Discrete Optimization Problems in Popular Matchings and Scheduling

Vladlena Powers

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

COLUMBIA UNIVERSITY

2020

ABSTRACT

Discrete Optimization Problems in Popular Matchings and Scheduling

Vladlena Powers

This thesis focuses on two central classes of problems in discrete optimization: matching and scheduling.

Matching problems lie at the intersection of different areas of mathematics, computer science, and economics. In two-sided markets, Gale-Shapley's model has been widely used and generalized to assign, e.g., students to schools and interns to hospitals. The goal is to find a matching that respects a certain concept of fairness called *stability*. This model has been generalized in many ways. Relaxing the stability condition to *popularity* allows to overcome one of the main drawbacks of stable matchings: the fact that two individuals (a *blocking pair*) can prevent the matching from being much larger. The first part of this thesis is devoted to understanding the complexity of various problems around popular matchings. Chapter 2 and Chapter 3 deal with variations of the *maximum weighted popular matching problem*. In particular, we show various NP-hardness results, while on the other hand showing that a popular matching of maximum weight (if any) can be found in polynomial time if the input graph has bounded treewidth. Chapter 4 investigates algorithmic questions on the relationship between popular, stable, and Pareto-optimal matchings.

The second part of the thesis deals with a combinatorial scheduling problem arising in cyber-security. Moving target defense strategies allow to mitigate cyber attacks. We analyze a strategic game, PLADD, which is an abstract model for these strategies. We provide an IP formulation, show it has unbounded integrality gap, and write a software that solves the problem via enumeration and visualizes the optimal solution. This is investigated in Chapter 5.

Contents

List of Figures

List of Tables

Acknowledgements

My Ph.D. journey has been a road full of excitement, challenges and discovery. It has been a road of search for continuous improvement and to prove to myself my capabilities and their usefulness to solve problems. First and foremost, I want to thank my advisor, Yuri, who has been my guide in this journey. I always looked up to him for his high standard of quality in everything and unlimited enthusiasm for advancing the research. His passion to research and teaching continuously spreads to other students, including me.

I want to thank my husband Nick. This thesis would not have been possible without him. Not only did he provide me emotional support but also his tremendous talent and genius in software development helped me to develop programs that I used continuously through my research to test ideas, find counterexamples and develop intuition to problems through experiments that are the foundation to my findings. On the third year of my Ph.D. program our daughter Scarlett was born. She is the biggest motivation and inspiration for me to be a better person for tomorrow both in research and personal development.

I would like to thank my mother Tetiana who prioritized my interests and became a mother to my daughter while I was working on my research.

I would like to give a special thank you to my friend Goutam for his continuous support and help.

To my daughter, husband and mother, for everything.

Introduction

This thesis deals with matching and scheduling problems.

The matching problems we will consider go under the generic name of *Matching under preferences*. Together with the input graph (often, but not always, bipartite), we are given, for each node, a strictly ordered list of its neighbors. The goal is to output a matching of the graph that satisfies additional properties. The most classical one, introduced by Gale and Shapley in [18], is *stability*. Recent literature has focused on properties other than stability. The most important for this thesis will be *popularity*, which generalizes stability, in the sense that every stable matching is also popular. While the complexity of many algorithmic problems on the class of stable matchings is well-known, much less was known on the classes of popular matchings. From a discrete optimization point of view, typical questions one would ask are: given a bipartite graph with strict preference lists and nonnegative profits, can one find efficiently a popular matching of maximum profit? What if we are given nonnegative costs, and one wants to find a popular matching of minimum cost? These questions has been asked multiple times in the literature [12, 13, 28, 33, 41]. As our first results, we settle them in Chapter 2, where we show tight hardness results for those and related problems. Interestingly, all the hardness results from Chapter 2 can be seen as a corollary of an easy-to-state problem on *stable matchings*, also introduced in this thesis. These techniques also allows us to give short proofs for other results on popular matchings that have recently appeared in the literature.

When a problem is hard, it is natural to add restrictions on the input, and investi-

gate whether the complexity of the problem changes. For instance, many notoriously hard problems become solvable in polynomial time when the input graph is assumed to have bounded *treewidth*. In Chapter 3, we show this is the case for (extensions of) the max profit / min cost popular matchings problems mentioned above. This also applies in the roommate case, i.e., when the input graph is not bipartite, for which in general graphs even deciding if a popular matching exists is NP-Complete. Results from Chapter 2 and Chapter 3 have appeared in [14, 16, 17].

Stability and popularity are only two among many concepts introduced in the literature. Another important concept is that of *Pareto-optimality*. While popularity generalizes stability, Pareto-optimality and stability are often seen as in opposition to one another. Of course, a matching that is both Pareto-optimal and stable (or popular) is highly sought after, because it satisfies desirable, seemingly contradicting, properties. So a natural question is: can we find a matching that is both stable (resp. popular) and Pareto-optimal, if any? And if there is none, can we find a matching that satisfies one property and is "not too far away" (in some well-defined sense) from the other? These and related questions are investigated in Chapter 4. This is a joint project with Yuri Faenza.

In Chapter 5 we investigate a new combinatorial scheduling problem arising from modeling moving target defense strategies in cyber-security. We analyze a strategic game, PLADD, which is an abstract model for these strategies and provide an IP formulation, show it has unbounded integrality gap, and write a software that solves the problem via enumeration and visualizes the optimal solution. This is a joint project with Ojas Parekh, Cynthia Phillips, Nouri Sakr and Cliff Stein.

We start this thesis by settling notation and introducing many useful concepts in Chapter 1.

Chapter 1

Preliminaries

1.1 Notation

As it is customary, we denote by $\mathbb{N} = \{1, 2, \ldots\}$ the set of natural numbers, by $\mathbb{Q}$ the set of rational numbers, and by $\mathbb{R}$ the set of real numbers. For $S \subseteq \mathbb{R}$, S_+ denotes the intersection of S with the set of nonnegative reals, while we write $S_0 := S \cup \{0\}$. For $k \in \mathbb{N}$, we let $[k] := \{1, 2, \ldots, k\}$. For a set T, $w : T \to \mathbb{R}$, and $S \subseteq T$, we write $w(S) := \sum_{e \in S} w(e)$. For a graph G, we denote by $V(G)$ its vertices and by $E(G)$ its edges.

Given a graph $G(V, E)$, a *matching* is a collection M of edges from E such that each node of V has degree at most one in M. A node is *covered* by a matching M if its degree in M is exactly one. Given a matching M and $v \in V$, we denote by $V(M)$ the set of nodes covered by M and by $M(v)$ the unique node u such that $uv \in M$ if it exists, and $\perp$ otherwise. Given a subgraph H of G and a matching M in G, we denote by M_H the matching induced by M in H, i.e., $M_H := M \setminus E(H)$. We will often work with bipartite graphs $G(A \cup B, E)$. In this case, given a matching M of G, we write $A(M) := A \setminus V(M)$ and similarly $B(M) := B \setminus V(M)$.

All paths and cycles in this thesis are simple, hence they are identified by a sequence of nodes $P = v_1, v_2, \ldots, v_k$. The *inverse* of P is the path $v_k, v_{k-1}, \ldots, v_1$. For a node v of a graph G, we denote by $\Gamma(v)$ the neighborhood of v.

1.2 Sets, posets, lattices

All sets from this thesis are assumed to be finite. Given two sets $A; B$, we let $A \quad B$ be their symmetric difference, i.e.,

$$A \quad B := (A \, n \, B) \; [\; (B \, n \, A):$$

A *partial order* on a set S is a reflexive, anti-symmetric and transitive binary relation on elements of S. A set together with a partial order, , is called partially ordered set (or, *poset*), and denoted by $(S; \quad)$. Given elements $a; b; c$ from a poset, we say that c is a *lower bound* (resp. upper bound) to $a; b$ if $c \quad a; b$ (resp. $c \quad a; b$). A *lattice* is a poset such that each pair of elements a and b has a unique *least* upper bound, denoted by $a \_ b$, and a unique *greatest* lower bound, denoted by $a \wedge b$. $a \_ b$ is called a least upper bound if there is no element c such that $a \quad c$, $b \quad c$ and $c \quad a \_ b$. Similarly, $a \wedge b$ is called a greatest lower bound if there is no element c such that $c \quad a$, $c \quad b$ and $a \wedge b \quad c$. A *sublattice* of a lattice $(S; \quad)$ is given by a pair $(S^0; \quad ^0)$ with $S^0 \quad S$ and ^{0} the restriction of to S^0, with the property that, for each $a; b \; 2 \; S$, we have that $a; b \; 2 \; S^0$ implies $a \_ b; a \wedge b \; 2 \; S^0$.

1.3 Matching under preferences

In the *marriage model* we are given a bipartite graph $G(A \; [\; B; E)$. We usually refer to A as the set of *men* and to B as the set of *women*. We are also given in input, for each node v of the graph (or *agent*), a strict *ranking* of its neighbors. We denote this ranking by $>_v$, with the convention that v prefers u to z if $u >_v z$. We use numerical labels for preferences, with the convention that between two numbers $i < j$, the smaller number i represents the more preferred choice. Hence, label 1 indicates the top choice. A *marriage instance* is therefore represented by the pair $(G; <)$ with

$<= f <_v g_{v2A[B}$. The input of the *roommate model* is still a pair $(G; <)$ as above, with the difference that the input graph is not required to be bipartite. We will also call an instance of the roommate model a *preference system*. Many problems in this thesis deals with finding matchings in instances of the marriage or roommate model with special properties. We next present basic concepts and facts that will be useful throughout the thesis, and unless otherwise noted, can be found in [22].

Stability

In the academic community, the marriage model came to prominence through the seminal work by Gale and Shapley [18], who introduced the concept of *stability*.

Consider a marriage instance $(G; <)$ with a matching M. An edge of graph G between nodes u and v will be denoted as $(u; v)$ or uv. We call an edge *blocking* or a *blocking pair* if the endpoints strictly prefer each other to their assignments in M. If M does not have blocking pairs, then M is *stable*. In [18] the authors showed that in the marriage model a stable matching always exists and can be found in linear time by the following algorithm:

Algorithm 1: Gale-Shapley

```
start from all men and women being unmatched
while there is a man m (a node from set A) that is unmatched and has a
 neighboring woman (a node from set B) that he did not propose to do
    w := first woman in m's list to whom m has not proposed yet:
    if w unmatched then
        assign m and w to be matched to each other
    end
    else
        if w prefers m to its match m' then
            assign m and w to be matched and m' be unmatched
        end
        else
            w rejects m
        end
    end
end
output formed pairs
```

We now know [46] that the problem was formulated and solved in practice in the US labor market for medical interns and residents well before the paper by Gale and Shapley. Nowadays, matching markets are used to model many different frameworks, from online dating [25], to house and school choice [2], to the assignment of users to servers in large networks [40].

We now present some basic facts about stable matchings. Some will apply to both roommate and marriage instances, while others only to the marriage case. We denote be $S(G; \succ)$ the set of stable matchings of $(G; \succ)$. We omit the dependency on $(G; \succ)$ when clear from the context. The first useful fact is that all stable matchings match the same vertices. In particular, they all have the same cardinality.

Theorem 1.3.1. *Let $(G; \succ)$ be a preference system and $M; M' \in S(G; \succ)$. Then $V(M) = V(M')$.*

Consider the following partial order over set of matchings:

$$M \preceq M' \quad , \quad M(m) \preceq_m M'(m) \text{ for all } m \in A:$$

When $(G; \succ)$ is a marriage instance, $(S; \preceq)$ is a lattice. In particular, if $M; M' \in S$, then $M^{\wedge}; M^{\vee} \in S$, where for each man $m \in A$, $M^{\wedge}(m)$ (resp. $M^{\vee}$) is set to be the most (resp. least) favourite partner of m between $M(m)$ and $M'(m)$. This implies that, among all stable matchings, there is one that all men (weakly) prefer to any other stable matching. It is called *man-optimal* matching. Similarly, there is a stable matching that all women prefer to any other stable matching: we call it *woman-optimal*.

Theorem 1.3.2. *Let $(G; \succ)$ be a marriage instance. Gale-Shapley algorithm, when men propose, results in the man-optimal stable matching. Similarly, Gale-Shapley algorithm, when women propose, results in the woman-optimal stable matching.*

The lattice structure can be exploited for developing fast algorithms for a number of problems related to stable matchings. In this thesis, we will be mostly interested in its consequences for edge-weighted stable matching problems.

Problem 1.3.1. *[weighted stable matching problem]* **Given:** *A marriage instance* $(G; <)$*, and a function* $w : E \to \mathbb{R}$*.* ***Find:*** *a stable matching* M *that maximizes* $w(M)$*.*

Theorem 1.3.3. *The weighted stable matching problem can be solved in polynomial time. Moreover, the convex hull* $P(G; <)$ *of the characteristic vectors of stable matchings in a marriage instance* $(G; <)$ *can be described via a polynomial number of constraints.*

Interestingly, the lattice structure is preserved if we restrict to the subset of stable matching containing and/or avoiding certain subsets of edges.

Lemma 1.3.1. *Let* $(G; <)$ *be an instance of the marriage problem, and let* $E^{in}; E^{out} \subseteq E(G)$*. Define*

$$S(E^{in}; E^{out}) := \{S \in \mathcal{S} : E^{in} \subseteq S \text{ and } E^{out} \cap S = \emptyset\}.$$

Then $(S(E^{in}; E^{out}); \preceq)$ *is a sublattice of* S*. In particular, there exists a man-optimal stable matching in* $(S(E^{in}; E^{out}); \preceq)$*.*

Proof. Immediately from the fact that if $M, M' \in S(E^{\text{in}}; E^{\text{out}})$, then $M \wedge M', M \vee M' \in S(E^{\text{in}}; E^{\text{out}})$. □

The polysize description of $P(G; <)$ from Theorem 1.3.3 immediately implies the following.

Corollary 1.3.1. *Given a marriage instance $(G; <)$, a function $w : E \to \mathbb{R}$, sets $E^{in}; E^{out} \subseteq E(G)$, a matching of $(S(E^{in}; E^{out}); \ )$ of maximum weight can be found in polynomial time.*

Popularity

In 1975 in [19], Gärdenfors introduced the notion of popularity and showed that it is a generalization of stability. Recall that every stable matching has the same size, see Theorem 1.3.1, and there could be just two agents who are blocking a matching. This is a very restricting property. In popular matchings, on the other hand, the input from all agents on a given matching is taken into consideration.

Consider a preference system $(G(V; E); <)$. We say a vertex $v \in V$ *prefers* matching M to matching M' if either v is matched in M and unmatched in M', or v is matched in $M; M'$ and strictly prefers its partner in M to its partner in M'. Let $\phi(M; M')$ be the number of vertices that prefer M to M'. A matching M is *more popular* than M', if $\phi(M; M') > \phi(M'; M)$. A matching M is *popular* if there is no matching M' that is more popular than M, meaning $\phi(M; M') \geq \phi(M'; M)$ for all matchings M' in $(G; <)$. The following fact has been shown in [27].

