}^*$.
- The insertion time t_i of each item $i \in S_{*T}$ is the one maximizing p_{it_i} over $\{t \in [T] : W_t \geq F(m-1, \psi_{m-1}, Q_{>m-1}) + C_{\pi^*_{\mathcal{E}^*}}(i)\}$. Note that the latter set is indeed non-empty, since

$$\begin{aligned} C_{\pi^*_{\mathcal{E}^*}}(i) &\leq w(\mathcal{E}^*) \\ &= F(m, \psi_m, Q_{>m}) - F(m-1, \psi_{m-1}, Q_{>m-1}) \\ &\leq W_T - F(m-1, \psi_{m-1}, Q_{>m-1}) \ , \end{aligned}$$

  where the equality above is exactly (C.7), and the last inequality holds since $F(m, \psi_m, Q_{>m}) \leq W_T$, as assumed in Lemma 3.4.11.

The next claim establishes the feasibility and profit guarantee of $\mathcal{S}_*$ with respect to $\hat{\mathcal{I}}^{\omega}$. Below, $\Phi_{\omega}(\cdot)$ stands for the profit function with respect to this instance.

Claim C.3.4. *The chain $\mathcal{S}_*$ is feasible for $\hat{\mathcal{I}}^{\omega}$, with a profit of $\Phi_{\omega}(\mathcal{S}_*) = \sum_{i \in \mathcal{E}^*} \varphi^{\rightsquigarrow}_{\pi^*_{\mathcal{E}^*}}(i)$.*

Proof. To prove the feasibility of $\mathcal{S}_*$, we first observe that, for every time period $t \in [T]$,

$$w(S_{*t}) \leq w(\mathcal{E}^*) \leq \omega \ , \tag{C.8}$$

where the first inequality holds since $S_{*t} \subseteq S_{*T} = \mathcal{E}^*$, and the second inequality is precisely what Lemma C.3.2 assumes. In addition, by definition of $\mathcal{S}_*$, every item $i \in S_{*t}$ is associated with a completion time of $C_{\pi^*_{\mathcal{E}^*}}(i) \leq W_t - F(m-1, \psi_{m-1}, \mathcal{Q}_{>m-1})$. Thus, when the latter difference is negative, we have $S_{*t} = \emptyset$ and therefore $w(S_{*t}) = 0 \leq [W_t - \Delta]^+$. In the opposite case,

$$\begin{aligned} w\left(S_{*t}\right) &\leq W_t - F(m-1, \psi_{m-1}, \mathcal{Q}_{>m-1}) \\ &\leq W_t - \Delta \\ &\leq [W_t - \Delta]^+ , \end{aligned} \tag{C.9}$$

where the second inequality holds since $\Delta \leq F(m-1, \psi_{m-1}, \mathcal{Q}_{>m-1})$, as assumed in Lemma 3.4.11. Putting together inequalities (C.8) and (C.9), we have $w(S_{*t}) \leq \min\{[W_t - \Delta]^+, \omega\} = \hat{W}^\omega_t$, meaning that the chain $\mathcal{S}_*$ is indeed feasible for $\hat{\mathcal{I}}^\omega$.

Now, to derive the profit guarantee $\Phi_\omega(\mathcal{S}_*) = \sum_{i \in \mathcal{E}^*} \varphi^{\rightsquigarrow}_{\pi^*_{\mathcal{E}^*}}(i)$, we observe that since $\Phi_\omega(\mathcal{S}_*) = \sum_{i \in \mathcal{E}^*} p_{it_i}$, it suffices to show that $p_{it_i} = \varphi^{\rightsquigarrow}_{\pi^*_{\mathcal{E}^*}}(i)$ for each item $i \in \mathcal{E}^*$. To this end, note that our choice for the insertion time t_i of each item $i \in \mathcal{E}^*$ exactly follows the definition of $\varphi^{\rightsquigarrow}_{\pi^*_{\mathcal{E}^*}}(i)$, implying that $p_{it_i} = \varphi^{\rightsquigarrow}_{\pi^*_{\mathcal{E}^*}}(i)$. □

Concluding the proof. Having established this claim, we are now ready to show that the item set $\mathcal{E}_\omega$ and permutation $\pi_{\mathcal{E}_\omega}$ satisfy $\sum_{i \in \mathcal{E}_\omega} \varphi^{+\Delta}_{\pi_{\mathcal{E}_\omega}}(i) \geq (1-\epsilon) \cdot (\psi_m - \psi_{m-1})$. For this purpose, similarly to $\Phi_\omega(\cdot)$, let $\Psi_\omega(\cdot)$ be the profit function of a given permutation with respect to the instance $\hat{\mathcal{I}}^\omega$ in its sequencing formulation. With this notation, we obtain the required lower bound by arguing that

$$\begin{aligned} \sum_{i \in \mathcal{E}_\omega} \varphi^{+\Delta}_{\pi_{\mathcal{E}_\omega}}(i) &= \Psi_\omega\left(\pi_{\mathcal{E}_\omega}\right) \\ &\geq \Phi_\omega\left(\mathcal{G}^\omega \cup \mathcal{R}^{\mathcal{G}^\omega}\right) \\ &\geq (1-\epsilon) \cdot (\psi_m - \psi_{m-1}) . \end{aligned}$$

We prove the first equality and second inequality in Claims C.3.5 and C.3.6, respectively. To understand the first inequality, recall that the permutation $\pi_{\mathcal{E}_\omega}$ is constructed in step 3 according to Lemma 3.2.1 for the chain $\mathcal{G}^\omega \cup \mathcal{R}^{\mathcal{G}^\omega}$, which guarantees $\Psi_\omega(\pi_{\mathcal{E}_\omega}) \geq \Phi_\omega(\mathcal{G}^\omega \cup \mathcal{R}^{\mathcal{G}^\omega})$.

Claim C.3.5. $\sum_{i \in \mathcal{E}_\omega} \varphi^{+\Delta}_{\pi_{\mathcal{E}_\omega}}(i) = \Psi_\omega(\pi_{\mathcal{E}_\omega})$.

Proof. Let us use $\varphi^{\omega}_{\pi_{\mathcal{E}_\omega}}(i)$ to denote the profit contribution of item i with respect to the permutation $\pi_{\mathcal{E}_\omega}$ in the instance $\hat{\mathcal{I}}^\omega$. In other words, $\varphi^{\omega}_{\pi_{\mathcal{E}_\omega}}(i) = \max\{p_{it} : t \in [T+1] \text{ and } \hat{W}_t \geq C_{\pi_{\mathcal{E}_\omega}}(i)\}$. With this notation, we have $\Psi_\omega(\pi_{\mathcal{E}_\omega}) = \sum_{i \in \mathcal{E}_\omega} \varphi^{\omega}_{\pi_{\mathcal{E}_\omega}}(i)$, meaning that to prove the desired equality, it remains to show that $\varphi^{+\Delta}_{\pi_{\mathcal{E}_\omega}}(i) = \varphi^{\omega}_{\pi_{\mathcal{E}_\omega}}(i)$ for every item $i \in \mathcal{E}_\omega$. To verify this claim, note that

$$\begin{aligned}
\varphi^{+\Delta}_{\pi_{\mathcal{E}_\omega}}(i) &= \max\left\{p_{it} : t \in [T+1] \text{ and } W_t - \Delta \geq C_{\pi_{\mathcal{E}_\omega}}(i)\right\} \\
&= \max\left\{p_{it} : t \in [T+1] \text{ and } [W_t - \Delta]^+ \geq C_{\pi_{\mathcal{E}_\omega}}(i)\right\} \\
&= \max\left\{p_{it} : t \in [T+1] \text{ and } \min\{[W_t - \Delta]^+, \omega\} \geq C_{\pi_{\mathcal{E}_\omega}}(i)\right\} \\
&= \max\left\{p_{it} : t \in [T+1] \text{ and } \hat{W}_t \geq C_{\pi_{\mathcal{E}_\omega}}(i)\right\} \\
&= \varphi^{\omega}_{\pi_{\mathcal{E}_\omega}}(i) \ .
\end{aligned}$$

Here, the second equality holds since $C_{\pi_{\mathcal{E}_\omega}}(i) \geq 0$. The third equality is obtained by noting that $C_{\pi_{\mathcal{E}_\omega}}(i) \leq w(\mathcal{E}_\omega) = w(G^\omega_T) + w(R^{\mathcal{G}^\omega}_T) \leq \hat{W}_T \leq \omega$, where the equality follows by definition of $\mathcal{E}_\omega$ and the second inequality is implied by the feasibility of $\mathcal{G}^\omega \cup \mathcal{R}^{\mathcal{G}^\omega}$ for the instance $\hat{\mathcal{I}}^\omega$. The last two equalities follow from the definitions of $\hat{W}_t$ and $\varphi^{\omega}_{\pi_{\mathcal{E}_\omega}}(i)$. □

Claim C.3.6. $\Phi_\omega(\mathcal{G}^\omega \cup \mathcal{R}^{\mathcal{G}^\omega}) \geq (1-\epsilon) \cdot (\psi_m - \psi_{m-1})$.