Lemma 1.3.2. *Let S (resp. M) be a stable (resp. popular) matching of a preference system $(G; <)$. Then $V(S) \subseteq V(M)$. In particular, stable matchings are popular matchings of minimum size.*

Relaxing stability to popularity allows therefore to increase the size of matchings, while still preserving a notion of global stability. However, given a matching, checking if it is popular is not immediate, since we would have to compare it against all other matchings of the graph. However, this can be still performed in polynomial time, see [7, 27].

Theorem 1.3.4. *Given a preference system $(G; <)$ and a matching M of G, whether M is popular can be decided in polynomial time.*

One of the algorithms for efficiently recognizing whether a matching is popular is based on the following combinatorial characterization of popular matchings. Fix a matching M of $(G; <)$. An M-alternating path (resp. cycle) in G is a path (resp. cycle) whose edges are alternatively in M and in $E(G) \setminus M$. We associate labels to edges from $E \setminus M$ as follows: an edge uv is $(\ ;\ )$ if both u and v prefer their respective partners in M to each other; $uv = (+;+)$ if u and v prefer each other to their partners in M; $uv = (+;\ )$ if u prefers v to its partner in M and v prefers its partner in M to u. To denote labels of edge uv, sometimes we write *uv is a* $(+;+)$ *edge*, and similarly for the other cases. If we want to stress that u prefers his current partner to the other endpoint of the edge, we write that e.g. uv is a $(\ ;+)$ edge. The graph G_M is defined as the subgraph of G obtained by deleting edges that are labeled $(\ ;\ )$. Observe that M is also a matching of G_M, hence definitions of M-alternating path and cycles apply in G_M as well. These definitions can be used to obtain a characterization of popular matchings in terms of forbidden substructures of G_M [27]. For a path (resp. cycle) P and an edge e, we say that *P contains e* if $e = uv$ for two nodes $u; v$ that are consecutive in P (not necessarily in this order).

Theorem 1.3.5. *Let $(G; <)$ be a preference system, and M be a matching of G. Then M is popular if and only if G_M does not contain any of the following:*

1. *an M-alternating cycle that contains a $(+;+)$ edge.*
2. *an M-alternating path that contains two distinct $(+;+)$ edges.*
3. *an M-alternating path starting from an M-exposed node that contains a $(+;+)$ edge.*

If the matching M under consideration is clear, we omit mentioning that labels are wrt M, and paths, cycles etc. occur in the graph G_M: we simply say e.g. that uv

is a $(+;+)$ edge and P is an M-alternating path with a $(+;+)$ edge. In particular, unless stated otherwise, when talking about M-alternating paths or cycles, we always mean in the graph G_M (hence, after $(\ ;\ )$ edges have been removed).

We say a matching M *defeats* a matching M' (and M' is defeated by M) if either of these two conditions holds:

(i) M is more popular than M', i.e., $(M;M') > (M';M)$;

(ii) $(M;M') = (M';M)$ and $|M| > |M'|$.

A *dominant* matching is defined to be a matching that is never defeated. A dominant matching is by definition a popular matching of maximum size.

A maximum size popular matching can be twice of the size of a stable matching in the same instance. In [32], Kavitha shows how to find a maximum size popular matching efficiently in marriage instances using a combination of the Gale-Shapley algorithm and promotion of nodes that were rejected by all adjacent nodes. In fact, her algorithm finds a dominant matching.

The following characterization of dominant matchings appeared in [13], where it is also shown that a dominant matching always exists, and all dominant matchings are of the same size (a proof of this follows by Lemma 1.3.3 below).

Theorem 1.3.6. *Let M be a popular matching in a preference system $(G;\prec)$. M is dominant if and only if there is no M-augmenting path in G_M.*

The following fact is attributed in [12] to [24]. We give a proof here for completeness.

Lemma 1.3.3. *Let $(G;\prec)$ be a preference system, and M (resp. S, D) be a popular matching (resp. a popular matching of minimum size, a popular matching of maximum size) in $(G;\prec)$. Then $V(S) \quad V(M) \quad V(D)$.*

Proof. The fact that stable matchings are popular matchings of minimum size and that $V(S)$ $V(M)$ whenever S is stable and M is popular follows from Lemma 1.3.2. Hence, for any popular matching S' of minimum size, we have $V(S')$ $V(S)$ and $|S'| = |S|$, which implies $V(S') = V(S)$ $V(M)$ for any popular matching M, settling the first part. Now suppose that M again is popular, D is a dominant matching, and there exists $v \in V(M) \cap V(D)$. Then, $G(M \triangle D)$ contains an alternating path P starting at v with an edge of M. Since both matchings are popular, the number of nodes that prefer M to D in P are exactly the same number of nodes that prefer D to M in P. In particular, P has an even number of nodes and is therefore D-augmenting in G. Now label edges of $P \cap D$ with $+;$ as discussed at the beginning of the section. Note that the number of $(+;+)$ edges and $(\ ; \ )$ edges in P coincide, otherwise either D or M is not popular. We consider three cases: if there is no $(+;+)$ edge, hence no $(\ ; \ )$, the path is D-augmenting in G_D, contradicting the fact that D is dominant. Now suppose there is a $(+;+)$ edge e. Since the number of $(+;+)$ and $(\ ; \ )$ edges in P coincide, one of the following two facts happen: either we can assume wlog that e is traversed in P before any $(\ ; \ )$ edge, or there is a subpath of P that contains two $(+;+)$ edges and no $(\ ; \ )$ edge. In both circumstances, we contradict Theorem 1.3.5: condition (ii) in the second case, and condition (iii) in the first. Hence $V(M)$ $V(D)$. If M' is a popular matching of maximum size, we conclude $V(M') = V(D)$, as required. □

Similarly to what was done for stable matching, in the bipartite case we can define (and solve efficiently) the problem of computing a dominant matching of maximum weight.

Problem 1.3.2. *[weighted dominant matching problem]* **Given:** *A marriage instance $(G; <)$, and a function $w : E \to \mathbb{R}$.* ***Find:*** *a dominant matching M that maximizes $w(M)$.*

Theorem 1.3.7. *[13] Problem 1.3.2 can be solved in polynomial time for marriage instances. Moreover, the convex hull $D(G;\prec)$ of the characteristic vectors of dominant matchings in a marriage instance $(G;\prec)$ has a linear extension[1] that can be described via a polynomial number of constraints.*

Given $(G;\prec)$, we often represent $\prec$ via a *preference pattern*, i.e., a function L that maps ordered pairs $(u;v)$ with $uv \in E$ to the position of v in $\prec_u$. If $L(u;v) = 1$, we say that v is the *favorite partner* of u. Sometimes it will makes sense to specify a partially described preference pattern for a subgraph of the input graph. More formally, an *incomplete preference pattern* I for a bipartite graph G is a map $I : \{(u;v) : uv \in E(G)\} \to \mathbb{N} \cup \{1;;\}$, such that $I(u;v) = I(u;x)$ for some $u \neq x \neq v \in V$ implies $I(u;v) = I(u;x) = ;$, and $I(u;v) \in \mathbb{N}$ implies $I(u;v) \leq |\Gamma(u)|$. We say that a preference pattern L for a supergraph G' of G *agrees* with an incomplete preference pattern I if, for all $uv \in E(G)$ with $I(u;v) \neq ;$, we have: $L(u;v) = I(u;v)$ whenever $I(u;v) \in \mathbb{N}$ and $L(u;v) = |\Gamma_{G'}(u)|$ whenever $I(u;v) = 1$ (here $\Gamma_{G'}(u)$ is the neighborhood of u in G').

On possible applications for popular matchings

Stability has proven to be a powerful concept with a rich mathematical structure used in numerous real-world applications in matching markets, ranging from house allocation, to assignment of students to schools, users to servers, medical students to hospitals, refugees to cities. In [41] authors give an overview of the topic and mention many more applications such as kidney exchanges with incompatible patient-donor pairs [47, 48, 5], auction mechanisms for sponsored search [6], assigning children to daycare [36], placement of graduating rabbis [8], supply-chain networks [43], online dating in the US [26], and online matrimony in India [31], etc. However, it is known

[1] A linear extension of a polytope P is a polytope Q that linearly projects to P.

that all stable matchings have the same size, see Theorem 1.3.1, and the set of matched vertices is the same. In certain cases it is strongly preferred to obtain a matching of larger size (e.g. more people are matched or assigned) with the sacrifice of having some blocking pairs but still with the global notion of stability.

Popular matchings are the potential substitute for more restrictive stable matching in many of the above applications. For example, one potential practical use of popular matching is online dating. In [26] authors studied data from an online dating service and made predictions using the Gale-Shapley algorithm. The resulted matchings are similar to matchings formed by the dating site. The preferences are estimated from user's information and interactions on the website. In a new study [45] sociologists found that the most popular way heterosexual couples meet is online. If more people become users of online dating websites every year, then it is crucial to develop algorithms that would result in matchings leading to happy married life for as many people as possible. Since stable matchings are proven to underlie matchings on one of the major dating websites [26], then potential application of popular matchings could lead even to more optimistic results, where the number of people matched is larger.

Pareto-optimality

Popular matching provide an opportunity to overcome drawbacks of stability. There is another class of matchings, *Pareto-optimal*, that also relax stability by ignoring the preferences of one side. They are proven to be useful in numerous applications, such as assignment students to housings. They are also well-studied in economics. Consider a marriage instance $(G(A \cup B); \prec)$. A matching M *Pareto beats* a matching M' of $(G; \prec)$ if there exists a man which prefers M to M' and no man prefers M' to M. A matching M is *Pareto-optimal* if there is no matching M' which Pareto beats M. Popularity provides global stability by ensuring overall satisfaction, while

Pareto-optimally maximizes the satisfaction of one side.

A matching M is maximal if there is no edge $mw \in E(G)$ such that m and w are both unmatched in M. M is *trade-in-free* if there is no edge mw, $m \in A; w \in B$ such that woman w is unmatched and man m prefers w to its partner $M(m)$. An M-alternating cycle C is called a *coalition cycle* of M, if for each edge $mw \in E(C) \cap M$, $m \in A; w \in B$, m prefers w to $M(m)$. If M does not contain coalition cycles, then it is called *coalition-free*.

The following Lemma provides a characterization of Pareto-optimal matchings [4].

Lemma 1.3.4. *Let $(G; <)$ be a preference system, and M be a matching of G. Then M is Pareto-optimal if and only if M is maximal, trade-in-free, and coalition-free.*

Theorem 1.3.8 provides a characterization of popular Pareto-optimal matchings.

Theorem 1.3.8. *Let $(G; <)$ be a preference system, and M be a popular matching of G. Then M is Pareto-optimal if and only if M is coalition-free.*

Proof. Let M be popular and Pareto-optimal. From Lemma 1.3.4 we conclude that M is coalition-free.

Now let M be popular and coalition-free. We prove that M is maximal and trade-in-free. Assume by contradiction that M is not maximal, and there is an edge mw with both m and w being unmatched. In this case mw is an M-alternating path in G_M from an unmatched node m to $mw = (+;+)$. This contradicts popularity (see Lemma 1.3.5). Assume by contradiction that M is not trade-in-free, then there is an unmatched women w and m such that m prefers w to $m_M(w)$. Similarly, mw is an M-alternating path in G_M from an unmatched node w to $mw = (+;+)$. This is a contradiction to popularity, see again Lemma 1.3.5. □

In [49] the following matching mechanism *serial dictatorship* was introduced.

Algorithm 2: Serial dictatorship

start from all men and women being unmatched
mark all women "unmatched", and let A be the set of men
while A *is not empty* **do**
  choose any man m from A
  w := top choice for man m among women with mark "unmatched"
  assign w to m
  remove m from A
  change w's mark from "unmatched" to "matched"
end
output formed pairs

In [1] it was shown the following characterization of Pareto-optimality.

Theorem 1.3.9. *Let* $(G; <)$ *be a preference system, and* M *be a matching of* G. *Then* M *is Pareto-optimal if and only if* M *is obtained from the serial dictatorship mechanism.*

1.4 SAT and hardness of approximation

A boolean formula in CNF (*conjunctive normal form*) is a conjuction of clauses such that each clause is a disjunction of literals. Each literal can be positive or affirmative (denoted by x_i), or negative or negated (denoted by $\bar{x}_i$). Given a CNF boolean formula, we define an *assignment* to be a mapping from literals to $\{0; 1\}$, and as usual we say that a literal that maps to 0 is set to *false*, while a literal that maps to 1 is set to *true*. We say an assignment is *feasible* if in each clause, there is at least one literal set to true. We say that an assignment is *consistent* if a literal corresponding to x_i is set to true if and only if all literals corresponding to x_i are set to true and all literals corresponding to $\bar{x}_i$ are set to false. Note that consistent assignments are in bijection with maps from the set of *variables* to $\{0; 1\}$ (assuming wlog that each variable appears at least once). Hence, we often refer to a consistent assignment of literals as an *assignment of variables*. An assignment of literals that is both consistent and feasible is called *satisfying*.

A *Boolean satisfiability problem* is the following decision problem.

Problem 1.4.1. *[SAT]* **Given:** *A boolean formula in CNF.* ***Decide:*** *If there exists a satisfying assignment to .*

A SAT instance is *satisfiable* if it has a satisfying assignment. For $2 (0; 1]$, we say that a SAT instance is -satisfiable if there exists an assignment of variables that satisfies at least a fraction of the clauses of . The following special case of SAT is well-studied.

Problem 1.4.2. *[3-SAT]* **Given:** *A boolean formula in CNF, where each clause has at most 3 literals.* ***Decide:*** *If there exists a satisfying assignment to .*

Indeed, the following result is known to hold.

Theorem 1.4.1. *[23] Let " > 0 be a constant. Unless P=NP, there is no polynomial-time algorithm that, given an instance of 3-SAT, accepts if is satisfiable, rejects if is not $(7=8 + ")$-satisfiable, and is free to accept or reject if none of these cases apply.*

Monotone 3-SAT (called NM-E3SAT in [21]) is the version of 3-SAT where each clause consists of only positive or only negative literals.

Problem 1.4.3. *[Monotone 3-SAT]* ***Given:*** *A boolean formula in CNF where each clause contains at least 2 and at most 3 variables, either all negated or all positive.* ***Decide:*** *If there exists a satisfying assignment to .*

Monotone 3-SAT is also known to be NP-hard, since as observed in [11], a reduction from an instance of 3-SAT can be achieved using the following mapping: replace clause $c = (x_1 \_ :x_2 \_ :x_3)$ by $(x_1 \_ x_c) \wedge (:x_c \_ :x_2 \_ :x_3)$ and replace clause $c = (x_1 \_ x_2 \_ :x_3)$ by $(x_1 \_ x_2 \_ x_c) \wedge (:x_c \_ :x_3)$, where for each clause c, x_c is a new variable, and similarly for the clauses with two variables only. The other clauses remain unchanged. In [21], Theorem 1.4.1 is extended to the maximization version of Monotone 3-SAT.

Theorem 1.4.2. *Let $" > 0$. Unless P=NP, there is no polynomial-time algorithm that, given an instance of Monotone 3-SAT, accepts if is satis able, rejects if is not $(7=8 + ")$-satis able, and is free to accept or reject if none of these cases apply.*

1.5 Treewidth: definitions and some facts

The treewidth of a graph G is a parameter that measures, roughly speaking, how far G is from being a tree. For many optimization problems, one can formulate dynamic programming algorithms that depend polynomially on the size of the graphs, and exponentially on the treewidth. Hence, when the treeewidth is bounded, many NP-Hard problems become tractable, or admit better approximation algorithms. Examples of such problems include the `Maximum Independent Set`, `Minimum Vertex Cover`, and the `Graph Coloring Problem`.

We now formally introduce those concepts. Let $G = (V; E)$ be a graph. A *tree decomposition* of G is a pair $(T; B)$ where T is a tree and $B = fB(i) : i \ 2 \ V(T)g$ is a family of subsets of $V(G)$, called *bags*, one for each vertex of T, satisfying the following:

1. For each $(u; v) \ 2 \ E$, there is at least one bag $B \ 2 \ B$ such that $(u; v) \ 2 \ B$.
2. If $i \ 6= \ j \ 6= \ k \ 2 \ V(T)$ are such that k is on the unique path from i to j in T then $B(i) \setminus B(j) \quad B(k)$.

Note that the second property implies that, for any vertex $u \ 2 \ V(G)$, the bags which contain u form a subtree of T. We will sometimes abuse notation and denote by B both a vertex of $V(T)$ and the bag corresponding to it.

The *width* of a tree decomposition $(T; B)$ is $\max fjBj \quad 1 : B \ 2 \ Bg$. The *treewidth* of G is the minimum integer $!$ such that there is a tree decomposition of G of width $!$.

Let $G = (V; E)$ be a graph and $(T; B)$ be a tree decomposition of G of width $!$. Wlog we can assume that, for each pair of bags $B; B^{\emptyset}$ adjacent in T, $B \setminus B^{\emptyset}$ is a vertex

separator of G. Form a directed rooted tree by picking an arbitrary vertex as the root and orienting the remaining edges towards the root, and call this a *directed tree decomposition*. In the directed tree, each node X other than the root node has exactly one *successor* $S(B)$, i.e., there exists exactly one node $S(B)$ such that $(B; S(B))$ is an arc of the directed tree decomposition. If B is the root, we set $S(B) = ;$. We also say that B is a *predecessor* of $S(B)$ and notice that a bag may have multiple predecessors, and has none if and only if it is a leaf of T.

Dichotomic tree decomposition. Let G be a graph and $(T; B)$ a directed tree decomposition of G of width $!$. We can transform the directed tree decomposition of G it into a directed tree decomposition of the same graph and width where every node has indegree at most 2. We call such a tree decomposition *dichotomic*. Indeed, suppose edges $(B_1; B); (B_2; B); : : : ; (B_t; B)$ with $t \quad 3$ are part of the directed tree decomposition. Then we can create a copy $\bar{B}$ of B, add edge $(\bar{B}; B)$, and split the edges entering B as follows: $(B_1; \bar{B}); : : : ; (B_{dt=2e}; \bar{B})$ and $(B_{dt=2e+1}; B); : : : ; (B_t; B)$. The indegree of X and $\bar{X}$ is at most $bt=2c + 1 < t$. Notice that the digraph we obtain is still a tree decomposition of G, since each edge of G is still covered by a bag, and the bags which contain any vertex $v \; 2 \; G$ still form a continuous subtree.

If we repeat the above operation for a non-dichotomic tree once for each original node of indegree at least 3, the maximum indegree over all nodes of the tree goes from $t > 2$ to $bt=2c+1$, while the number of nodes is at most doubled. Hence, we can iterate the operation so as to obtain a tree decomposition with at most a quadratic number of vertices, all of which have indegree at most 2.

From here on, we assume without loss of generality that our directed tree decomposition is dichotomic. Let T^0 be a subtree of T, and $V(T^0)$ the set of nodes contained in at least a bag of T^0. We say that T^0 is a *closed subtree* of T if $B \; 2 \; V(T^0)$ and B^0 is a predecessor of B, then B^0 belongs to T^0. By connectivity, for each node B of a closed subtree T^0, $S(B)$ also belongs to T^0, with the exception of at most one node

that we call the *head* of T' and denoted by $H(T')$. If $T' \neq T$, the successor of $H(T')$ exists and it is also called the *successor of* T' and denoted by $S(T')$. Note that each bag B is the head of exactly one closed subtree of T, that we denote by T_B.

Remark 1.5.1. *Let* $(T; B)$ *be a dichotomic directed tree decomposition. Then the following holds: let* T' *be a closed subtree of* T *and* B *be the head of* T'*. The removal of* B *partitions* $T' \setminus B$ *in at most* 2 *closed subtrees. The successor of each of those subtrees is* B*.*

Chapter 2

Popular matching problems with edge weights

2.1 Introduction

Although there are efficient algorithms that compute popular matchings of minimum and maximum size in marriage instances, no approach has been developed to deal with more general classes of popular matching problems. This is probably due to the lack of understanding of the structure of popular matching when compared, for instance, to stable matchings. Popular matchings do not form a lattice wrt , and it is not hard to come up with examples where they exhibit a bizarre behavior, e.g., the gap between popular matching of maximum and minimum size is arbitrarily large, while no popular matchings of intermediate size exist.

While understanding the structure of popular matchings, if any, seems to be still beyond reach, one can investigate specific algorithmic questions. A typical way to generalize max and min cardinality problems is to assign weights to edges, and then ask for a feasible matching that maximizes the weight function. Recall that all weighted stable matching problems can be solved efficiently for stable matchings using the lattice structure, see Theorem 1.3.3. Those weights can be used to model a number of constraints, such as forbidding or forcing edges, or finding an *egalitarian* or *minimum regret* matching (see, e.g., [41] for a list of those problems).

We can therefore pose the same question for the set of popular matchings: can we give an efficient algorithm that finds a popular matching of maximum weight? This problem and special cases have been posed as open questions in many papers, includ-

ing [12, 13, 41, 28, 33]. However, very little was known about this. The only notable exception arises from the problem where we ask for a popular matching containing a given edge, or decide that no such matching exists. This can be interpreted as the problem of maximizing a function over the edges with one strictly positive entry, and all other entries being equal to 0. A polynomial algorithm for this problem has been given by Cseh and Kavitha [13]. This results also easily implies that one can find a popular matching without a given edge (resp. with a given node) in polynomial time, or conclude that no such matching exists. To the best of our knowledge, no other special case had been solved.

In this section, we introduce a technique to show hardness results for popular matching problems. This technique is based on a reduction from 3-SAT to a new problem on *stable* matchings, also introduced in here. We show that this idea is very fertile, as it allows us to give very fine-grained answers to the complexity questions mentioned above. These include both NP-hardness results and sharp inapproximability results. We also show how other hardness results that have recently appeared in the literature can be proved with these techniques. This leads to much simpler proofs of those facts. Hence, the "unifying" approach that could not be obtained for fast algorithms, can instead be obtained for hardness results.

In Chapter 4, we will push the connection between 3-SAT and matching under preferences further. In particular, we will show that more hardness results related to stable, popular, and Pareto-optimal matchings can be deduced using this connection to 3-SAT and classical hardness of approximability results the latter.

The rest of the chapter is organized as follows. In Section 2.1, we formally define the main problems under consideration and state our results. The main technical contribution is given in Section 2.2, where a reduction from Monotone 3-SAT to the existence of certain stable matchings is given. In Section 2.3, we use the reduction from Section 2.2 to investigate the complexity of popular matchings with edge and

node weights. We conclude with the analysis of the complexity of finding a popular matching that is not stable or dominant in bipartite graphs, and finding a popular matching in general graphs in Section 2.4.

Problems and results

The main building block of all our hardness results is the following problem.

Problem 2.1.1. Given: *A marriage instance* $(G; <)$. ***Decide:*** *If there exists a stable matching* M *such that* G_M *has no augmenting path.*

For the problem above, we show the following.

Theorem 2.1.1. *Problem 2.1.1 is NP-Complete.*

We then consider several special cases of the problem of finding a popular matching of maximum weight.

Problem 2.1.2 (Popular matching with forbidden and forced elements problem – `pmffe`)**. Given:** *A marriage instance* $(G; <)$, $U^{in}; U^{out}$ $V(G)$, $F^{in}; F^{out}$ $E(G)$, *such that* $U^{in} \setminus U^{out} = ;$, $F^{in} \setminus F^{out} = ;$, *and if* uv 2 F^{in} [F^{out} *then* $u; v$ 2 U^{in} [U^{out}. ***Decide:*** *If there exists a popular matching of* $(G; <)$ *that contains all vertices of* U^{in}, *all edges of* F^{in}, *no vertices of* U^{out} *and no edges of* F^{out}, *or conclude that such matching does not exist.*

Problem 2.1.3 (Maximum Popular matching with positive weights– `mwp`)**. Given:** *A marriage instance* $(G; <)$, *weights* $w : E(G)$! R_+. ***Find:*** *A popular matching* M *of* $(G; <)$ *that maximizes* $w(M)$.

Problem 2.1.4 (Minimum weight popular matching with positive weights – `miwp`)**. Given:** *A marriage instance* $(G; <)$, *weights* $w : E(G)$! R_+. ***Find:*** *A popular matching* M *of* $(G; <)$ *that minimizes* $w(M)$.

Let M be a popular matching that is neither dominant nor stable. If the size of M is strictly less than the size of a dominant matching and is strictly larger than the size of a stable matching, then we call M a *middle* matching; if it equals to the size of a dominant matching, we call it *hidden*.

Problem 2.1.5 (Hidden matching – hm). **Given:** *A marriage instance* $(G; <)$. ***Decide:*** *If there exists a hidden matching* M *of* $(G; <)$.

For those problems, we show the following tight hardness, approximability, and hardness of approximation results.

Theorem 2.1.2. *If* $U^{out} = F^{in} = F^{out} = ;$ *or* $U^{in} = F^{in} = F^{out} = ;$, *or* $jU^{in} [U^{out} [F^{in} [F^{out}j \quad 1$, *then* pmffe *can be solved in polynomial time. Otherwise, for each other set of values for* $jU^{in}j,jU^{out}j,jF^{in}j,jF^{out}j$, pmffe *is NP-hard.*

Theorem 2.1.3. miwp *and* mwp *are NP-Hard. Unless P=NP,* miwp *cannot be approximated in polynomial time up to any factor, and* mwp *cannot be approximated in polynomial time better than a factor* $\frac{1}{2}$. *On the other hand, there is a polynomial-time algorithm that computes a* $\frac{1}{2}$*-approximation to* mwp.

We remark that the hardness of mwp has been independently shown in [34], but to the best of our understanding, results from [34] does not imply any hardness of approximation.

Theorem 2.1.4. hm *is NP-complete.*

Hardness results follow from a similar reduction as the one introduced to prove Theorem 2.1.1 and from the combinatorial characterization of popular matchings given by Theorem 1.3.5. The approximation algorithm builds on a decomposition result from [13] that partitions a popular matching into a stable and a dominant matching.

We then move to the following problems, whose hardness had been recently established in the literature [20, 17, 35].

Problem 2.1.6 (Middle matching – mm)**. Given:** *A marriage instance system* $(G; <)$*.* ***Find:*** *A middle matching* M *of* $(G; <)$*.*

Problem 2.1.7 (Popular matching in general graphs – pmgg)**. Given:** *A preference system* $(G; <)$*.* ***Find:*** *A popular matching of* $(G; <)$*, or conclude that such matching does not exist.*

These hardness results can be shown by simple modifications of our construction, showing that a uniform approach to all those problems is possible. Those proofs are, we believe, simpler than the original.

Theorem 2.1.5. *pmgg and mm are NP-hard.*

2.2 The reduction

From a Monotone 3-SAT formula to $G(\ )$

Consider a Monotone 3-SAT instance . We denote the set of variables by $X = \{x_1, \ldots, x_n\}$, the set of clauses by $C = \{c_1, \ldots, c_m\}$, and the set of literals by $L = \{\ell_1, \ldots, \ell_p\}$, with $p \quad 3m$. Literals of the form x_i are called *positive* (*occurrences of variable* i), while those of the form $\neg x_i$ are called *negative* (*occurrences of variable* i). We refer therefore to clauses of a Max Monotone 3-SAT instance as *positive clauses* and *negative clauses* accordingly. From now on, we assume without loss of generality that does not consist of only positive (resp. negative) clauses and that no variable only appears as a positive (resp. negative) literal.

Define the *positive gadget* to be the graph H constructed as in Fig. 2.1, left. Node a is called the *apex* of H. Node c is called the *consistency enforcer* of H. Node g is

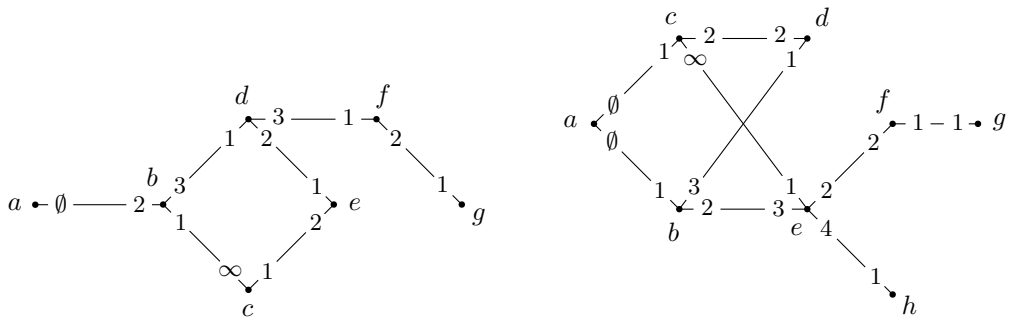

Figure 2.1: The gadget associated to the positive (left) and negative (right) literals, and the corresponding canonical incomplete preference patterns.

called the *gateway* of H.

Define the *negative gadget* to be the graph N constructed as in Fig. 2.1, right. Similarly to the positive gadget, we call node a (resp. c, g) the *apex* (resp. *consistency enforcer*, *gateway*) of N. Moreover, we call edge eh the *evicted* edge of N. Fig. 2.1, right and right, also define incomplete preference patterns I_N, which we again call *canonical*.