Proof. We begin by noting that since $(\mathcal{E}^*, \pi^*_{\mathcal{E}^*}) \in \text{Extra}[\,{}^{(m,\psi_m,\mathcal{Q}_{>m})}_{(m-1,\psi_{m-1},\mathcal{Q}_{>m-1})}]$, this item set and permutation necessarily satisfy constraint 1, which informs us that $\mathcal{E}^* \subseteq (C_m \uplus \mathcal{Q}_{>m}) \setminus \mathcal{Q}_{>m-1}$ and $\mathcal{Q}_{>m} \setminus \mathcal{Q}_{>m-1} \subseteq \mathcal{E}^*$. As a result, recalling that the collection of items introduced by the chain $\mathcal{S}_*$ is precisely $\mathcal{E}^*$, it follows that the latter chain can be expressed as $\mathcal{S}_* = \mathcal{S}_*|_{\mathcal{Q}_{>m} \setminus \mathcal{Q}_{>m-1}} \cup \mathcal{S}_*|_{C_m \setminus \mathcal{Q}_{>m-1}}$. We remind the reader that, based on the terminology of Section 3.3, the first term $\mathcal{S}_*|_{\mathcal{Q}_{>m} \setminus \mathcal{Q}_{>m-1}}$

is the restriction of $\mathcal{S}_*$ to the items in $Q_{>m} \setminus Q_{>m-1}$, whereas the second term $\mathcal{S}_*|_{C_m \setminus Q_{>m-1}}$ is its restriction to $C_m \setminus Q_{>m-1}$.

The crucial observation is that, since the chain $\mathcal{S}_*$ introduces all items in $Q_{>m} \setminus Q_{>m-1}$, its restriction $\mathcal{G}_* = \mathcal{S}_*|_{Q_{>m} \setminus Q_{>m-1}}$ is necessarily considered in step 2 of our algorithm; moreover, $\mathcal{S}_*|_{C_m \setminus Q_{>m-1}}$ constitutes a feasible chain for the residual instance $\hat{\mathcal{I}}^{\omega,-\mathcal{G}_*}$, by Lemma 3.3.4. As such, the corresponding chain $\mathcal{R}^{\mathcal{G}_*}$ we compute for the latter instance is guaranteed to have a profit of $\Phi_\omega(\mathcal{R}^{\mathcal{G}_*}) \geq (1-\epsilon) \cdot \Phi_\omega(\mathcal{S}_*|_{C_m \setminus Q_{>m-1}})$. Consequently, since the chain $\mathcal{G}^\omega$ is the one maximizing $\Phi_\omega(\mathcal{G}^\omega) + \Phi_\omega(\mathcal{R}^{\mathcal{G}^\omega})$ over all chains considered in step 2, we conclude that $\mathcal{G}^\omega \cup \mathcal{R}^{\mathcal{G}^\omega}$ is a feasible chain for $\hat{\mathcal{I}}^\omega$ with a profit of

$$\begin{aligned}
\Phi_\omega\left(\mathcal{G}^\omega \cup \mathcal{R}^{\mathcal{G}^\omega}\right) &= \Phi_\omega\left(\mathcal{G}^\omega\right) + \Phi_\omega\left(\mathcal{R}^{\mathcal{G}^\omega}\right) \\
&\geq \Phi_\omega\left(\mathcal{G}_*\right) + \Phi_\omega\left(\mathcal{R}^{\mathcal{G}_*}\right) \\
&\geq \Phi_\omega\left(\mathcal{S}_*|_{Q_{>m} \setminus Q_{>m-1}}\right) + (1-\epsilon) \cdot \Phi_\omega\left(\mathcal{S}_*|_{C_m \setminus Q_{>m-1}}\right) \\
&\geq (1-\epsilon) \cdot \Phi_\omega\left(\mathcal{S}_*\right) \\
&= (1-\epsilon) \cdot \sum_{i \in \mathcal{E}^*} \varphi^{\rightsquigarrow}_{\pi^*_{\mathcal{E}^*}}(i) \\
&\geq (1-\epsilon) \cdot (\psi_m - \psi_{m-1}) \ .
\end{aligned}$$

Here, the first and second equalities follow from Lemma 3.3.3 and Claim C.3.4, respectively. The last inequality holds since $(\mathcal{E}^*, \pi^*_{\mathcal{E}^*}) \in \mathrm{Extra}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}]$ by definition, and hence, constraint 2 is necessarily satisfied. □

C.3.8 Proof of Lemma 3.4.12

We prove the lemma by induction on m.

Base case: $m = 0$**.** In this case, for any state with $F(0, \psi_0, Q_{>0}) \leq W_T$, we actually have $\hat{F}(0, \psi_0, Q_{>0}) = F(0, \psi_0, Q_{>0})$, by the way terminal states of $\hat{F}$ are handled. In addition, letting $\hat{\pi}_{\hat{S}_0}$ be the permutation of $\hat{S}_0 = Q_{>0}$ that attains $\hat{F}(0, \psi_0, Q_{>0})$, it follows that $\hat{S}_0 \subseteq C_{[1,0]} \uplus Q_{>0}$,

$Q_{>0} \subseteq \hat{S}_0$, and $\Psi(\hat{\pi}_{\hat{S}_0}) \geq \psi_0$, again by definition.

General case: $m \geq 1$**.** Let $(m, \psi_m, Q_{>m})$ be a state for which $F(m, \psi_m, Q_{>m}) \leq W_T$. We first show that $\hat{F}(m, \psi_m, Q_{>m}) \leq F(m, \psi_m, Q_{>m})$. To this end, recall that the function value $\hat{F}(m, \psi_m, Q_{>m})$ is determined by minimizing $\hat{F}(m-1, \psi_{m-1}, Q_{>m-1}) + w(\hat{\mathcal{E}})$ over all conceivable states $(m-1, \psi_{m-1}, Q_{>m-1})$, where the item set $\hat{\mathcal{E}}$ and its permutation $\hat{\pi}_{\hat{\mathcal{E}}} : \hat{\mathcal{E}} \to [|\hat{\mathcal{E}}|]$ are obtained by instantiating Lemma 3.4.11 with $\Delta = \hat{F}(m-1, \psi_{m-1}, Q_{>m-1})$ and satisfy $(\hat{\mathcal{E}}, \hat{\pi}_{\hat{\mathcal{E}}}) \in \mathrm{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$. Therefore, specifically for the state $(m-1, \psi^*_{m-1}, Q^*_{>m-1}) = \mathrm{Best}(m, \psi_m, Q_{>m})$, we have $\Delta = \hat{F}(m-1, \psi^*_{m-1}, Q^*_{>m-1}) \leq F(m-1, \psi^*_{m-1}, Q^*_{>m-1})$ by the induction hypothesis. In turn, our auxiliary procedure computes a corresponding item set and permutation $(\hat{\mathcal{E}}^*, \hat{\pi}^*_{\hat{\mathcal{E}}}) \in \mathrm{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi^*_{m-1},Q^*_{>m-1})}\,]$ with total weight $w(\hat{\mathcal{E}}^*) \leq F(m, \psi_m, Q_{>m}) - F(m-1, \psi^*_{m-1}, Q^*_{>m-1})$, as guaranteed by Lemma 3.4.11. Consequently,

$$\begin{aligned}
\hat{F}(m, \psi_m, Q_{>m}) &\leq \hat{F}\left(m-1, \psi^*_{m-1}, Q^*_{>m-1}\right) + w\left(\hat{\mathcal{E}}^*\right) \\
&\leq F\left(m-1, \psi^*_{m-1}, Q^*_{>m-1}\right) \\
&\quad + \left(F(m, \psi_m, Q_{>m}) - F\left(m-1, \psi^*_{m-1}, Q^*_{>m-1}\right)\right) \\
&= F(m, \psi_m, Q_{>m}) \,,
\end{aligned}$$

which is precisely the required upper bound on $\hat{F}(m, \psi_m, Q_{>m})$.