To each positive (resp. negative) literal ℓ of , we associate one positive (resp. negative) gadget $H(\ell)$ (resp. $N(\ell)$). We will write $G(\ell) = (V(\ell); E(\ell))$ to denote the gadget associated to a literal ℓ, whether it is positive or negative. Nodes of $G(\ell)$ will be denoted by $a(\ell)$, $b(\ell)$, etc.

Fix an arbitrary order of clauses of and of literals in each clause. We define four graphs $G_s := G_s(\ )$, $G_g := G_g(\ )$, $G_m := G_m(\ )$ and $G_h := G_h(\ )$ as follows. Those four graphs will be used in different hardness reductions. They are all variations of a graph defined as follows. First, construct disjoint graphs $G(\ell)$ for each literal ℓ, and add nodes u and v. Now fix a clause c, and let $\ell_1; \ell_2; \ell_3$ be the literals from the clause, in this order (with ℓ_3 possibly not present). Identify the apex of $G(\ell_1)$ with u; the apex of $G(\ell_i)$, i 2 with the gateway of $G(\ell_{i-1})$; and add edge $g(\ell_k)v$, where ℓ_k is the last literal of the clause. Repeat the operation for all clauses (note that vertices u and v are unique). Let $\bar{G} := G(\bar{V}; \bar{E})$ be the graph obtained. Let

$G_s = G(\underbrace{\bar{V} \cup \{s, t\}}_{V(G_s)}, \underbrace{\bar{E} \cup E \cup \{st, tu\}}_{E(G_s)})$, where

$E := \bigcup_{i=1,\ldots,n} \{c(\ell)c(\ell') : \ell$ (resp. ℓ') is a positive (resp. negative) occurrence of variable $i\}$

and $s, t \notin \bar{V}$. Define graphs G_g, G_m, G_h as follows:

$V(G_g) := V(G_s) \cup \{p_1, p_2, p_3, p_4, w, x, y\}$ and $E(G_g) := E(G_s) \cup E_1^g$, where

$$E_1^g := \{p_1s, p_2s, p_3p_4, p_3p_1, p_4p_2, vw, wx, xy, ywg\},$$

$V(G_m) := V(G_s) \cup \{w, x, y\}$ and $E(G_m) := E(G_s) \cup E_1^m$, where:

$$E_1^m := \{vw, wx, xyg\},$$

$V(G_h) := V(G_s) \cup \{t_1, u_1, w, x, y\}$ and $E(G_h) := E(G_s) \cup E_1^h$, where:

$$E_1^h := \{tt_1, uu_1, vw, wx, xy, yvg\}.$$

See Fig. 2.3, 2.4, 2.5 for an example of graphs $G_g(\ )$, $G_m(\ )$, and $G_h(\ )$ respectively.

Lemma 2.2.1. *$G_s(\ )$, $G_m(\ )$ and $G_h(\ )$ are bipartite.*

Proof. First, we will show that $G_m(\ )$ is bipartite. We assign each vertex from $V := V(G_m(\ ))$ to exactly one of two sets A or B, and prove that if $u, v \in A$ (or $u, v \in B$) then $uv \notin E := E(G_m(\ ))$. The proof can be followed in Fig. 2.4, where nodes are colored according to the set of the bipartition they belong to. Assign s, u, w, y to A and t, v, x to B. For any ℓ corresponding to a positive literal, assign $a(\ell), d(\ell), c(\ell), g(\ell)$ to A and $b(\ell), e(\ell), f(\ell)$ to B. For any ℓ corresponding to a negative literal, assign $a(\ell), d(\ell), e(\ell), g(\ell)$ to A and $b(\ell), c(\ell), f(\ell), h(\ell)$ to B. Note that this assignment is consistent with the fact that $a(\ell) \neq g(\ell')$ for some literals ℓ and

ℓ', and that some $a(\ell)$ are identified with u (they are all assigned to A). One easily checks that edges in all gadgets connect nodes in different sets of the bipartition. Now consider $c(\ell)c(\ell')$. By construction, we can assume wlog that ℓ (resp. ℓ') is a positive (resp. negative) literal. Hence $c(\ell)\ 2\ A$, $c(\ell')\ 2\ B$, as required. Edges $st; tu; vw; wx; xy; yv$ have vertices in different sets of the bipartition as well.

$G_h(\ )$ is also bipartite, since we can use the vertex assignment of graph $G_m(\ )$ and additionally assign t_1 and u_1 to A and B respectively.

$G_s(\ )$ is also bipartite, since we can use the vertex assignment of graph $G_m(\ )$ and remove vertices $w; y$ and their assignments. □

In the following, we omit explicit dependence on . Moreover, when a statement is true for all graphs G_s, G_g, G_m, and G_h, we drop the subscript and simply write it holds for G.

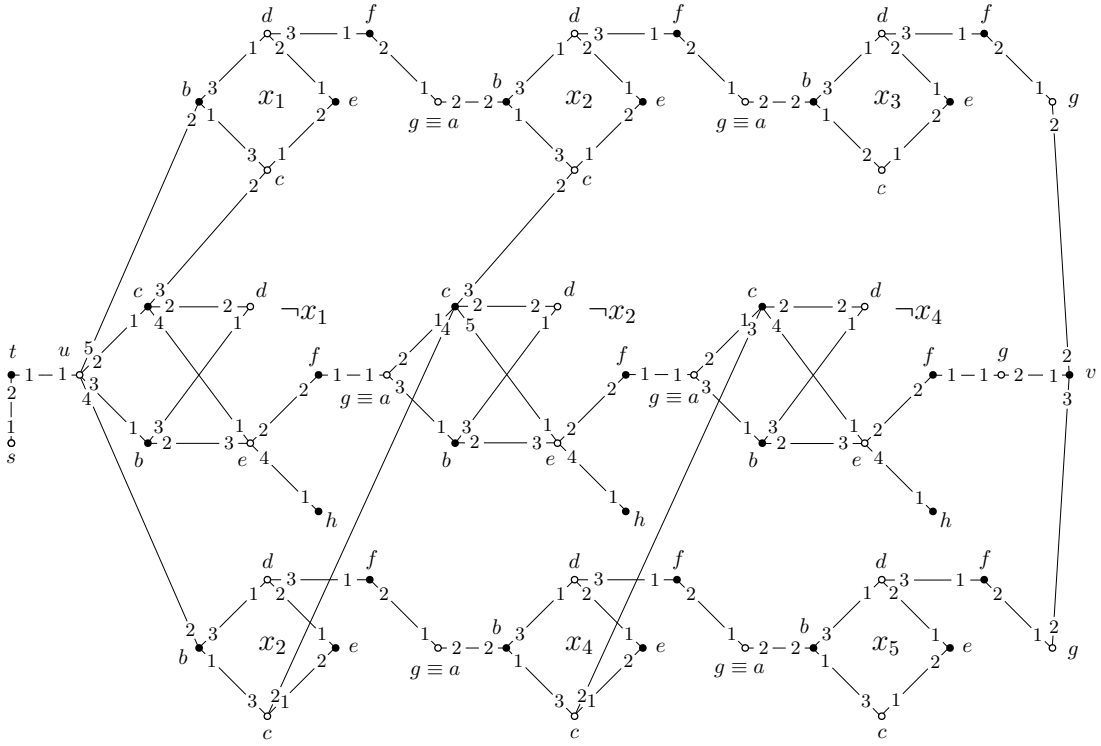

Figure 2.2: A monotone 3-SAT instance $= (x_1 \_ x_2 \_ x_3) \wedge (\neg x_1 \_ \neg x_2 \_ \neg x_4) \wedge (x_2 \_ x_4 \_ x_5)$ and the associated graph $G_s(\ )$ and preference system. Disk and circle labels of vertices provide a bipartition of $G_s(\ )$.

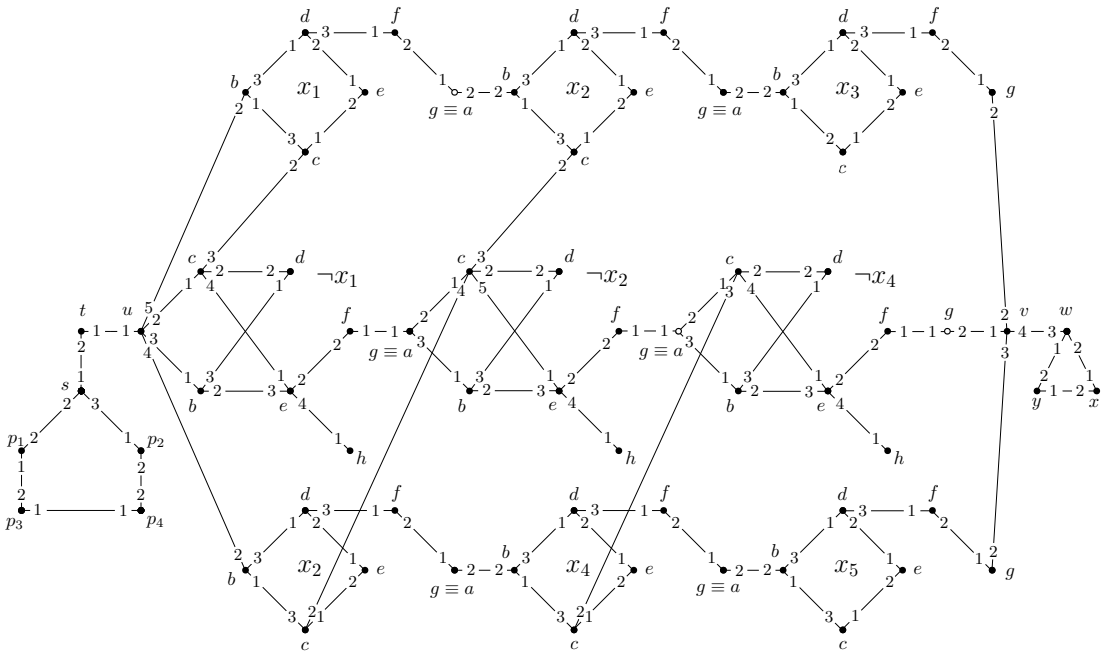

Figure 2.3: A monotone 3-SAT instance $= (x_1 \_ x_2 \_ x_3) \wedge (\neg x_1 \_ \neg x_2 \_ \neg x_4) \wedge (x_2 \_ x_4 \_ x_5)$ and the associated graph $G_g(\ )$ and preference system. Disk and circle labels of vertices provide a bipartition of $G_g(\ )$.

The preference system

We now define the preference system for the graphs G_s, G_g, G_m, and G_h defined in the previous section.

Consider the union I of all canonical incomplete preference patterns for gadgets corresponding to literals of . Recall that vertex sets of gadgets are disjoint subsets of V, with the exception of $g(\ell_1)$ $a(\ell_2)$, with ℓ_2 being the literal following ℓ_1 in some clause, and u $a(\ell)$, where ℓ is the first literal in some clause. However, the canonical incomplete preference patterns assign $;$ to edges incident to nodes $a(\ell)$. Hence, I is an incomplete preference pattern for G.

We will construct now preference patterns $L_g; L_m$ and L_h, see Fig. 2.2, 2.3, 2.4, 2.5. (Recall that, for the sake of shortness, we write $L(v; v') =$ when $L_s(v; v') = L_g(v; v') = L_m(v; v') = L_h(v; v') =$). They are defined as follows:

$L_s; L_g; L_m$ and L_h agree with I;

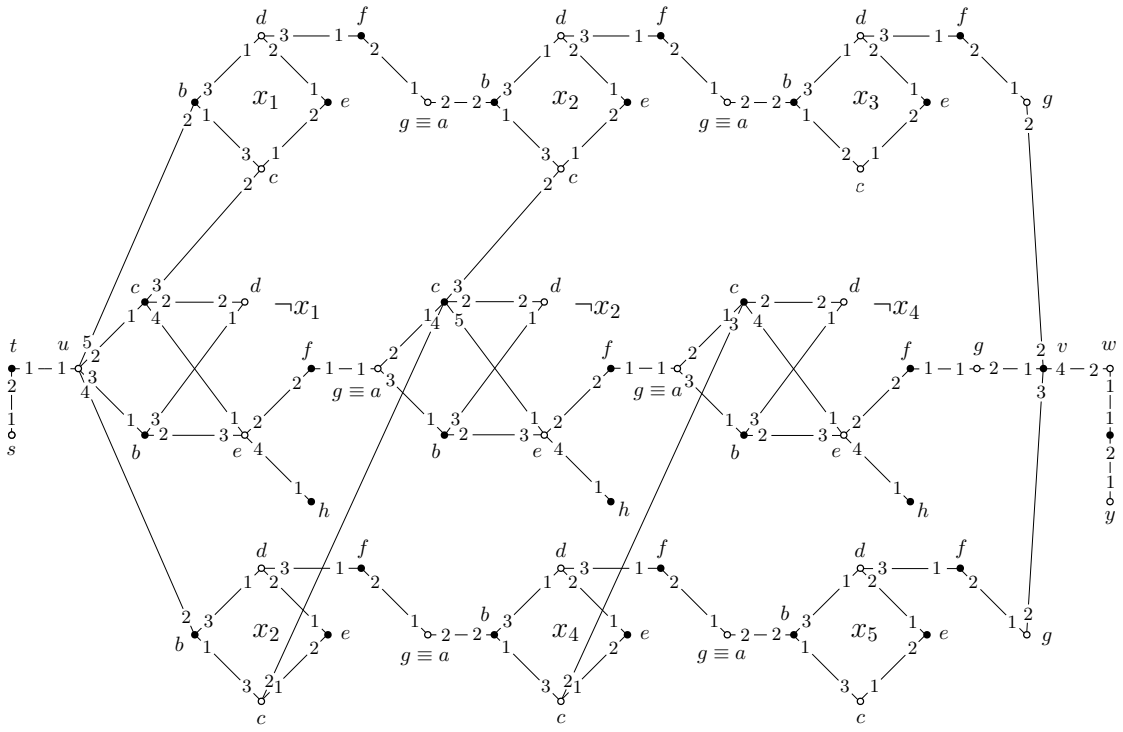

Figure 2.4: A monotone 3-SAT instance $= (x_1 \_ x_2 \_ x_3) \wedge (:x_1 \_ :x_2 \_ :x_4) \wedge (x_2 \_ x_4 \_ x_5)$ and the associated graph $G_m(\ )$ and preference system. Disk and circle labels of vertices provide a bipartition of $G_m(\ )$.

For edges not contained in $\bar{E}$, set: $L(s;t) = 1$, $L_s(t;s) = L_g(t;s) = L_m(t;s) = 2; L_h(t;s) = 3$, $L(u;t) = L(t;u) = 1$, $L_g(s;p_1) = 2$, $L_g(s;p_2) = 3$, $L_g(p_1;s) = 2$, $L_g(p_2;s) = 1$, $L_g(p_1;p_3) = 1$, $L_g(p_3;p_1) = 2$, $L_g(p_3;p_4) = L_g(p_4;p_3) = 1$, $L_g(p_4;p_2) = L_g(p_2;p_4) = 2$, $L_h(t;t_1) = L_h(u;u_1) = 2$, $L_h(t_1;t) = L_h(u_1;u) = 1$, $L_h(u_1;t_1) = L_h(u_1;t_1) = 2$, $L_g(v;w) = j\Gamma(v)j$, $L_g(w;x) = 2$, $L(x;w) = 1$, $L(x;y) = 2$, $L(y;x) = 1$, $L_g(y;w) = 2$, $L_g(w;y) = 1$, $L_m(v;w) = L_h(v;w) = j\Gamma(v)j \quad 1$, $L_m(v;y) = L_h(v;y) = j\Gamma(v)j$, $L_m(w;v) = L_h(w;v) = 2$, $L_m(y;v) = L_h(y;v) = 2$, $L_g(w;v) = 3$, $L(x;w) = 1$, $L(x;y) = 2$, $L(y;x) = 1$;

we complete the preference list for the remaining edges as follows:

- $ub(`)$, for all literals $`$ that are first in some clause, and $uc(`)$ for all negative literals that are first in some clause. Since $L_g(u;t) = L_m(u;t) = 1$ and no other edge is incident to $u$, we can assign $L_g(u;v) = L_g(u;v)$ for $v = b(`)$

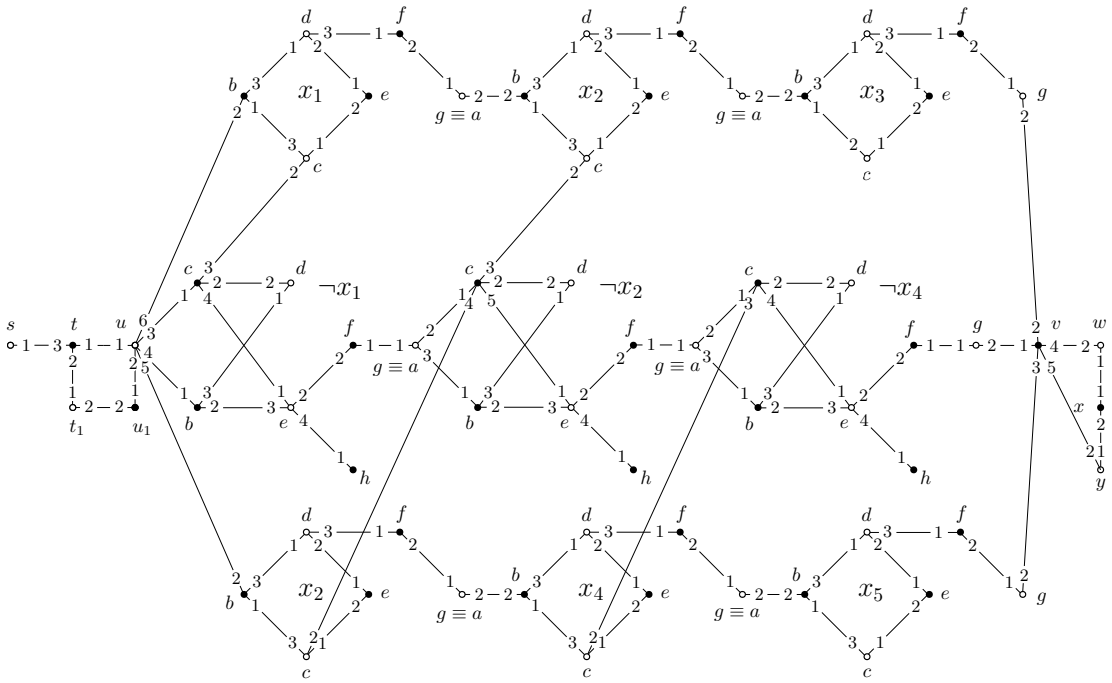

Figure 2.5: A monotone 3-SAT instance $= (x_1 \_ x_2 \_ x_3) \wedge (:x_1 \_ :x_2 \_ :x_4) \wedge (x_2 \_ x_4 \_ x_5)$ and the associated graph $G_h(\ )$ and preference system. Disk and circle labels of vertices provide a bipartition of $G_h(\ )$.

or $v \in \{b(`), c(`)\}g$, through an arbitrary bijection to $\{2, \ldots, |\Gamma(v)|\}g$. Since $L_h(u,t) = 1, L_h(u,u_1) = 2$ and no other edge is incident to u, we can assign $L_h(u,v))$ for $v = b(`)$ or $v \in \{b(`), c(`)\}g$, through an arbitrary bijection to $\{3, \ldots, |\Gamma(v)|\}g$.

- $g(`)v$ and $vg(`)$, for all literals $`$ that are last in some clause. Since all such $g(`)$ have degree 2 in G, and $L(g(`), f(`)) = 1$, we set $L(g(`), v) = 2$. Moreover, $\Gamma(v)$ is composed of all such $g(`)$ and vertex w for G_g, to which we already assigned $L_g(v,w) = |\Gamma(v)|$. We assign therefore $L_g(v, g(`))$ via an arbitrary bijective map to $\{1, \ldots, |\Gamma(v)| \quad 1\}g$. For $G_m$ and $G_h$, $\Gamma(v)$ is composed of all such $g(`)$, vertices w and y, to which we already assigned $L_m(v,w) = L_h(v,w) = |\Gamma(v)| \quad 1$ and $L_m(v,y) = L_h(v,y) = |\Gamma(v)|$. We assign therefore $L_m(v, g(`)) = L_h(v, g(`))$ via an arbitrary bijective map to $\{1, \ldots, |\Gamma(v)| \quad 2\}g$.

- $c(\ell)c(\ell')$ and $c(\ell')c(\ell)$, where ℓ is a positive occurrence and ℓ' is a negative occurrence of the same variable. Note that we already assigned $L(c(\ell); e(\ell)) = 1$, $L(c(\ell); b(\ell)) = |\Gamma(c(\ell))|$. We assign $L(c(\ell); c(\ell'))$ for a fixed $c(\ell)$ via an arbitrary bijective map to $\{2, \ldots, |\Gamma(c(\ell))| - 1\}$. Similarly, we assign $L(c(\ell'); c(\ell))$ for a fixed $c(\ell')$ via an arbitrary bijective map to $\{3, \ldots, |\Gamma(c(\ell'))| - 1\}$
- $a(\ell)b(\ell)$ for all positive literals ℓ that are not first in some clause. Recall that we have $L(a(\ell); f(\ell')) = 1$ for some ℓ'. We set therefore $L(a(\ell); b(\ell)) = 2$.
- $a(\ell)b(\ell)$, $a(\ell)b(\ell)$ for all negative literals ℓ that are not first in some clause. Recall that we have $L(a(\ell); f(\ell')) = 1$ for some ℓ'. We set therefore $L(a(\ell); b(\ell)) = 3$, $L(a(\ell); c(\ell)) = 2$.

We now derive properties of matchings that are popular in any of the preference systems introduced above and satisfy some further hypothesis. Recall that we omit the subscript and denote by $(G, \prec)$ any of the instances defined above .

In the proof of the next and following lemmas, we often use the characterization of popular matchings given in Theorem 1.3.5, hence we omit referring explicitly to this theorem each time.

Properties of certain popular matchings of $(G, \prec)$

Lemma 2.2.2. *Let M be a popular matching of $(G, \prec)$ that does not contain any evicted edge. Then M does not contain any consistency edge, i.e., $M \setminus E = \emptyset$.*

Proof. Suppose by contradiction that M contains a consistency edge, connecting a positive gadget $H(\ell)$ and a negative gadget $N(\ell')$. We focus now on nodes of $N(\ell')$, hence omit the dependency on ℓ'. Since the consistency edge incident to c is matched, if d is not matched, then cd would be a $(+, +)$ edge adjacent to an unmatched

vertex, which contradicts popularity. Hence, $bd \in M$, which in turn implies $ef \in M$, otherwise be is a $(+;+)$ edge adjacent to an unmatched vertex, again a contradiction (recall that $eh \in M$ by hypothesis). This implies that he is a $(+;\ )$ edge, fg is a $(+;+)$ edge, and $h;e;f;g$ is an M-alternating path containing a $(+;+)$ edge from an unmatched vertex, a contradiction. □

Lemma 2.2.3. *Let M be a popular matching of $(G;<)$ that contains tu. Then:*

1. *The gateway (resp. apex) of each gadget is matched to its favorite partner;*
2. *M does not contain any consistency or evicted edges;*
3. *In each positive gadget $H(\grave{})$, exactly one of the following is true (dependency on $\grave{}$ is omitted):*

   a) $F(H) := fbd;ceg \quad M$, or

   b) $T(H) := fbc;deg \quad M$.

   Similarly, in each negative gadget $N(\grave{}')$, exactly one of the following is true (dependency on $\grave{}'$ is omitted):

   a) $F(N) := fcd;beg \quad M$, or

   b) $T(N) := fce;bdg \quad M$.

4. *If $\grave{}$ is a positive and $\grave{}'$ a negative occurrence of the same variable, then we cannot have both $T(H(\grave{})) \quad M$ and $T(N(\grave{})) \quad M$.*

Proof. 1. We consider negative and positive gadgets separately. Note that it is enough to prove the statement for the gateway, since the one for the apex follows by the fact that each apex is either also a gateway, or it is node u (that is matched to its favorite partner t by hypothesis).

Pick first a negative clause and one of its literals $\grave{}$. We omit the dependency on $\grave{}$ in nodes of $N(\grave{})$. The favorite partner of g is f. Hence, suppose by contradiction $fg \in M$. Note that since g is also f's favorite partner, fg is a $(+;+)$ edge, which

implies $fe \in M$, else f is an unmatched node adjacent to a $(+,+)$ edge starting from an unmatched node, a contradiction. Edge eh is therefore labeled $(\ ,+)$. The path h,e,f,g is therefore an M-alternating path from an unmatched vertex to a $(+,+)$ edge, a contradiction concluding the proof for this case.

Consider any positive clause, and take, among its literals, the first one that violates the thesis. Call this literal ℓ. Again, we omit the dependency on ℓ in nodes of $H(\ell)$, and observe that f is g's favorite partner. Therefore, $fd \in M$, else f is unmatched and fg is a $(+,+)$ edge adjacent to an unmatched node, a contradiction. If e is unmatched, then de is a $(+,+)$ adjacent to an unmatched node, again a contradiction. Since $\Gamma(e) = \{d,c\}$ and $df \in M$, we conclude $ec \in M$. A similar argument (with e replaced by b) implies $ab \in M$. Now, if ℓ is the first literal of the clause, $a = u$, contradicting $tu \in M$. Else, $a = g(\ell')$ for the literal ℓ' preceding ℓ in the clause, contradicting the choice of ℓ.

2. Suppose by contradiction M contains an evicted edge. Pick any clause c with a negative literal such that M contains one or more evicted edges, and let $N(\ell)$ be the gadget associated to the first literal in clause c with $e(\ell)h(\ell) \in M$. We omit the dependency on ℓ in nodes of $N(\ell)$. Part 1 implies that a is matched to its favorite partner. The hypothesis implies that $eh \in M$. Hence, if $bd \notin M$, then be is a $(+,+)$ edge incident to the unmatched node b, a contradiction. Hence $bd \in M$. Since a,d,e are not matched to c in M, if c is not matched through a consistency edge it is unmatched, and ce is $(+,+)$ edge, a contradiction.

Hence $cc(\ell') \in M$ for some $H(\ell')$. Arguments similar to those above imply $d(\ell')e(\ell') \in M$ and $a(\ell')b(\ell') \in M$, contradicting part 1. This concludes the proof that M does not contain any evicted edge. The fact that M does not contain any consistency edge immediately follows from what just proved and Lemma 2.2.2.

3. Fix a gadget $G(\ell)$ and omit dependency on ℓ in nodes. By part 1 and 2, a is matched to its first choice, and M does not contain any consistency or evicted edges.

Hence, we can restrict our attention to the subgraph induced by $b; c; d; e$. Simple case checking shows that, if any of those is unmatched, then there is a $(+;+)$ edge incident to an unmatched node, a contradiction to popularity. Hence, each of these vertices must be matched, and since they form a cycle with 4 vertices, one of the possibilities from the thesis of the lemma must hold.

4. Suppose we have both $T(H(\grave{}))$ M and $T(N(\grave{}'))$ M. Then the consistency edge $c(\grave{})c(\grave{}')$ is $(+;+)$. Hence, $h(\grave{}'); e(\grave{}'); c(\grave{}'); c(\grave{})$ is an M-alternating path from an unmatched node to a $(+;+)$ edge, a contradiction. □

Corollary 2.2.1. *If M is a popular matching of $(G;<)$, and tu 2 M, then there are no edges labelled $(+;+)$ in the subgraph of G induced by $\bar{V}$.*

Proof. From Lemma 2.2.3 we know that the gateway of each gadget is matched to its favorite partner because of Lemma 2.2.3, parts 3 and 4. Using Lemma 2.2.3, part 3 and 4, one easily checks that no edge between nodes of $\bar{V}$ can be labelled $(+;+)$. □

Lemma 2.2.4. *Let M be a popular matching of $(G;<)$ such that tu 2 M and there exists a clause c such that $F(\grave{})$ M for all literals $\grave{}$ of c. Then there is M-alternating path from s to v in G_M.*

Proof. Since tu 2 M, by Lemma 2.2.3, apexes and gateways are matched to their favourite partner. We assume that c is given by exactly three literals – say $\grave{}_1; \grave{}_2; \grave{}_3$. Similar arguments apply for the case with less than three literals. The construction of the M-augmenting path can be followed in Fig. 2.6.