Next, we show that $\hat{F}(m, \psi_m, Q_{>m})$ is attained by an item set $\hat{S}_m$ and a permutation $\hat{\pi}_{\hat{S}_m}$ satisfying $\hat{S}_m \subseteq C_{[1,m]} \uplus Q_{>m}$, $Q_{>m} \subseteq \hat{S}_m$, and $\Psi(\hat{\pi}_{\hat{S}_m}) \geq (1-\epsilon) \cdot \psi_m$. For this purpose, let $(m-1, \psi_{m-1}, Q_{>m-1})$, $\hat{\mathcal{E}}$, and $\hat{\pi}_{\hat{\mathcal{E}}}$ be the conceivable state, item set, and permutation at which $\hat{F}(m, \psi_m, Q_{>m}) = \hat{F}(m-1, \psi_{m-1}, Q_{>m-1}) + w(\hat{\mathcal{E}})$ is attained, meaning in particular that $Q_{>m-1} \setminus C_m \subseteq Q_{>m}$ by definition of conceivable states, and that $(\hat{\mathcal{E}}, \hat{\pi}_{\hat{\mathcal{E}}}) \in \mathrm{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$ by the way general states of $\hat{F}$ are handled. We proceed by observing that, by the induction hypothesis, $\hat{F}(m-1, \psi_{m-1}, Q_{>m-1})$ is attained by an item set $\hat{S}_{m-1}$ and a permutation $\hat{\pi}_{\hat{S}_{m-1}}$ satisfying $\hat{S}_{m-1} \subseteq C_{[1,m-1]} \uplus Q_{>m-1}$, $Q_{>m-1} \subseteq \hat{S}_{m-1}$, and $\Psi(\hat{\pi}_{\hat{S}_{m-1}}) \geq (1-\epsilon) \cdot \psi_{m-1}$. With these ingredients,

let us define the item set $\hat{S}_m$ and permutation $\hat{\pi}_{\hat{S}_m}$ as follows:

- The item set $\hat{S}_m$ is given by $\hat{S}_m = \hat{S}_{m-1} \uplus \hat{\mathcal{E}}$. To understand why $\hat{S}_{m-1}$ and $\hat{\mathcal{E}}$ are disjoint, recall that $(\hat{\mathcal{E}}, \hat{\pi}_{\hat{\mathcal{E}}}) \in \mathrm{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}]$, which implies by constraint 1 that $\hat{\mathcal{E}} \subseteq (C_m \uplus Q_{>m}) \setminus Q_{>m-1} \subseteq C_{[m,M]} \setminus Q_{>m-1}$; however, $\hat{S}_{m-1} \subseteq C_{[1,m-1]} \uplus Q_{>m-1}$ by the induction hypothesis. These observations allow us to concurrently argue that $\hat{S}_m \subseteq C_{[1,m]} \uplus Q_{>m}$ as required, since $\hat{\mathcal{E}} \subseteq (C_m \uplus Q_{>m}) \setminus Q_{>m-1} \subseteq C_{[1,m]} \uplus Q_{>m}$ and since

$$\begin{aligned} \hat{S}_{m-1} &\subseteq C_{[1,m-1]} \uplus Q_{>m-1} \\ &\subseteq C_{[1,m]} \uplus (Q_{>m-1} \setminus C_m) \\ &\subseteq C_{[1,m]} \uplus Q_{>m} \,, \end{aligned}$$

  where the last inclusion follows by noting that $Q_{>m-1} \setminus C_m \subseteq Q_{>m}$ due to state $(m-1, \psi_{m-1}, Q_{>m-1})$ being conceivable. In addition,

$$\begin{aligned} Q_{>m} &\subseteq Q_{>m-1} \uplus (Q_{>m} \setminus Q_{>m-1}) \\ &\subseteq \hat{S}_{m-1} \uplus \hat{\mathcal{E}} \\ &= \hat{S}_m \,, \end{aligned}$$

  where the second inclusion holds since $Q_{>m-1} \subseteq \hat{S}_{m-1}$ by the induction hypothesis and since $Q_{>m} \setminus Q_{>m-1} \subseteq \hat{\mathcal{E}}$, again by constraint 1.

- To define the permutation $\hat{\pi}_{\hat{S}_m} : \hat{S}_m \to [|\hat{S}_m|]$, we simply append $\hat{\pi}_{\hat{\mathcal{E}}}$ to $\hat{\pi}_{\hat{S}_{m-1}}$. As a result, we obtain a profit of

$$\begin{aligned} \Psi\left(\hat{\pi}_{\hat{S}_m}\right) &= \Psi\left(\hat{\pi}_{\hat{S}_{m-1}}\right) + \sum_{i \in \hat{\mathcal{E}}} \varphi^{+w(\hat{S}_{m-1})}_{\hat{\pi}_{\hat{\mathcal{E}}}}(i) \\ &= \Psi\left(\hat{\pi}_{\hat{S}_{m-1}}\right) + \sum_{i \in \hat{\mathcal{E}}} \varphi^{+\Delta}_{\hat{\pi}_{\hat{\mathcal{E}}}}(i) \\ &\geq (1-\epsilon) \cdot \psi_{m-1} + (1-\epsilon) \cdot (\psi_m - \psi_{m-1}) \end{aligned}$$

$$= \quad (1-\epsilon)\cdot\psi_m \ .$$

Here, the second equality holds since $w(\hat{S}_{m-1}) = \hat{F}(m-1, \psi_{m-1}, Q_{>m-1}) = \Delta$. To understand the inequality above, note that $\Psi(\hat{\pi}_{\hat{S}_{m-1}}) \geq (1-\epsilon)\cdot\psi_{m-1}$ by the inductive hypothesis, and in addition, $\sum_{i\in\hat{\mathcal{E}}} \varphi^{+\Delta}_{\hat{\pi}_{\hat{\mathcal{E}}}}(i) \geq (1-\epsilon)\cdot(\psi_m - \psi_{m-1})$, since $(\hat{\mathcal{E}}, \hat{\pi}_{\hat{\mathcal{E}}}) \in \mathrm{Extra}_{\epsilon,\Delta}[\,{}^{(m,\psi_m,Q_{>m})}_{(m-1,\psi_{m-1},Q_{>m-1})}\,]$ implies that constraint 2 is (ϵ,Δ)-satisfied.

Appendix D: Some easier, and some not harder, incremental knapsack problems

D.1 Additional proofs from Section 4.2

D.1.1 Proof of Lemma 4.2.2

We give the following constructive proof to find a cycle $C \subseteq S$:

1. Let $C = S$.
2. While there exists $i \in C$ such that $C \setminus \{i\}$ is dependent, let $C = C \setminus \{i\}$.
3. Output C.

We claim the C outputted by the procedure above is a cycle. By construction, each element is removed from C only if the resulting set is still dependent. So, C is dependent. Also by construction, at termination, $C \setminus \{i\}$ is independent for all $i \in C$. Finally, the procedure must terminate with $|C| \geq 2$, since we assume all singletons are independent.

D.1.2 Proof of Lemma 4.2.3

By hypothesis, $\gamma(S) = \sum_{j \in S} p_j$. Now let $S' \subseteq S$ be dependent. Take a cycle $C \subseteq S'$, whose existence is guaranteed by Lemma 4.2.2, and let $i \in C$. By definition of a cycle, $C \setminus \{i\}$ is independent and C is dependent, hence

$$\gamma(C) = \gamma(C \setminus \{i\}) = \sum_{j \in C \setminus \{i\}} p_j.$$

Thus,

$$0 = \gamma(C) - \gamma(C \setminus \{i\}) \geq \gamma(S) - \gamma(S \setminus \{i\}) = p_i,$$

where the first equality follows from the equation above, the inequality by submodularity, and the second equality by independence of S. Hence, $p_i \leq 0$, a contradiction.

D.1.3 Proof of Lemma 4.2.4

If S is independent, take $S' = S$ and the statement is trivial. Now, assume, S is dependent. Suppose by contradiction that for every independent set $S' \subseteq S$, we have $\gamma(S) > \gamma(S')$ by monotonicity. The existence of S' is guaranteed since we assume every set of cardinality 1 is independent. Let $\bar{S} \subseteq S$ be an independent set of maximum cardinality and let $\{e_1, \ldots, e_k\} = S \setminus \bar{S}$. For every $e \in S \setminus \bar{S}$, we must have $\gamma(\bar{S} \cup \{e\}) = \gamma(\bar{S})$, else $\bar{S} \cup \{e\}$ is independent as well, a contradiciton to the maximality of $\bar{S}$.

Thus, we have by submodularity that, for $i \in [k]$,

$$0 = \gamma(\bar{S} \cup \{e\}) - \gamma(\bar{S}) \geq \gamma(\bar{S} \cup \{e_1, \ldots, e_i\}) - \gamma(\bar{S} \cup \{e_1, \ldots, e_{i-1}\}).$$

Hence by telescoping sum

$$\gamma(S) - \gamma(\bar{S}) = \gamma(\bar{S} \cup \{e_1, \ldots, e_k\}) - \gamma(\bar{S}) \leq 0,$$

contradicting $\gamma(S) > \gamma(\bar{S})$.

D.2 Additional proofs from Section 4.3

D.2.1 Proof of Claim 4.3.2

We will first show that $\bar{x}$ has at least one less fractional component than x^*. First note that, for all $(i,t) \in [n] \times [T]$ with $x^*_{i,t} \in \{0,1\}$, we have $\bar{x}_{i,t} = x^*_{i,t}$. Hence, the number of fractional components in $\bar{x}$ is at most the number of fractional components in x^*. If $\epsilon = 1 - x^*_{k,t''}$, then $\bar{x}_{k,t''} = 1$, while $x^*_{k,t''}$ is fractional by construction. On the other hand, if $\epsilon = x^*_{j,t'} \frac{w_j}{w_k}$, then $\bar{x}_{j,t'} = x^*_{j,t'} - x^*_{j,t'} \frac{w_j}{w_k} (\frac{w_k}{w_j}) = 0$, and again $x^*_{j,t'}$ is fractional by construction. This proves the statement on the number of fractional

components.

We now show feasibility of $\bar{x}$. First, we check that for all $(i,t) \in [n] \times [T]$, $0 \leq x_{i,t} \leq 1$. For $t = t', \ldots, t''$, we have:

$$\bar{x}_{j,t} < x^*_{j,t} < 1,$$

and

$$\bar{x}_{j,t} = x^*_{j,t} - \epsilon \frac{w_k}{w_j} \geq x^*_{j,t} - (x^*_{j,t'} \frac{w_j}{w_k}) \frac{w_k}{w_j} = x^*_{j,t} - x^*_{j,t'} \geq 0,$$

where the first inequality follows by definition of ϵ, and the second inequality follows since x^* is feasible so $x^*_{j,t} \geq x^*_{j,t'}$ for all $t \geq t'$.

Similarly, we have, for $t = t', \ldots, t''$:

$$\bar{x}_{k,t} > x^*_{k,t} \geq 0,$$

and

$$\bar{x}_{k,t} = x^*_{k,t} + \epsilon \leq x^*_{k,t} + 1 - x^*_{k,t''} \leq x^*_{k,t''} + 1 - x^*_{k,t''} = 1,$$

where the first inequality follows by definition of ϵ and the second inequality follows since x^* is feasible so $x^*_{k,t} \leq x^*_{k,t''}$ for all $t \leq t''$. For all other $(i,t) \in [n] \times [T]$, $\bar{x}_{i,t} = x^*_{i,t}$. Thus $\bar{x}$ satisfy the constraints $0 \leq \bar{x}_{i,t} \leq 1$ for all $(i,t) \in [n] \times [T]$.

Now, we check that $\bar{x}_{i,t} \leq \bar{x}_{i,t+1}$ for $i \in [n]$ and $t \in [T-1]$. Let $i = j, k$. For $t = t', \ldots, t'' - 1$

$$\bar{x}_{i,t+1} - \bar{x}_{i,t} = x^*_{i,t+1} - x^*_{i,t} \geq 0,$$

again by feasibility of x^*. Moreover, $\bar{x}_{i,t} = x^*_{i,t} = 0$ for $t \leq t' - 1$ and $\bar{x}_{i,t} = x^*_{i,t} = 1$ for $t \geq t'' + 1$, showing that $\bar{x}_{i,t} \leq \bar{x}_{i,t+1}$ is satisfied for $i = j, k$ and $t \in [T-1]$. For $i \notin \{j,k\}$, $\bar{x}_{i,t} = x^*_{i,t}$ for all $t \in [T]$. Therefore the constraint $\bar{x}_{i,t} \leq \bar{x}_{i,t+1}$ is satisfied due to feasibility of x^*.

Lastly,we check that $\sum_{i\in[n]} w_i x_{i,t} \le W_t$ is satisfied for all $t \in [T]$. For $t \le t' - 1$ or $t \ge t'' + 1$:

$$\sum_{i\in[n]} w_i \bar{x}_{i,t} = \sum_{i\in[n]} w_i x_{i,t} \le W_t.$$

For $t = t', \ldots, t''$:

$$\begin{aligned}
\sum_{i\in[n]} w_i \bar{x}_{i,t} &= \sum_{i\in[n]\setminus\{j,k\}} w_i x^*_{i,t} + w_j \bar{x}_{j,t} + w_k \bar{x}_{k,t} \\
&= \sum_{i\in[n]\setminus\{j,k\}} w_i x^*_{i,t} + w_j (x^*_{j,t} - \epsilon \frac{w_k}{w_j}) + w_k (x^*_{k,t} + \epsilon) \\
&= \sum_{i\in[n]\setminus\{j,k\}} w_k x^*_{t,k} + w_i x^*_{j,t} + w_j x^*_{k,t} \le W_t,
\end{aligned}$$

concluding the proof.

D.2.2 Proof of Claim 4.3.3

Observe that the difference between the profit of $\bar{x}$ and x^* is given by

$$\begin{aligned}
&\sum_{i\in[n]} \sum_{t\in[T]} p_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) - \sum_{i\in[n]} \sum_{t\in[T]} p_{i,t}(x^*_{i,t} - x^*_{i,t-1}) \\
&\quad= \sum_{i\in[n]} \sum_{t\in[T]} p_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1} - x^*_{i,t} + x^*_{i,t-1}) \\
&\quad= \sum_{i=j,k} \sum_{t'\le t\le t''+1} p_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1} - x^*_{i,t} + x_{i,t-1}) \\
&\quad= \sum_{i=j,k} \left[p_{i,t'}(\bar{x}_{i,t'} - \bar{x}_{i,t'-1} - x^*_{i,t'} + x^*_{i,t'-1}) + p_{i,t''+1}(\bar{x}_{i,t''+1} - \bar{x}_{i,t''} - x^*_{i,t''+1} + x^*_{i,t''}) \right] \\
&\quad= \sum_{i=j,k} \left[p_{i,t'}(\bar{x}_{i,t'} - x^*_{i,t'}) + p_{i,t''+1}(-\bar{x}_{i,t''} + x^*_{i,t''}) \right] \\
&\quad= p_{k,t'} \cdot \epsilon - p_{k,t''+1} \cdot \epsilon - p_{j,t'} \cdot \epsilon \frac{w_k}{w_j} + p_{j,t''+1} \cdot \epsilon \frac{w_k}{w_j} \\
&\quad= \epsilon(p_{k,t'} - p_{k,t''+1}) - \epsilon \frac{w_k}{w_j}(p_{j,t'} - p_{j,t''+1})
\end{aligned}$$

$$\geq 0,$$

where the second equality follows from $\bar{x}_{i,t} = x^*_{i,t}$ for $t \notin \{t', \ldots, t''\}$ or $i \neq j, k$; the third equality follows from $\bar{x}_{i,t} - \bar{x}_{i,t-1} = x^*_{t,k} - x^*_{t-1,k}$ for $i = j, k$ and $t = t'+1, \ldots, t''$; the fourth equality follows since $\bar{x}_{i,t'-1} = x^*_{i,t'-1}$ and $\bar{x}_{i,t''+1} = x^*_{i,t''+1}$; the fifth equality from the definition of $\bar{x}$; the inequality follows by (4.8). Since $\bar{x}$ gives a higher objective function value than x^*, the optimality of $\bar{x}$ follows from the optimality of x^*.