Assume first $\grave{}$ to be a positive clause, let $H(\grave{}_1); H(\grave{}_2); H(\grave{}_3)$ be the corresponding gadgets, and assume $F(\grave{}_1); F(\grave{}_2); F(\grave{}_3)$ M. Then st, $a(\grave{})b(\grave{})$, $d(\grave{})f(\grave{})$, $g(\grave{}_3)v$ are $(+;\ )$ or $(\ ;+)$ edges for $\grave{} = \grave{}_1; \grave{}_2; \grave{}_3$, and the promised M-alternating path is:

$$s; t; u; b(\grave{}_1); d(\grave{}_1); f(\grave{}_1); g(\grave{}_1); b(\grave{}_2); d(\grave{}_2); f(\grave{}_2); g(\grave{}_2); b(\grave{}_3); d(\grave{}_3); f(\grave{}_3); g(\grave{}_3); v:$$

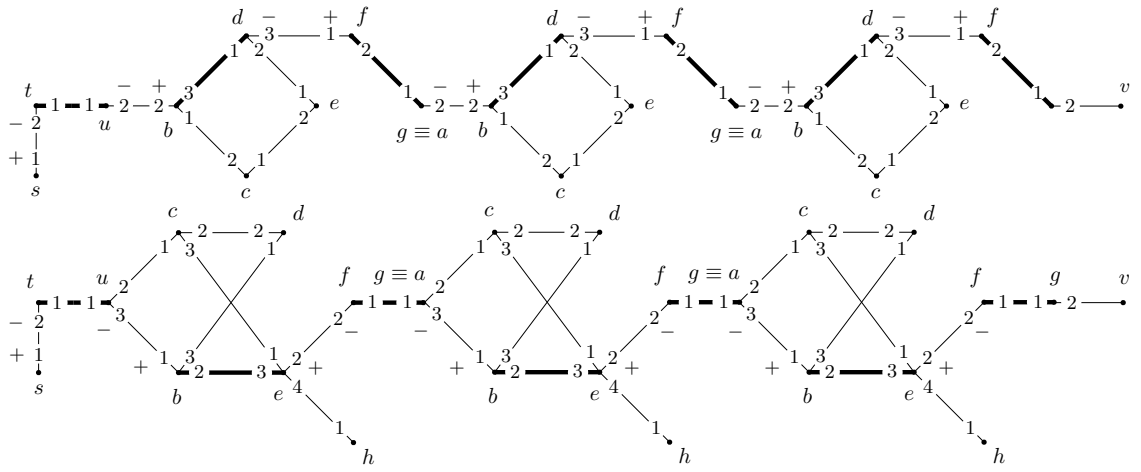

Figure 2.6: An alternating path from s to w when all literals are False in positive (above) and negative (below) clauses.

Now assume that ℓ is a negative clause, and let $N(\ell_1); N(\ell_2); N(\ell_3)$ be the gadget corresponding to its literals Then st, $a(\ell)b(\ell)$, $e(\ell)f(\ell)$, $g(\ell_3)v$ are $(+;\ )$ or $(\ ;+)$ edges for $\ell = \ell_1; \ell_2; \ell_3$, and the the augmented path is

$$s; t; u; b(\ell_1); e(\ell_1); f(\ell_1); g(\ell_1); b(\ell_2); e(\ell_2); f(\ell_2); g(\ell_2); b(\ell_3); e(\ell_3); f(\ell_3); g(\ell_3); v:$$

□

From matchings to feasible assignments

We are now ready to prove one direction of the relation between feasible assignments for and matchings in the graphs we constructed.

Lemma 2.2.5. *Let M be a stable matching without an s v augmenting path in G_s (resp. a popular matching that does not contain wx and contains tu in G_m, G_h or G_g). Then there exists a feasible assignment in .*

Proof. Note that in all cases M is popular. In G_s, t and u are each other's favorite partners and M is assumed to be stable. We conclude that $tu \in M$ for all cases.

Define an assignment $\bar{x}(M)$ of the variables of as follows: if $T(\grave{})$ M for some positive (resp. negative) occurrence $\grave{}$ of variable x_i, then set $\bar{x}_i(M)$ to be true (resp. false). Else, set $\bar{x}_i(M)$ to true or false arbitrarily. Note that $\bar{x}(M)$ is well-defined, since by Lemma 2.2.3, part 4, we cannot assign the same variable to true and false simultaneously.

We now prove that $\bar{x}(M)$ is a feasible assignment. It suffices to show that, for each clause c, $T(\grave{})$ M for some literal $\grave{}$ in clause c. For G_s, the statement follows by Lemma 2.2.4.

In the cases of G_m and G_h, wx is a $(+;+)$ edge and s is unmatched. Assume by contradiction that there is a clause c' such that all literals are set to false. We consider cases with positive and negative literals separately. Without loss of generality assume that c' contains exactly three positive literals $\grave{}_1;\grave{}_2;\grave{}_3$. Notice that by Lemma 2.2.3, part 3, $F(\grave{}_1); F(\grave{}_2); F(\grave{}_3)$ M, and st, $a(\grave{})b(\grave{})$, $d(\grave{})f(\grave{})$, $g(\grave{}_3)v$ are $(+;\ )$ or $(\ ;+)$ edges for $\grave{} = \grave{}_1;\grave{}_2;\grave{}_3$. There is an M-alternating path:

$$s; t; u; b(\grave{}_1); d(\grave{}_1); f(\grave{}_1); g(\grave{}_1); b(\grave{}_2); d(\grave{}_2); f(\grave{}_2); g(\grave{}_2); b(\grave{}_3); d(\grave{}_3); f(\grave{}_3); g(\grave{}_3); v; w; x:$$

Consider now a case with three negative literals $\grave{}_1;\grave{}_2;\grave{}_3$. Then st, $a(\grave{})b(\grave{})$, $e(\grave{})f(\grave{})$, $g(\grave{}_3)v$ are $(+;\ )$ or $(\ ;+)$ edges for $\grave{} = \grave{}_1;\grave{}_2;\grave{}_3$. The following is an M-alternating path:

$$s; t; u; b(\grave{}_1); e(\grave{}_1); f(\grave{}_1); g(\grave{}_1); b(\grave{}_2); e(\grave{}_2); f(\grave{}_2); g(\grave{}_2); b(\grave{}_3); e(\grave{}_3); f(\grave{}_3); g(\grave{}_3); v; w; x; y:$$

In both case we obtain a contradiction to popularity, since the above alternating paths contain an unmatched node s and $wx = (+;+)$.

In the case of G_g, wx is a $(+;+)$ edge and p_2 is unmatched, otherwise $p_2 s$ 2 M or s is unmatched. In both cases this would contradict popularity. We conclude that s is matched, otherwise $sp_2 = (+;+)$ and p_2 is unmatched, which would contradict

popularity. Assume by contradiction that there is a clause c' such that all literals are set to false. We consider cases with positive and negative literals separately. Without loss of generality assume that c' contains exactly three positive literals ℓ_1, ℓ_2, ℓ_3. Notice that by Lemma 2.2.3, part 3, $F(\ell_1), F(\ell_2), F(\ell_3) \in M$, and p_3p_1, p_2p_4, st, $a(\ell)b(\ell)$, $d(\ell)f(\ell)$, $g(\ell_3)v$ are $(+,-)$ or $(-,+)$ edges for $\ell = \ell_1, \ell_2, \ell_3$. There is an M-alternating path: $p_2, p_4, p_3, p_1, s, t, u, b(\ell_1), d(\ell_1), f(\ell_1), g(\ell_1), b(\ell_2), d(\ell_2), f(\ell_2), g(\ell_2), b(\ell_3), d(\ell_3), f(\ell_3), g(\ell_3), v, w, x$. Consider now a case with three negative literals ℓ_1, ℓ_2, ℓ_3. Then p_3p_1, p_2p_4, st, $a(\ell)b(\ell)$, $e(\ell)f(\ell)$, $g(\ell_3)v$ are $(+,-)$ or $(-,+)$ edges for $\ell = \ell_1, \ell_2, \ell_3$. The following is an M-alternating path: $p_2, p_4, p_3, p_1, s, t, u, b(\ell_1), e(\ell_1), f(\ell_1), g(\ell_1), b(\ell_2), e(\ell_2), f(\ell_2), g(\ell_2), b(\ell_3), e(\ell_3), f(\ell_3), g(\ell_3), v, w, x, y$. Similarly to above in both case we obtain a contradiction to popularity, since the above alternating paths contain an unmatched node p_2 and $wx = (+,+)$.

Since in each clause there is at least an ℓ such that $T(\ell) \in M$, we conclude that is satisfied by the assignment we constructed. □

From feasible assignments to matchings

Now suppose that we have a consistent assignment of the literals, i.e., an assignment of the variables, which satisfies . We show how to construct matchings M_s in G_s, M_m in G_m, M_g in G_g, and M_h in G_h. As before, we drop the subscript when a certain statement holds for all the four cases.

add tu and, for each literal ℓ, $f(\ell)g(\ell)$ to M;

for for each positive literal ℓ, add $T(\ell)$ to M if the corresponding variable is set to true; add $F(\ell)$ to M if the corresponding variable is set to false;

for each negative literal ℓ, add $T(\ell)$ to M if the corresponding variable is set to false; add $F(\ell)$ to M if the corresponding variable is set to true;

add vw, xy to M when $M \not\subseteq M_s$;

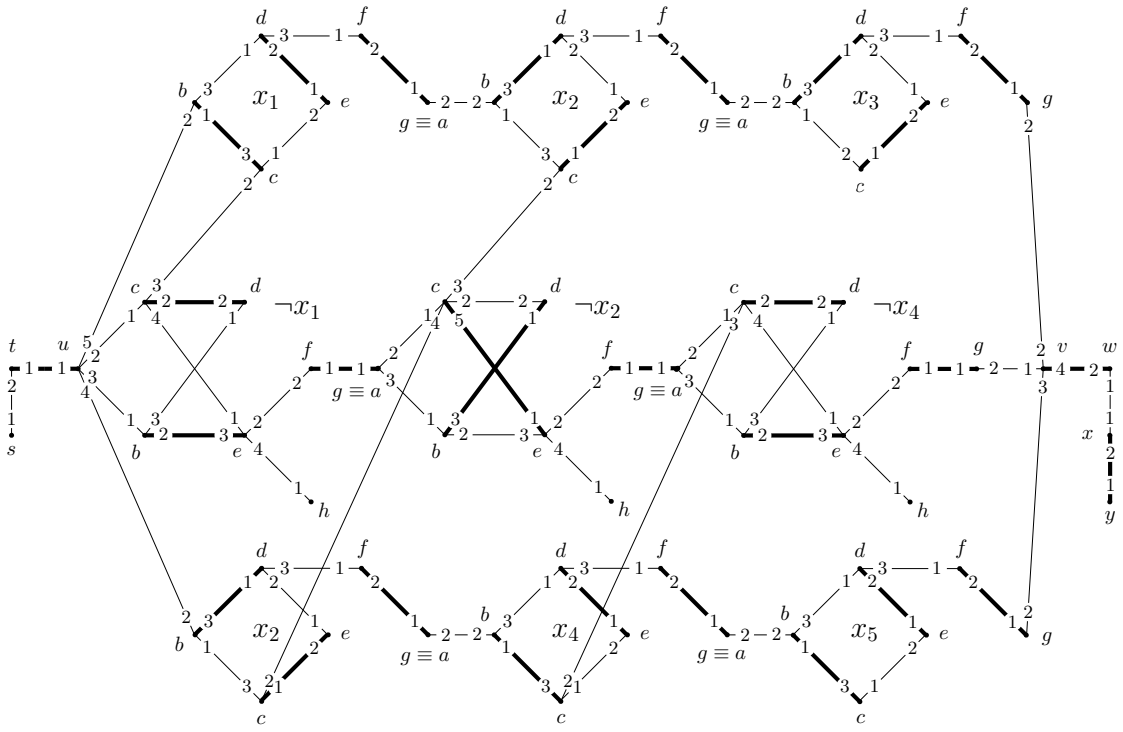

Figure 2.7: A matching M (bold edges) of G_m corresponding to a feasible assignment of $= (x_1 \_ x_2 \_ x_3) \wedge (: x_1 \_ : x_2 \_ : x_4) \wedge (x_2 \_ x_4 \_ x_5)$ from Fig. 2.4: $x_1; x_4; x_5$ are set to True and $x_2; x_3$ to False.

add p_3p_4, p_1s to M_g; add t_1u_1 to M_h.

Fig. 2.7 gives an example of a matching M_m corresponding to a feasible assignment of the clause from Fig. 2.4 in G_m. We show the following.

Lemma 2.2.6. *M is popular in G. Moreover, M_s is a stable matching in G_s that does not contain an M-augmenting path between s and v in G_M.*

Proof. We start by investigated the location of $(+;+)$ edges in G.

Claim 2.2.1. *There are no edges labelled $(+;+)$ in M in the subgraph of G induced by $\overline{V}$.*

Proof. Let ij be a $(+;+)$ edge in M for some vertices i and j in the given subgraph. This cannot be incident to $t; u; a(\grave{}); g(\grave{})$ for each $\grave{}$, and $f(\grave{})$ for negative literals $\grave{}$, since those are matched to their favorite partner. Since the neighbors of both s and v

are either t or $g(\ell)$ for some ℓ, it cannot be incident to v and s as well. Moreover, in positive literals ℓ, we have that $d(\ell)$ is always matched to someone that it prefers to $f(\ell)$, hence no $(+,+)$ edge is incident to $f(\ell)$. We are left with edges between nodes b, c, d, e of the gadgets.

Suppose first, ij is a consistency edge, say $c(\ell)c(\ell')$, with ℓ positive literal. Then $c(\ell)b(\ell), c(\ell')e(\ell') \in M$. By construction, this implies that $T(H(\ell)), T(N(\ell')) \notin M$, a contradiction to the fact that $\overline{X}$ is an assignment to variables in .

Hence, ij must connect nodes b, c, d, e within the same gadget. This cannot happen. □

Claim 2.2.2. *The only edge labeled $(+,+)$ in G_g, G_m and G_h is wx. There are no edges labeled $(+,+)$ in G_s.*

Proof. From Claim 2.2.1, it suffices to consider edges incident to at least one of $\{s, t, t_1, u_1, p_1, p_2, p_3, p_4, w, x, y\}$. One easily checks that none of these edges if labelled $(+,+)$, other than wx. This also implies the second statement. □

Note that Claim 2.2.2 allows us to conclude that M_s is stable, and that conditions (i) and (ii) of Theorem 1.3.5 are satisfied for M_h, M_g, M_m. In order to conclude the proof, we need to understand M-alternating paths in G_M. Recall that, in G_M, (,) edges are deleted. We started by investigating the location of (,) edges in G.

Claim 2.2.3. *If a positive clause contains a true literal ℓ, then $a(\ell), b(\ell) = (,)$ in M. If a negative clause contains a true literal ℓ', then $e(\ell')f(\ell') = (,)$ in M.*

Proof. Consider $H(\ell)$ and $N(\ell')$ as in the hypothesis of the claim. By construction, $b(\ell)c(\ell), d(\ell)e(\ell) \in M$ and $c(\ell')e(\ell'), b(\ell')d(\ell') \in M$. This implies that $a(\ell)b(\ell) = (,)$ and $e(\ell')f(\ell') = (,)$. □

Define an M-alternating path P in G_g, G_m, G_h to be *malicious* if it starts from an unmatched node and contains a $(+,+)$ edge. Observe that, by construction, the only

unmatched nodes are p_2 in G_g, s in $G_m; G_h$, and $h(\ell)$ in all graphs for all negative literals ℓ.