D.2.3 Proof of Claim 4.3.5

By definition,

$$\Phi(\mathcal{R}) = \sum_{i \in R_T} p_{i,t^*(i)}.$$

By construction, since G_T contains the first $\frac{T^2(T+1)}{2\epsilon}$ items in S^*_T ranked by non-increasing order of $p_{i,t^*(i)}$. It follows that if $i \in R_T$, then $p_{i,t^*(i)} \leq p^-$. Thus, in the residual instance $\mathcal{I}^{-\mathcal{G}}_{p^-}$, $p_{i,t^*(i)}$ remains unchanged, implying that

$$\Phi_{p^-}(\mathcal{R}) = \Phi(\mathcal{R}).$$

D.2.4 Proof of Claim 4.3.6

Let $\bar{x}$ be the optimal solution given by Lemma 4.3.1. Let x' be the integer solution corresponding to $\mathcal{R}'$ obtained by rounding down $\bar{x}$. Thus, $\Phi_{p^-}(\mathcal{R}') = \sum_{i \in \mathcal{N}^{-\mathcal{G}}} \sum_{t \in [T]} p^-_{i,t}(x'_{i,t} - x'_{i,t-1})$.

For each $t \in [T]$, let $I_t = \{i \in \mathcal{N}^{-\mathcal{G}} : \bar{x}_{i,t} = 1\}$ and let $F_t = \{i \in \mathcal{N}^{-\mathcal{G}} : 0 < \bar{x}_{i,t} < 1\}$. We have

$$\begin{aligned}
\sum_{i \in \mathcal{N}^{-\mathcal{G}}} \sum_{t \in [T]} p^-_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) - \sum_{t \in [T]} \sum_{i \in F_t} p^-_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) &= \sum_{t \in [T]} \sum_{i \in I_t} p^-_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) \\
&\leq \sum_{t \in [T]} \sum_{i \in I_t} p^-_{i,t}(x'_{i,t} - x'_{i,t-1}) \\
&\leq \sum_{i \in \mathcal{N}^{-\mathcal{G}}} \sum_{t \in [T]} p^-_{i,t}(x'_{i,t} - x'_{i,t-1}) \quad \text{(D.1)}
\end{aligned}$$

where the first equality follows by noting that for $i \in \mathcal{N}^{-\mathcal{G}} \setminus (I_t \cup F_t)$, $\bar{x}_{i,t} = \bar{x}_{i,t-1} = 0$. The first

inequality follows since for $i \in I_t$, $x'_{i,t} = \bar{x}_{i,t}$ and $x'_{i,t-1} \leq \bar{x}_{i,t-1}$. Furthermore, let $p^-_{\max} = \max\{p^-_{i,t} : i \in \mathcal{N}^{-\mathcal{G}}, t \in [T]\}$, we have:

$$\begin{aligned}
\sum_{t\in[T]} \sum_{i\in F_t} p^-_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) &\leq \frac{T^2(T+1)}{2} p^-_{\max} \\
&\leq \frac{T^2(T+1)}{2} p^- \\
&\leq \epsilon\Phi(\mathcal{G}) \qquad \text{(D.2)}
\end{aligned}$$

where the first inequality follows since by Lemma 4.3.1, $\bar{x}$ has at most $\frac{T^2(T+1)}{2}$ fractional components, thus $\sum_{t\in[T]} |F_t| \leq \frac{T^2(T+1)}{2}$. The final inequality follows since by definition of $\Phi(\mathcal{G}) \geq |G_T|p^- = \frac{T^2(T+1)}{2\epsilon} p^-$. Putting together the above results, we deduce:

$$\begin{aligned}
\Phi_{p^-}(\mathcal{R}') &= \sum_{i\in\mathcal{N}^{-\mathcal{G}}} \sum_{t\in[T]} p^-_{i,t}(x'_{i,t} - x'_{i,t-1}) \\
&\geq \sum_{i\in\mathcal{N}^{-\mathcal{G}}} \sum_{t\in[T]} p^-_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) - \sum_{t\in[T]} \sum_{i\in F_t} p^-_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) \\
&\geq \sum_{i\in\mathcal{N}^{-\mathcal{G}}} \sum_{t\in[T]} p^-_{i,t}(\bar{x}_{i,t} - \bar{x}_{i,t-1}) - \epsilon\Phi(\mathcal{G}) \\
&\geq \Phi_{p^-}(\mathcal{R}) - \epsilon\Phi(\mathcal{G}),
\end{aligned}$$

where the equality follows by construction of $\mathcal{R}'$, the first inequality follows by (D.1), the second inequality follows by (D.2), and the final inequality follows since $\mathcal{R}$ is a feasible solution of $\mathcal{I}^{-\mathcal{G}}_{p^-}$.

D.3 Additional proof from Section 4.4

D.3.1 Proof of Lemma 4.4.1

For all $t \in [T]$, let $S_t = \{j \in S : t(j) \leq t\}$. Clearly $\mathcal{S}$ can be constructed in $O(nT)$ time.

Note that by construction, for every $t \in [T]$, if $j \in S_t \setminus S_{t-1}$, $t(j) = t$. It follows that $\Phi(\mathcal{S}) = \sum_{t\in[T]} \sum_{j\in S_t\setminus S_{t-1}} p_{j,t} = \sum_{j\in S} p_j \geq p$. It remains to show that $\mathcal{S}$ is feasible. For every $t \in [T]$,

$$w(S_t) = \sum_{j \in S : t(j) \leq t} w_j = \sum_{j \in S \cap [n_t]} w_j \leq W_t,$$

where the inequality is exactly condition 2.

Appendix E: Single-machine algorithms for incremental packing problems

E.1 Proof of Lemma 5.3.2

First, we prove Point 2, where $m = 1$. Given an instance of the single-bin incremental packing problem $\mathcal{I}$, we construct a single-machine sequencing problem instance $\mathcal{I}'$ as follows. For every $S \in 2^{[n]}$, permutation π of S and $i \in S$, let $C^i_\pi = \{k \in S : \pi(k) \leq \pi(i)\}$. Let the profit of item i with respect to permutation π be $\varphi_\pi(i) = \max\{p_{i,t} : t \in [T+1] \text{ and } C^i_\pi \in \mathcal{F}_t\}$, where by convention $\mathcal{F}_{T+1} = 2^{[n]}$ and $p_{i,T+1} = 0$. For any $i \notin S$, let $\varphi_\pi(i) = 0$.

Note that the profit function satisfies inclusionwise non-increasing profit profile. Property 1 is satisfied by definition. To check Property 2, for some π, π' and $i \in [n]$ such that i is in the domain of both π and π', if $\{j : \pi(j) \leq \pi(i)\} \subseteq \{j : \pi'(j) \leq \pi'(i)\}$, then $C^i_\pi \subseteq C^i_{\pi'}$. Thus, for any $t \in [T]$, if $C^i_{\pi'} \in \mathcal{F}_t$, then $C^i_\pi \in \mathcal{F}_t$. It follows that $\varphi_\pi(i) \geq \varphi_{\pi'}(i)$. Moreover, given $\mathcal{I}$ and a permutation π of $S \subseteq [n]$ and $i \in S$, $\varphi_\pi(i)$ can be computed in polynomial time. Setting $R = ||p||_\infty$, the reformulation into $\mathcal{I}'$ can be done in time polynomial in the size of $\mathcal{I}$. We now relate the profits of the optimal solutions to the two problems.

Following the construction of the sequencing problem, given a feasible chain $\mathcal{S} = (S_1, S_2, \ldots, S_T)$ (where we set $S_0 = \emptyset$), we construct the permutation π as follows:

- For each $t \in [T]$, let π^t be an arbitrary permutation of the items introduced in this period, $S_t \setminus S_{t-1}$.
- The permutation π is defined as the concatenation of $\pi^1, \ldots, \pi^T$ in this order. Namely, for $i \in S_t \setminus S_{t-1}$ with $t \in [T]$, we have $\pi(i) = \pi^t(i) + |S_{t-1}|$.

For every $i \in S_T$, let t_i denote the insertion time of item i with respect to the chain $\mathcal{S}$, where we recall from the proof of Lemma 2.1.1 that the insertion time of item i is defined as the time t

when an item is first introduced in the chain $\mathcal{S}$. To prove that $\Psi(\pi) \geq \Phi(\mathcal{S})$, it suffices to argue that $\varphi_\pi(i) \geq p_{i,t_i}$ for every item $i \in S_T$. To derive this relation, note that $C^i_\pi \subseteq S_{t_i}$ for all $i \in S_T$. Since $S_{t_i} \in \mathcal{F}_{t_i}$, we have $C^i_\pi \in \mathcal{F}_{t_i}$. Therefore, $\varphi_\pi(i) = \max\{p_{i,t} : t \in [T+1] \text{ and } C^i_\pi \in \mathcal{F}_t\} \geq p_{i,t_i}$.

Conversely, given a permutation π of any subset of items, we construct a chain $\mathcal{S} = (S_1, \ldots, S_T)$ that contains all items i such that $C^i_\pi \in \mathcal{F}_T$. Specifically, for $i \in \mathcal{F}_T$, let the insertion time $t_i = \text{argmax}\{p_{i,t} : t \in [T], C^i_\pi \in \mathcal{F}_t\}$. Hence, $i \in S_\tau$ for all $\tau \geq t_i$, and $\mathcal{S}$ is a chain. For $t \in [T]$, let $i_t = \text{argmax}\{\pi(i) : C^i_\pi \in \mathcal{F}_t\}$. To show that $\mathcal{S}$ is feasible, note that $S_t \subseteq C^{i_t}_\pi$ for all $t \in [T]$. Since $C^{i_t}_\pi \in \mathcal{F}_t$, we have $S_t \in \mathcal{F}_t$.

To show $\Phi(\mathcal{S}) = \Psi(\pi)$, it suffices to show $p_{i,t_i} = \varphi_\pi(i)$ for inserted items and $\varphi_\pi(i) = 0$ for non-inserted ones. To this end, note that our choice for the insertion time t_i mirrors exactly the definition of $\varphi_\pi(i)$, meaning that $p_{i,t_i} = \varphi_\pi(i)$. On the other hand, for any item i we do not insert into $\mathcal{S}$, one has $\varphi_\pi(i) = 0$, since $C^i_\pi \notin \mathcal{F}_T$. Observing that from an optimal solution of $\mathcal{I}'$ we can follow the construction above to obtain an optimal solution of $\mathcal{I}$, we conclude the proof for Point 2.

To see Point 3, consider a single-bin packing problem with a feasibility set $(\mathcal{F}_1, \ldots, \mathcal{F}_T)$, where item i has profit $p_{i,t}$ when inserted in time t. Let φ_π be the profit function of the single-machine sequencing problem to which the single-bin packing problem is mapped via the reduction above. For any $\lambda \in \mathbb{R}^n$, let $p'_{i,t} = \max\{0, p_{i,t} - \lambda_i\}$. We claim that the single-bin packing problem with feasibility set $\mathcal{F}$ and profit function p' is mapped via the construction from Point 2 to the single-machine sequencing problem with profit $\varphi'_\pi(i) = \max\{0, \varphi_\pi(i) - \lambda_i\}$, which would allow us to conclude Point 3. To see this, following the construction of Point 2,

$$\begin{aligned}
\varphi'_\pi(i) &= \max\{0, \max\{p_{i,t} - \lambda_i : t \in [T+1] \text{ and } C^i_\pi \in \mathcal{F}_t\}\} \\
&= \max\{0, \max\{p_{i,t} : t \in [T+1] \text{ and } C^i_\pi \in \mathcal{F}_t\} - \lambda_i\} \\
&= \max\{0, \varphi_\pi(i) - \lambda_i\}.
\end{aligned}$$

Now, we show Point 1. Following the reduction from Point 2, we create for each single-bin incremental packing problem $\mathcal{I}_j$ ($j \in [m]$) bin-underlying $\mathcal{I}$, a single-machine sequencing problem $\mathcal{I}'_j$. This allows us to define the multi-machine sequencing problem $\mathcal{I}'$ whose underlying single-

machine problems are $\mathcal{I}'_1, \ldots, \mathcal{I}'_m$. Since each single-machine sequencing problem reformulation can be done in time polynomial in the size of $\mathcal{I}_j$, this multi-machine sequencing problem reformulation can be done in time polynomial in the size $\mathcal{I}$. By Point 2, for $j \in [m]$ any feasible chain $\mathcal{S}_j$ for $\mathcal{I}_j$ can be mapped to a permutation π_j such that, if $S_{j,T}$ and $S_{j',T}$ are disjoint for all $j, j' \in [m]$, the domains of π_j and $\pi_{j'}$ are disjoint for all $j, j' \in [m]$ as well. This gives the feasibility of $\pi_1, \ldots, \pi_m$ in the multi-machine sequencing problem. Moreover, by Point 2, $\sum_{j\in[m]} \Psi_j(\pi_j) \geq \sum_{j\in[m]} \Phi_j(\mathcal{S}_j)$.

To show the reverse direction, let $\pi_1, \ldots, \pi_m$ be a feasible solution to the multi-machine sequencing problem. Again by Point 2, $\pi_1, \ldots, \pi_m$ can be mapped to feasible chains $\mathcal{S}_1, \ldots, \mathcal{S}_m$ such that $\mathcal{S}_j$ is feasible in the instance $\mathcal{I}_j$ and $\sum_{j\in[m]} \Phi_j(\mathcal{S}_j) = \sum_{j\in[m]} \Psi_j(\pi_j)$. Again, by Point 2, each set $S_{j,T}$ only contains items in the domain of π_j. Thus, if the domains of π_j and $\pi_{j'}$ are disjoint for all $j, j' \in [m]$, so are $S_{j,T}$ and $S_{j',T}$ for all $j, j' \in [m]$. Thus, $\mathcal{S}_1, \ldots, \mathcal{S}_m$ are feasible chains to the incremental packing problem. Note that from Point 2 that we can obtain optimal solutions to $\mathcal{I}'_j$ in polynomial time for each $j \in [m]$. Hence, we can obtain an optimal solution to $\mathcal{I}'$ in polynomial time.

E.2 Proof of Lemma 5.3.3

Let $\mathcal{I}$ be an instance of the incremental generalized assignment problem. Let $\mathcal{I}_1, \ldots, \mathcal{I}_m$ be the single-bin incremental packing problems bin-underlying $\mathcal{I}$. Note that, for $j \in [m]$, $\mathcal{I}_j$ is an instance of the generalized incremental knapsack problem.

Since the generalized incremental knapsack problem $\mathcal{I}_j$ is a single-bin incremental packing problem, by Point 2 of Lemma 5.3.2, it can be reformulated as a single-machine sequencing problem $\mathcal{I}'_j$. By Point 3 of Lemma 5.3.2, all sequencing instances obtained from generalized incremental knapsack problems with a fixed feasibility sets form a closed family. Hence, the family given by the union of all (infinitely many) such instances is closed.

Following the proof of Lemma 5.3.2, Point 1, we deduce that the multi-machine sequencing problem $\mathcal{I}'$ whose underlying m single-machine sequencing problems are $\mathcal{I}'_1, \ldots, \mathcal{I}'_m$ is the image

under the map from Lemma 5.3.2 of $\mathcal{I}$, concluding the proof.

E.3 Proof of Lemma 5.4.2

For $v \in \mathbb{Q}$, we can use the ellipsoid algorithm (see, for example, [38, Chapter 3]) with our β-approximate separation routine over the polyhedron defined by the dual constraints plus the constraint $\sum_{j\in[m]} q_j + \sum_{i\in[n]} \lambda_i \leq v$, to decide if there is a feasible dual solution $(\frac{q}{\beta}, \lambda)$ that satisfies $\sum_{j\in[m]} q_j + \sum_{i\in[n]} \lambda_i \leq v$, or if all feasible dual solutions (q, λ) satisfy $\sum_{j\in[m]} q_j + \sum_{i\in[n]} \lambda_i > v$. In the former case, we call v a *yes-value*, else we call it a *no-value*. If v is a no-value, by standard duality theory we deduce that the optimal solution to the primal have objective function has value at least v.

When our algorithm concludes that v is a yes-value, it does so by constructing a problem (MMS-DP') with the following properties: (MMS-DP') contains a subset of the constraints of (MMS-DP) of size polynomial in the size of the instance $\mathcal{I}$; there is a point $(\frac{q}{\beta}, \lambda)$ with $\sum_{j\in[m]} q_j + \sum_{i\in[n]} \lambda_i \leq v$ that is feasible for both (MMS-DP) and (MMS-DP'). By standard duality theory, the dual of (MMS-DP'), call it (MMS-P'), has all constraints of (MMS-LP) and a subset of its variables (the others being removed from the constraints they are in). Hence, we can find the optimal solution of (MMS-P') efficiently, and note that this can be completed to a feasible solution for (MMS-LP) by setting the missing variables to 0. Moreover, again using LP duality, the value of an optimal solution of (MMS-LP) and (MMS-P') is at least v and at most $\frac{v}{\beta}$.