Claim 2.2.4. *For any clause c, there is no malicious path from s to wx in $G_m; G_h$ (from p_2 to wx in G_g) that is contained in $\{s; t; u; v; w; x; p_1; : : : ; p_4\} \cup \bigcup_{\ell \in c} V(\ell)$, and there is no M-augmenting path from s to v that is contained in $\{s; t; v\} \cup \bigcup_{\ell \in c} V(\ell)$.*

Proof. Since we have an assignment of the literals which satisfies , each clause contains at least one true literal. According to Claim 2.2.3, $a(\ell)b(\ell) = (\ ; \ )$ and $e(\ell')f(\ell') = (\ ; \ )$ for some literals ℓ and ℓ' in positive and negative clauses correspondingly. Since any path that satisfies the hypothesis with c a positive (resp. negative) clause must pass through edges $a(\ell)b(\ell)$ (resp. $e(\ell)f(\ell)$) for all $\ell \in c$, the thesis follows. □

Claim 2.2.5. *A malicious path starting at $h(\ell)$ does not traverse $f(\ell)$. In particular, there is no malicious path from $h(\ell)$ to wx that is contained in $\{s; t; t_1; u_1; p_1; p_2; p_3; p_4; u; v; w; x; y\} \cup \bigcup_{\ell \in c} V(\ell)$ for any negative clause c.*

Proof. $h(\ell)$ is unmatched and $e(\ell)f(\ell) \in M$ regardless of the true or false assignment to literal ℓ. Hence P cannot take both $h(\ell)e(\ell)$ and $e(\ell)f(\ell)$, since none of them is in M. Since $f(\ell)$ has degree 2, $f(\ell) \in P$. We conclude that there is no alternating path from $h(\ell)$ to wx contained only in c. □

Claim 2.2.6. *Let ℓ be a positive literal, and let P be a malicious path in $G_h; G_m$; or G_g or an M-augmenting path in G_s. Then $c(\ell)c(\ell')$ is an edge of P for at most one ℓ' and if that happens $f(\ell) \in P$.*

Proof. The first statement follows from the definition of path. Now assume there is ℓ' such that $c(\ell)c(\ell') \in P$. We omit dependency on ℓ in nodes. By construction, $cc(\ell') \in M$. Suppose by contradiction $f \in P$. Since f has degree 2, fd is an edge of P. We have two possibilities. First, if ℓ is true, then $\{bc; de\} \in M$. Then P or its

inverse (i.e. the path obtained traversing nodes from P in opposite order) contains the subpath $f; d; e; c; c(\ell')$, a contradiction to the fact that P is M-alternating. Second, if ℓ is false, then $bd; ce \in M$. Then P or its inverse contains the subpath $f; d; b; c; c(\ell')$, again a contradiction to the fact that it is M-alternating. □

Claim 2.2.7. *Let ℓ be a negative literal, and let P be a malicious path in $G_h; G_m$; or G_g or an M-augmenting path in G_s. Then $c(\ell)c(\ell')$ is an edge of P for at most one ℓ' and if that happens and $a(\ell) \neq u$, then $a(\ell) \in P$.*

Proof. Assume there is ℓ' such that $c(\ell)c(\ell') \in P$. Again, we omit explicit dependency on ℓ in nodes, and the first part of the statement is immediate. If $a \neq u$, then $\Gamma(a) = \{f, f(\ell''), b, c, g\}$ for some ℓ''. We have two possibilities. First, if ℓ is true, then $ce; bd \in M$. This implies that P or its inverse contains the subpath $c(\ell'); c; e$. Hence if $a \in P$, then ac is not an edge of P. Since $af(\ell'') \in M$ for some ℓ'' by construction and P is M-alternating, both ab and $bd \in M$ are edges of P. But then we must have that cd is an edge of P, a contradiction to P being a path. Second, if ℓ is false, then $cd; be \in M$. Hence, either P or its inverse contains the subpath $c(\ell')c; d; b; e$. Thus, $ac; ab$ are not edges of P, which implies that $a \notin P$. □

We can now conclude the proof of the lemma. First, we show that G_s has no M-augmenting path between s and v. By contradiction, assume there is such path P in G_s. Based on Claim 2.2.4 there is a consistency edge that belongs to P. Take the first consistency edge $c(\ell)c(\ell')$ that P traverses, and suppose that $c(\ell)$ is traversed by P before $c(\ell')$. This means that $a(\ell)$ and z are two consecutive nodes of P, traversed in this order and before $c(\ell)$, for some $z \in V(\ell)$ (possibly $z = c(\ell)$). Suppose first ℓ is a positive literal. Hence, $a(\ell)b(\ell)$ is an edge of P. By Claim 2.2.3, the variable corresponding to ℓ is set to false, hence $F(N(\ell')) \quad M$ by construction. Using again Claim 2.2.3, we deduce $f(\ell') \notin P$. Hence P, in order to reach v, must traverse $a(\ell')$ after $c(\ell')$. If $a(\ell') = u$, then P cannot traverse u again. Hence $a(\ell') \neq$

u and, by Claim 2.2.7, $a(\grave{}') \in P$. In both cases, we obtain a contradiction. If conversely $\grave{}$ is a negative literal, by Claim 2.2.7, $a = u(\grave{})$. If $F(N(\grave{})) \notin M$, then by construction $cd; be \in M$. Hence, P contains the subpath $u; c; d$ or the subpath $u; b; e; c; d$ (we omitted dependency on $\grave{}$). In both cases, $c(\grave{})c(\grave{}')$ cannot be an edge of P, a contradiction. If conversely $fce; bdg = T(N(\grave{})) \notin M$, then P contains the subpath $u; c; e$ or $u; b; d; c; e$ (again, we omitted the dependency on $\grave{}$). In both cases, $cc(\grave{}')$ cannot be an edge of P, a contradiction.

In order to deduce that M_m, M_h, M_g are popular, it suffices to show that condition (iii) of Theorem 1.3.5 is satisfied. Suppose, by contradiction, there exists a malicious path P starting from s in $G_m; G_h$, or from p_2 in G_g. Then $P = s; t; u; :::$ in $G_m; G_h$, or $P = p_2; p_4; p_3; p_1; s; t; u; :::$ in G_g. Claim 2.2.4 implies that there is at least one consistency edge that belongs to P. Take the first consistency edge $c(\grave{})c(\grave{}')$ that P traverses, and suppose that $c(\grave{})$ is traversed by P before $c(\grave{}')$. This means that $a(\grave{})$ and z are two consecutive nodes of P, traversed in this order and before $c(\grave{})$, for some $z \in V(\grave{})$ (possibly $z = c(\grave{})$). Suppose first $\grave{}$ is a positive literal. Hence, $a(\grave{}); b(\grave{}) \in P$. By Claim 2.2.3, $\grave{}$ is false, hence $\grave{}'$ is true. Using again Claim 2.2.3, we deduce $f(\grave{}') \notin P$. Hence P, in order to reach wx, must traverse $a(\grave{}')$ after $c(\grave{}')$. If $a(\grave{}') = u$, then P cannot traverse u again. Hence $a(\grave{}') \neq u$ and, by Claim 2.2.7, $a(\grave{}') \notin P$. In both cases, P cannot reach wx, a contradiction. If conversely $\grave{}$ is a negative literal, by Claim 2.2.7 $a = u(\grave{})$. If $\grave{}$ is set to true, then by construction $cd; be \in M$. Hence, P contains the subpath $u; b; c; d; e$ or the subpath $u; c; e$ (we omitted dependency on $\grave{}$). In both cases, $c(\grave{})c(\grave{}')$ cannot be an edge of P, a contradiction. If conversely $\grave{}$ is set to false, then P contains the subpath $u; b; e$ or $u; c; d$ (again, we omitted the dependency on $\grave{}$). In the latter case, $cc(\grave{}')$ cannot be an edge of P. In the former, the inverse of P must contain the subpath $c(\grave{}'); c; d; b$, a contradiction (b would have degree 3 in P).

Suppose now there exists a malicious path P starting from $h(\grave{}'')$. Since $tu \in M$,

$u \in P$. From Claim 2.2.5, P must traverse (in this order) a consistency edge $c(\ell')c(\ell)$ with ℓ' being a false literal. Pick the first such edge. By Claim 2.2.6, $f(\ell) \in P$. Hence, after traversing $c(\ell)$, P visits only gadgets corresponding to literals preceding ℓ in the clause, until it traverses (in this order) another consistency edge $c(\ell'')c(\ell''')$, since as we argued above, $u \in P$. This contradicts Claim 2.2.6, since it must also be that $f(\ell'') \notin P$.

Since s and the $h(\ell)$ with ℓ being any negative literal are the only unmatched vertices, we conclude that the condition (iii) is also satisfied, and M_s, M_h, M_g are popular. □

2.3 Proof of Theorems 2.1.1–2.1.3

We have now all facts needed to prove our first results.

Proof of Theorem 2.1.1. The problem is clearly in NP. Consider the graph $G = G_s$ constructed in Section 2.2 and the corresponding `Monotone 3-SAT` formula . Lemma 2.2.5 implies that, if G_s has a stable matching M with no M-augmenting path in G_M, then is satisfiable. Conversely, Lemma 2.2.6 implies that, if is satisfiable, then G_s contains a stable matching M with no augmenting path in G_M. The thesis then follows from Theorem 1.4.3.

Proof of Theorem 2.1.2. Suppose first $U^{in} = F^{in} = F^{out} = \emptyset$. Then the problem boils down to deciding if there exists a popular matching that does not contain a given set of nodes. By Lemma 1.3.3, it is enough to check if this is true for a(ny) stable matchings in $(G, \prec)$. It is well-known that this can be done in polynomial time, see e.g. [22].

Suppose now $U^{out} = F^{in} = F^{out} = \emptyset$. Then, again by Lemma 1.3.3, the problem corresponds to deciding if there exists a dominant matching of $(G, \prec)$ that contains a given set of nodes. This can be done efficiently due to Theorem 1.3.7.

Suppose now $U^{out} = U^{in} = F^{out} = ;$, and $F^{in} = feg$. This is the *popular edge problem*, that is shown in [13] to be solvable in polynomial time. We call the case when $U^{out} = U^{in} = F^{in} = ;$, and $F^{out} = feg$ the *unpopular edge problem*. We show that a solution to the latter follows from the solution to the popular edge problem [13]. Define E_s and $\bar{E}_s$ as:

$$E_s = fe \ 2 \ E : 9 \text{ stable matching } M \text{ s.t. } e \ 2 \ Mg$$

$$\bar{E}_s = fe \ 2 \ E : 9 \text{ stable matching } M \text{ s.t. } e \ 2 \ Mg$$

E_d, $\bar{E}_d$ (resp. E_p, $\bar{E}_p$) are defined similarly, by replacing "stable" with "dominant" (resp. popular). It is proved in [13] that $E_p = E_s \ [\ E_d$. We now argue that $\bar{E}_p = \bar{E}_s \ [\ \bar{E}_d$.

Consider any $e \ 2 \ \bar{E}_s \ [\ \bar{E}_d$. Since there is a stable or dominant matching that does not contain e, it follows that $e \ 2 \ \bar{E}_p$. We conclude that $\bar{E}_p \quad \bar{E}_s \ [\ \bar{E}_d$. Consider any $e = ij \ 2 \ \bar{E}_p$. There is a popular matching P s.t. $e \ 2 \ P$, and i or j is matched (if i and j are unmatched then $(i; j) = (+; +)$, and M is not popular). Wlog assume $jk \ 2 \ M$. It follows from $E_p = E_s \ [\ E_d$ that there exists a stable or dominant matching M^0 s.t. $jk \ 2 \ M^0$, hence $ij \ 2 \ M^0$. We conclude that $\bar{E}_p \quad \bar{E}_s \ [\ \bar{E}_d$.

Since $\bar{E}_p = \bar{E}_s \ [\ \bar{E}_d$, we can solve the forbidden edge problem by checking if $e \ 2 \ \bar{E}_s$ or $e \ 2 \ \bar{E}_d$. This can be done efficiently using Theorem 1.3.3 and Theorem 1.3.7.

We now move to NP-Complete cases. Testing whether a matching is popular can be performed in polynomial time, see Theorem 1.3.4. Hence all considered problems are in NP. Consider now a `Monotone 3-SAT` instance and the graph $G = G_m$ constructed as in Section 2.2. We claim that G admits a popular matching M with edges tu and xy if and only if is satisfiable. The *only if* case follows from Lemma 2.2.5. For the *if case*, recall that, in Section 2.2, starting from a feasible assignment to , we construct a matching M such that $tu; xy \ 2 \ M$ (by construction) and M is popular

(by Lemma 2.2.6).

Moreover, note that, for a popular matching M of G, the following are equivalent:

$$tu \in M \quad , \quad st \notin M \quad , \quad s \in V(M).$$

Similarly, the following are equivalent:

$$xy \in M \quad , \quad wx \notin M \quad , \quad y \in V(M).$$

Hence, we settled all NP-Complete cases with $|U^{in} \cup U^{out} \cup F^{in} \cup F^{out}| = 2$.

To show NP-Completeness when sets U^{out} and/or F^{out} have bigger size, we can add nodes $h(\ell)$ to U^{out} and/or evicted edges to F^{out}, since we know from Lemma 2.2.3 that those nodes and edges are never in a popular matching that contains tu. If U^{in} and/or F^{in} have bigger sizes, we can repeatedly add nodes z, k, with z adjacent to k and k adjacent to $g(\ell)$ for some literal ℓ such that z, k are each other's favorite partner. Then clearly zk belongs to any popular matching, so it can be added to F^{in}, and z to U^{in}.

Proof of Theorem 2.1.3. Because of the case $|F^{in}| = 2$, $U^{out} = U^{in} = F^{out} = \emptyset$ of Theorem 2.1.2, it is NP-hard to decide whether a graph has a popular matching with two given edges. Give those edges weight 1, and every other edge weight 0, and solve `mwp`. Theorem 2.1.2 implies that `mwp` cannot be solved within a factor better than 1=2.

Because of the case $|F^{out}| = 2$, $U^{out} = U^{in} = F^{in} = \emptyset$ of Theorem 2.1.2, it is NP-hard to decide whether a graph has a popular matching without two given edges. Give those edges weight 1, and every other edge weight 0, and solve `miwp`. Theorem 2.1.2 implies that it is NP-hard to decide if the optimum is 0, and the statement follows.

In order to prove the existence of a $\frac{1}{2}$-approximation algorithm for `mwp`, consider the following claim.