By strong duality and the definition of single-machine scheduling problems, the optimal value of (MMS-DP) is a fractional number between 0 and Rn. Hence, we can run binary search to find the smallest value $v^* \in [0, Rn]$ for which our algorithm guarantees that v^* is a yes-value. If we let δ be the tolerance of the binary search, we conclude that the optimal solution of (MMS-LP) is at most $\frac{v^*-\delta}{\beta} \leq \frac{v^*}{\beta}$, and at least $v^* - \delta$, and a feasible solution for (MMS-LP) whose value is between those bounds can be obtained as explained above.

Appendix F: On the facets of the incremental knapsack polytope

F.1 Additional proofs from Section 6.3

F.1.1 Additional details from Example 6.3.3

We first show that (6.10) is valid. Suppose not, then there exists $\bar{x}$ feasible for (6.9) that violates (6.10). Notice that $\{2, 3, 4\}$ forms a cover in time 1. Therefore, we must have $\bar{x}_{2,1}+\bar{x}_{3,1}+\bar{x}_{4,1} \leq 2$. Thus, if (6.10) is violated, we must have $\bar{x}_{1,2} = \bar{x}_{5,2} = 1$. Without loss of generality, suppose $\bar{x}_{3,1} = \bar{x}_{4,1} = 1$. By constraints $\bar{x}_{i,1} \leq \bar{x}_{i,2}$, we have $\bar{x}_{3,2} = \bar{x}_{4,2} = 1$. Now, notice $w_1 + w_3 + w_4 + w_5 = 10 > 9$, violating the constraint $\sum_{i\in[n]} w_i\bar{x}_{i,2} \geq W_2$. Hence, (6.10) must be valid.

To see that (6.10) defines a facet, we give 10 linearly independent feasible solutions satisfying (6.10) at equality. In the matrix below, each row represents such a feasible solution. The variable each entry corresponds to is labelled in the top row. It is straightforward to verify that the solutions are linearly independent by verifying that the matrix below is full rank.

$$
\begin{array}{c}
\begin{array}{cccccccccc}
x_{1,1} & x_{1,2} & x_{2,1} & x_{2,2} & x_{3,1} & x_{3,2} & x_{4,1} & x_{4,2} & x_{5,1} & x_{5,2}
\end{array} \\
\begin{pmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0
\end{pmatrix}
\end{array}
$$

F.1.2 Proof of Claim 6.3.4

It is straightforward to verify that $x_{i,t} \leq x_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$. Next, we verify that for all $t \in [T]$, $\sum_{i\in[n]} w_i x_{i,t} \leq W_t$. For any $t < t'_{\hat{\imath}}$, since $x_{i,t} = 0$ for all $i \in [n]$, and the constraint is trivial. For any t such that $t'_{\hat{\imath}} \leq t < t_S$, only $x_{\hat{\imath},t} = 1$. By definition of $t'_{\hat{\imath}}$, we have $w_{\hat{\imath}} \leq W_{t'_{\hat{\imath}}} \leq W_t$. Finally, for any $t \geq t_S$, by construction $x_{j,t} = 1$ for $|S| - 1$ items in S. Therefore, for every $i \in S$,

$$\sum_{j\in S\setminus\{i\}} w_j \leq W_{t_S} \leq W_t,$$

where the first inequality follows from Condition C_1 of a strong cover. Hence, x is feasible.

As previously explained, $x_{j,t_S} = 1$ for $|S| - 1$ elements in S, therefore, (6.8) is satisfied at equality.

F.1.3 Proof of Claim 6.3.5

It is straightforward to verify that $x_{i,t} \leq x_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$. Next, we verify that for all $t \in [T]$, $\sum_{i\in[n]} w_i x_{i,t} \leq W_t$. For any $t < t_S$, we have $x_{i,t} = 0$ for all $i \in [n]$, and the

constraint is trivial. For $t = t_S < \hat{t}$, we have

$$\sum_{j \in [n]} w_j x_{j,t} = \sum_{j \in S \setminus \{\hat{\imath}\}} w_j \leq W_t$$

by Condition C_1 of a strong cover. For $t_S < t \leq T$,

$$\sum_{j \in [n]} w_j x_{j,t} \leq \sum_{j \in S} w_j \leq W_t$$

by the definition of t_S given in (6.6).

Finally, (6.8) is satisfied at equality since $x_{j,t_S} = 1$ for all $j \in S \setminus \{\hat{\imath}\}$.

F.1.4 Proof of Claim 6.3.6

It is straightforward to verify that $x_{i,t} \leq x_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$. Next, we verify that for all $t \in [T]$, $\sum_{i \in [n]} w_i x_{i,t} \leq W_t$. For $t < t'_{\hat{\imath}}$, $x_{i,t} = 0$ for all $i \in [n]$ and the constraint is trivial. For $t'_{\hat{\imath}} \leq t < t_S$, we have

$$\sum_{i \in [n]} w_i x_{i,t} \leq w_{\hat{\imath}} \leq W_{t'_{\hat{\imath}}} \leq W_t,$$

where the second inequality follows by definition of $t'_{\hat{\imath}}$.

For $t \geq t_S$, we have

$$\sum_{j \in [n]} w_j x_{j,t} \leq \sum_{j \in S \setminus \{j_1, j_2\}} w_j + w_{\hat{\imath}} \leq W_{t_S} \leq W_t,$$

where the second inequality follows from the assumption that

$$\sum_{j \in S \setminus \{j_1, j_2\}} w_j + w_{\max} \leq W_{t_S}.$$

Finally, (6.8) is satisfied at equality since $x_{j,t_S} = 1$ for $|S| - 2$ items in the cover and $x_{\hat{\imath},t_{\hat{\imath}}} = 1$.

F.1.5 Proof of Claim 6.3.7

It is straightforward to verify that $x_{i,t} \le x_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$. Next, we verify that for all $t \in [T]$, $\sum_{i\in[n]} w_i x_{i,t} \le W_t$. For $t < t_S$, we have $x_{i,t} = 0$ for all $i \in [n]$ and the constraint is trivial. For $t_S \le t < \hat{t}$, we have

$$\sum_{j\in[n]} w_j x_{j,t} = \sum_{j\in S\setminus\{j_1\}} w_j \le W_t,$$

by Condition C_1 of a strong cover. For $t \ge \hat{t} > t_{\hat{\imath}}$, we have

$$\sum_{j\in[n]} w_j x_{j,t} = \sum_{j\in S\setminus\{j_1\}} w_j + w_{\hat{\imath}} \le W_t,$$

by (6.7).

Finally, (6.8) is satisfied at equality since $x_{j,t_S} = 1$ for all $j \in S \setminus \{j_1\}$.

F.1.6 Proof of Claim 6.3.8

It is straightforward to verify that $x_{i,t} \le x_{i,t+1}$ for all $i \in [n]$ and $t \in [T-1]$. Next, we verify that for all $t \in [T]$, $\sum_{i\in[n]} w_i x_{i,t} \le W_t$. For $t < t'_{\hat{\imath}}$, we have $x_{i,t} = 0$ for all $i \in [n]$ and the constraint is trivial. For $t'_{\hat{\imath}} \le t < t_S$, we have

$$\sum_{i\in[n]} w_i x_{i,t} \le w_{\hat{\imath}} \le W_{t'_{\hat{\imath}}} \le W_t,$$

where the second inequality is by definition of $t'_{\hat{\imath}}$. For $t \ge t_S$, we have

$$\sum_{j\in[n]} w_j x_{j,t} \le \sum_{j\in S\setminus\{j_1\}} w_j + w_{\hat{\imath}} \le W_{t_S} \le W_t,$$

where the second inequality is by Condition C_2 of a strong cover.

Finally, (6.8) is satisfied at equality since $x_{j,t_S} = 1$ for all $j \in S \setminus \{j_1\}$.

F.2 Additional proofs from Section 6.4

F.2.1 Proof of Claim 6.4.3

Let y^* be a minimizer of (6.23) and let S^* be the associated cover. By definition, $E(S^*) \setminus S^* \subseteq \{k^*+1, \ldots, n\}$. Now suppose $E(S^*) \setminus S^* \neq \{k^*+1, \ldots, n\}$, that is, there exists $i < k^*$ with $i \notin S^*$ such that $w_i = w_{k^*}$. Let $y'_{i,t} = 1$ and $y'_{k^*,t} = 0$. For all other $j \notin \{i, k^*\}$, let $y'_{j,t} = y^*_{j,t}$ It is straight forward to check that the associated cover S' satisfies the following properties:

1. $S' = S^* \setminus \{k^*\} \cup \{i\}$;
2. $E(S') = E(S^*)$ and $E(S') \setminus S' = (E(S^*) \setminus S^*) \cup \{k^*\} \setminus \{i\}$;
3. for all $j \in E(S') = E(S^*)$, we have $t_{j,S'} = t_{j,S^*}$, where the definitions are given by (6.6) and (6.7). Due to this equivalence, for the rest of the proof, with a slight abuse of notation, we will simply denote them as t_j.

In what follows, we show that y' is also a minimizer of (6.23).

$$\begin{aligned}
\underbrace{\sum_{j=1}^{n}(1-\bar{x}_{j,t_j})y'_{j,t}}_{A} - \underbrace{\sum_{j\in E(S')\setminus S'}\bar{x}_{j,t_j}}_{B} &= \underbrace{\sum_{j=1}^{n}(1-\bar{x}_{j,t_j})y^*_{j,t} - (1-\bar{x}_{k^*,t_{k^*}}) + (1-\bar{x}_{i,t_i})}_{A} - \underbrace{\sum_{j\in E(S')\setminus S'}\bar{x}_{j,t_j}}_{B} \\
&= A - \underbrace{\left[\sum_{j\in E(S^*)\setminus S^*}\bar{x}_{j,t_j} + \bar{x}_{k^*,t_{k^*}} - \bar{x}_{i,t_i}\right]}_{B} \\
&= \sum_{j=1}^{n}(1-\bar{x}_{j,t_j})y^*_{j,t} - \sum_{j\in E(S^*)\setminus S^*}\bar{x}_{j,t_j},
\end{aligned}$$

where the first equality follows by definition of y', the second equality follows from Property 2 above, and the final equality follows by algebra.

We conclude that if y^* is a minimizer of (6.23), so is y'. Thus, we can repeat the above operation until no such i exists and $E(S^*) \setminus S^* = \{k^*+1, \ldots, n\}$, proving the claim.

F.2.2 Proof of Claim 6.4.4

First note that given any cover S and associated y, (6.23) evaluates to:

$$\sum_{j=1}^{n}(1-\bar{x}_{j,t_S})y_{j,t_S} - \sum_{j=E(S)\setminus S}\bar{x}_{j,t_j} = |S| - \sum_{j\in S}\bar{x}_{j,t_S} - \sum_{j=E(S)\setminus S}\bar{x}_{j,t_j} = |S| - \sum_{j\in E(S)}\bar{x}_{j,t_j}.$$

Using the equality above, since y^* minimizes (6.23), for any other S such that $|S| = |S^*|$, we have:

$$|S^*| - \sum_{j\in E(S^*)}\bar{x}_{j,t_j} \le |S| - \sum_{j\in E(S)}\bar{x}_{j,t_j} \Leftrightarrow \sum_{j\in E(S^*)}\bar{x}_{j,t_j} \ge \sum_{j\in E(S)}\bar{x}_{j,t_j}$$

Thus, if we have $(1-\epsilon)\sum_{j\in E(S^*)}\bar{x}_{j,t_j} \le |S^*| - 1$, we have

$$(1-\epsilon)\sum_{j\in E(S)}\bar{x}_{j,t_j} \le (1-\epsilon)\sum_{j\in E(S^*)}\bar{x}_{j,t_j} \le |S^*| - 1.$$

The claim follows by noting $|S| = |S^*|$.

F.2.3 Proof of Claim 6.4.5

First note that by definition

$$\sum_{j=1}^{k^*}\tilde{p}_{j,t^*}\bar{y}_{j,t^*} = \sum_{j=1}^{k^*}\tilde{p}_{j,t^*}y^*_{j,t^*} = q^*. \tag{F.1}$$

Since $\tilde{p}_{j,t^*} = \lfloor\frac{\bar{p}_{j,t^*}}{c_{i^*}}\rfloor$, it follows that

$$c_{i^*}\cdot\tilde{p}_{j,t^*} \le \bar{p}_{j,t^*} \le c_{i^*}\cdot(\tilde{p}_{j,t^*}+1) \tag{F.2}$$

Hence, we deduce,

$$\begin{aligned}
\sum_{j=1}^{k^*} \bar{p}_{j,t^*} \bar{y}_{j,t^*} &\leq c_{i^*} \sum_{j=1}^{k^*} (\tilde{p}_{j,t^*} + 1) \bar{y}_{j,t^*} \\
&= c_{i^*} \sum_{j=1}^{k^*} \tilde{p}_{j,t^*} y^*_{j,t^*} + c_{i^*} \sum_{j=1}^{k^*} \bar{y}_{j,t^*} \\
&\leq \sum_{j=1}^{k^*} \bar{p}_{j,t^*} y^*_{j,t^*} + c_{i^*} \sum_{j=1}^{k^*} \bar{y}_{j,t^*} \\
&\leq \sum_{j=1}^{k^*} \bar{p}_{j,t^*} y^*_{j,t^*} + c_{i^*} \cdot i^*, \qquad \text{(F.3)}
\end{aligned}$$

where the first inequality follows from the upper bound of $\bar{p}_{j,t}$ given in (F.2), the first equality follows from (F.1), and the second inequality follows from the lower bound of $\bar{p}_{j,t}$ given in (F.2). The final inequality uses that $\bar{y}_{j,t^*} = 0$ for all $v^t(j) > i^*$, which implies $\sum_{j=1}^{k^*} \bar{y}_{j,t^*} \leq i^*$.

Now, by recalling that $c_{i^*} = \frac{\tilde{\epsilon} \bar{p}_{v^{t^*}_{i^*}, t^*}}{i^*}$, we conclude

$$\begin{aligned}
\sum_{j=1}^{k^*} \bar{p}_{j,t^*} y^*_{j,t^*} + c_{i^*} \cdot i^* &= \sum_{j=1}^{k^*} \bar{p}_{j,t^*} y^*_{j,t^*} + \tilde{\epsilon} \bar{p}_{v^{t^*}_{i^*}, t*} \\
&\leq (1 + \tilde{\epsilon}) \sum_{j=1}^{k^*} \bar{p}_{j,t^*} y^*_{j,t^*}, \qquad \text{(F.4)}
\end{aligned}$$

where the equality is by definition of c_{i^*}, and the inequality is by noting $y^*_{v^{t^*}_{i^*}, t^*} = 1$ by definition of i^*. Combining (F.3) and (F.4) concludes the proof.

F.2.4 Proof of Claim 6.4.6

Plugging in $\tilde{\epsilon} = \frac{\epsilon}{n}$, we have

$$1 \leq (1 + \frac{\epsilon}{n})(|S^*| - \sum_{j \in S^*} \bar{x}_{j,t^*}) - \sum_{j \in E(S^*) \setminus S^*} \bar{x}_{j,t_j}.$$

Equivalently, by rearranging the above inequality, we obtain

$$\begin{aligned}(1+\frac{\epsilon}{n})\sum_{j\in S^*}\bar{x}_{j,t_{S^*}}+\sum_{j\in E(S^*)\setminus S^*}\bar{x}_{j,t_j} &\leq (1+\frac{\epsilon}{n})|S^*|-1\\ &\leq |S^*|+\epsilon-1, \end{aligned} \tag{F.5}$$

where the second inequality follows since $|S^*| \leq n$.

Now, we consider two cases:

1. $$-\left(\frac{\epsilon(1+n)}{n}\sum_{j\in S^*}\bar{x}_{j,t_{S^*}}+\epsilon\sum_{j\in E(S^*)\setminus S^*}\bar{x}_{j,t_j}\right)\leq -\epsilon.$$

   In this case, we can simply add the inequality to (F.5) and conclude the claim.

2. $$-\left(\frac{\epsilon(1+n)}{n}\sum_{j\in S^*}\bar{x}_{j,t_{S^*}}+\epsilon\sum_{j\in E(S^*)\setminus S^*}\bar{x}_{j,t_j}\right)> -\epsilon.$$

   Rewriting the above inequliaty, we obtain

   $$\frac{1}{n}\sum_{j\in S^*}\bar{x}_{j,t_{S^*}}+\sum_{j\in S^*}\bar{x}_{j,t_{S^*}}+\sum_{j\in E(S^*)\setminus S^*}\bar{x}_{j,t_j}<1.$$

   Therefore,

   $$\sum_{j\in S^*}\bar{x}_{j,t_{S^*}}+\sum_{j\in E(S^*)\setminus S^*}\bar{x}_{j,t_j}<1\leq |S^*|-1,$$

   where the final inequality follows from the assumption that $|S^*| \geq 2$.