Claim 2.3.1. *Consider an instance $(G; <); w$ of mwp. Let M_{st} and M_{dom} be the maximum weight stable and dominant matchings of a respectively. Then $w(M)$ $2W$ for any popular matching M, where $W = \max(w(M_{st}); w(M_{dom}))$.*

Proof. Assume by contradiction that there exists a popular matching M such that $w(M) > 2W$. It was shown in [13] that any popular matching M can be partitioned into a stable matching M_0 restricted to a subgraph $G^0(\ )$ and a dominant matching $M_1 = M \cap M_0$ restricted to $G(\ ) \cap G^0(\ )$, and there always exist matchings M_0^0 and M_1^0 such that $M_{st}^0 = M_0 \cup M_1^0$ is stable and $M_{dom}^0 = M_0^0 \cup M_1$ is dominant in $G(\ )$. First assume $w(M_0)$ $w(M_1)$. This implies $w(M_0) = \frac{1}{2}(w(M_0) + w(M_0))$ $\frac{1}{2}(w(M_0) + w(M_1)) = \frac{1}{2}w(M)$. Consider $M_{st}^0 = M_0 \cup M_1^0$, $w(M_{st}^0) = w(M_0) + w(M_1^0)$. Since weights are non-negative, $w(M_{st}^0)$ $w(M_0)$ $\frac{1}{2}w(M)$. We conclude that W $w(M_{st}^0)$ $\frac{1}{2}w(M)$. This contradicts to the assumption that $w(M) > 2W$. If $w(M_1)$ $w(M_0)$, apply the same argument to M_{dom}^0. □

The 1=2-approximation immediately follows from the previous claim, and the fact that a dominant and stable matching of maximum weight can be computed in polynomial time, see Chapter 1.

Next corollary deals with weighted popular matching problems where weights are given on the *nodes* instead of the edges.

Corollary 2.3.1. *The following problem*

Problem 2.3.1 (`Node-weighted popular matching problem`). **Given:** *A marriage instance $(G; <)$, with weights w on nodes.* ***Find:*** *A popular matching M of $(G; <)$ that maximizes $w(M)$.*

is NP-Hard and not approximable to any factor in polynomial time, unless P=NP. Conversely, if w 0 or w 0, it can be solved in polynomial time.

Proof. Consider the instance $(G_m; <)$ as defined in Section 2.2, and w being the vector with 1 in the component corresponding to s, 1 in the components corresponding

to y, and 0 otherwise. As discusses in the proof of Theorem 2.1.2, it is NP-hard to decide if the optimum is strictly greater than 0. On the other hand, if $w \quad 0$ (resp. $w \quad 0$), Lemma 1.3.3 implies that the minimum (resp. maximum) of $w(M)$ with M popular is achieved at any stable (resp. dominant) matching, which can be easily computed, see again Chapter 1. □

The next problem can be seen as an extension of `pmffe` when F^{out} is the only non-empty set.

Corollary 2.3.2. *The following problem:*

Problem 2.3.2 (`Bounded intersection popular matching`). **Given:** *A marriage instance* $(G; <)$, $F \quad E(G)$ *and* q *2* N. ***Decide:*** *If there exists a popular matching* M *of* $(G; <)$ *such that* $jM \setminus Fj \quad q$.

is NP-Complete.

Proof. Clearly the problem is in NP. Consider the preference system $(G_m; <)$ as defined in Section 2.2. As in the proof of Theorem 2.1.2, add q pairs of nodes $k; z$, such that kz is in any popular matching. Let F be the set given by this q pairs, plus st, wx. Then a popular matching M satisfies $jM \setminus Fj \quad q$ if and only if $M \setminus fst; wxg = ;$ if and only if is satisfiable. □

Recall that testing whether a matching is popular can be performed in polynomial time, see Theorem 1.3.4. We conclude this section by showing that, on the other hand, deciding whether a subset of nodes can be matched among themselves only as to form a popular matching of the original instance is NP-Complete. We call this the `exclusive popular set problem.` It can also be seen as an extension of `pmfee` when U^{in} (or U^{out}) is the only non-empty set.

Corollary 2.3.3. *The following problem:*

Problem 2.3.3 (Exclusive popular set problem - eps)**. Given:** *A marriage instance* $(G; <)$*,* $U \quad V(G)$ *and* q *2* N*.* ***Decide:*** *If there exists a popular matching* M *of* $(G; <)$ *with* $V(M) = U$*.*

is NP-Complete.

Proof. Clearly the problem is in NP. Consider the instance $G = G_m$ from Section 2.2, and set $U = (ft; u; v; w; x; yg \; [\cdot \; V(`)) \; n \; [\cdot h(`)$, where the union ranges over all literals. Let M be a popular matching of $(G; <)$. We claim that $V(M) = U$ if and only if $st; wx \; 2 \; M$. The thesis then follows by the proof of Theorem 2.1.2. Let $V(M) = U$. Then by definition $st \; 2 \; M$, and $wx \; 2 \; M$, otherwise y is unmatched. Now suppose $st; wx \; 2 \; M$. Then $tu \; 2 \; M$. By Lemma 2.2.3, we know that for each literal $`$, $g(`)f(`) \; 2 \; M$, $h(`)$ is M-exposed, and $c(`)c(`^0) \; 2 \; M$ for all $`^0$. Simple arguments (see e.g. the proof of Lemma 2.2.3, part 3), imply that, for a fixed $`$, nodes $b(`); c(`); d(`); e(`)$ must be all matched, and all among themselves. This implies $V(M) = U$, as required. □

2.4 Proof of Theorems 2.1.4 and 2.1.5

Proof of Theorem 2.1.5. We start by proving NP-completeness of pmgg. pmgg belongs to NP because of Theorem 1.3.4. We refer again to the graph G_g from Section 2.2.

Claim 2.4.1. *Let* M *be popular of* $(G_g; <)$*. Then* $tu \; 2 \; M$ *and* $wx \; 2 \; M$*.*

Proof. Assume by contradiction that $tu \; 2 \; M$. $st \; 2 \; M$, otherwise s is unmatched and adjacent to $tu = (+; +)$. $p_1; p_2$ must be matched, otherwise there is an alternating path $u; t; s; p_1$ or $u; t; s; p_2$ respectively that contains $tu = (+; +)$ and unmatched node p_1 or p_2. This implies that $p_3p_4 \; 2 \; M$. In this case $t; u; s; p_2; p_4; p_3$ is an M-alternating path with $tu; p_4p_3 = (+; +)$, a contradiction to the popularity of M.

If $wx \; 2 \; M$, then $wy = (+; +)$ is adjacent to unmatched node y. □

Now suppose that the instance associated to G_g is satisfiable. Then, by Lemma 2.2.6, we conclude that G_g has a popular matching. Now let G_g has a popular matching M. By Claim 2.4.1, we conclude that M satisfies the hypothesis of Lemma 2.2.5, hence that is satisfiable. We conclude that pmgg is NP-complete.

In the remainder of the proof, we will use the modification of graph G_m to which edge vy has been added. We remark that all statements seen in Section 2.2 for G_m extend to this case. For simplicity of notation, we will still denote this new graph by G_m.

In order to conclude that mm (hm) is NP-complete, we cannot apply directly Lemma 2.2.5, since there are popular matchings that do not satisfy the hypothesis of the lemma.

For a popular matching M of $(G_m; <)$ $((G_h; <))$, we therefore argue in Claims 2.4.2 - 2.4.8 that if M is a popular matching such that $wx \not\in M$ and $tu \in M$, then M is a middle (hidden) matching, while all other popular matchings in the instance are stable or dominant.

All popular matchings of $(G; <)$ can be partitioned in the following four sets:

$M_1 = \{M$ is a popular matching : $tu \in M; xw \not\in M\}$,

$M_2 = \{M$ is a popular matching : $tu \in M; xw \in M\}$,

$M_3 = \{M$ is a popular matching : $tu \not\in M; xw \in M\}$,

$M_4 = \{M$ is a popular matching : $tu \not\in M; xw \not\in M\}$.

In Claims 2.4.2 - 2.4.4 we consider $(G_m; <)$ and $(G_h; <)$ simultaneously.

Claim 2.4.2. *Let $M \in M_1$. Then M is neither stable nor dominant.*

Proof. Observe that $wx = (+;+)$ and s is exposed. Since $wx = (+;+)$ then M is not stable. Consider a negative gadget $N(\grave{})$ such that $a(\grave{}) = u$. From Lemma 2.2.3 we know that either $c(\grave{})e(\grave{}); b(\grave{})d(\grave{}) \in M$ or $c(\grave{})d(\grave{}); b(\grave{})e(\grave{}) \in M$. In both cases

there is an augmenting path $s; t; u; c(\grave{}); e(\grave{}); h(\grave{})$ or $s; t; z; c(\grave{}); d(\grave{}); b(\grave{}); e(\grave{}); h$. This implies that M is not dominant by Theorem 1.3.6. □

Claim 2.4.3. *Let $M \in M_2$. Then M is stable.*

Proof. We show the statement for G_h, the one for G_m following analogously. From Corollary 2.2.1 it follows that M restricted to $G \setminus \{s; t; t_1; u_1; u; v; w; x; y\}$ does not contain any $(+;+)$ edges. From Lemma 2.2.3 we know that $f(\grave{})g(\grave{}) \in M$ for all $\grave{}$. This implies that $g(\grave{})v \notin M$. We conclude that $vy \in M$, otherwise $vy = (+;+)$ and $v; y$ are unmatched. Note that $st; vw; yx; tt_1; uu_1 = (+; \ )$. M does not contain $(+;+)$ edges, hence it is stable. □

Claim 2.4.4. *Let M be a popular matching and $tu \notin M$. Then*

1. *M does not contain $a(\grave{})b(\grave{})$ for any positive literal $\grave{}$;*
2. *M does not contain any consistency edges;*
3. *In each positive gadget $H(\grave{})$, exactly one of the following is true (dependency on $\grave{}$ is omitted):*

   a) $\{bd; ce\} \subseteq M$, *or*

   b) $\{bc; de\} \subseteq M$.

   Moreover, $f(\grave{})g(\grave{}) \in M$.
4. *If $f(\grave{})g(\grave{}) \in M$ for some negative clause $\grave{}$, then $e(\grave{}^0)h(\grave{}^0) \notin M$, for any literal $\grave{}^0$ that follows literal $\grave{}$ in the same negative clause;*

Proof. 1. Suppose by contradiction that $a(\grave{})b(\grave{}) \in M$ in some $H(\grave{})$, and let $\grave{}$ be the first literal in the clause that violates the thesis. If $a(\grave{}) \in u$, then $c(\grave{})$ must be matched. If $c(\grave{})e(\grave{}) \in M$, then $t; u; b(\grave{}); c(\grave{}); e(\grave{}); d(\grave{})$ is an alternating path with $tu; e(\grave{})d(\grave{}) = (+;+)$, a contradiction to popularity. If $c(\grave{})c(\grave{}^0) \in M$ for a negative literal $\grave{}^0$, then $d(\grave{}^0)$ must be matched. We conclude $d(\grave{}^0)b(\grave{}^0) \in M$. Similarly to above, $e(\grave{}^0)$ must be matched. Notice that $e(\grave{}^0)f(\grave{}^0) \notin M$, otherwise

$h(\ell')e(\ell')f(\ell')g(\ell')$ contains an unmatched node $h(\ell')$ and $f(\ell')g(\ell') = (+;+)$. This implies that $e(\ell')h(\ell') \in M$. In this case $t;u;b(\ell);c(\ell);c(\ell');d(\ell');b(\ell');e(\ell')$ contains $tu;b(\ell')e(\ell') = (+;+)$, a contradiction to popularity. If ℓ is not the first literal in the clause, let ℓ'' be the literal that immediately precedes ℓ in the clause. $f(\ell'')d(\ell'') \in M$, otherwise $f(\ell'')$ is unmatched, and $f(\ell'')g(\ell'') = (+;+)$. Similarly, we argue that $e(\ell'')c(\ell'');b(\ell'')a(\ell'') \in M$. This contradicts to the assumption that ℓ is the first literal in the clause such that $b(\ell)a(\ell) \in M$.

2. Assume that M contains a consistency edge, connecting a positive gadget $H(\ell)$ and a negative gadget $N(\ell')$. Edge $e(\ell)d(\ell) \in M$, otherwise $c(\ell)e(\ell) = (+;+)$ is adjacent to unmatched node $e(\ell)$. $a(\ell)b(\ell) \in M$, otherwise $b(\ell)d(\ell) = (+;+)$ and $b(\ell)$ is unmatched. This is a contradiction to part 1.

3. Consider a positive gadget $H(\ell)$. We omit the dependency on ℓ in nodes of $H(\ell)$. Suppose by contradiction that $cb;ce \notin M$. Recall from part 2 that M does not contain any consistency edges. This implies that c is unmatched and cb is a $(+;+)$ edge incident to an unmatched node b. If $db;de \notin M$, then $ec \in M$. Since $ab \notin M$ from part 1, then an alternating path $decb$ contains $de = (+;+)$ and unmatched node b. In both cases we obtain a contradiction to popularity. $fg \in M$, since d is matched to b or e, otherwise $fg = (+;+)$ and f is unmatched, which contradicts popularity.

4. Assume $f(\ell)g(\ell) \in M$ and $e(\ell')h(\ell') \in M$, for a literal ℓ' that follows ℓ. Node $b(\ell')$ must be matched. If $b(\ell')d(\ell') \in M$, then $c(\ell')a(\ell') \in M$ by part 2 and $f(\ell'')a(\ell') \notin M$. This contradicts to $f(\ell)g(\ell) \in M$, if $\ell'' \neq \ell$, or to popularity of M, since in the later case $e(\ell'')f(\ell'') \in M$, and $h(\ell'')$ is unmatched. If $b(\ell')a(\ell') \in M$ with a similar argument we deduce that $a(\ell');b(\ell');d(\ell');c(\ell')$ is an alternating cycle with a $(+;+)$ edge, a contradiction. □

Claim 2.4.5. *Let $M \in M_3 \cup M_4$ of $(G_m;\succ)$. Then M is dominant.*

Proof. We first show that there is exactly one evicted edge $e(\ell)h(\ell) \in M$. From Claim

2.4.4, part 1, we deduce that $M(u) \in N(\ell)$ for some ℓ. We omit the dependency on ℓ in nodes of $N(\ell)$. If $uc \in M$, then $db \in M$, otherwise $t; u; c; d$ is an alternating path with $tu = (+;+)$ and unmatched node d. Similarly, e is matched. If $ef \in M$, then $h; e; f; g$ is an alternating path with $fg = (+;+)$ and h being unmatched. We conclude that eh (hence fg) $\in M$. From Claim 2.4.4, part 4, we know that all other evicted edges in this clause are unmatched.

If $ub \in M$, then $cd \in M$, otherwise $t; u; b; d$ is an alternating path with $tu = (+;+)$ and unmatched node d. As above, e is matched, $ef \notin M$. This implies again that eh (hence fg) $\in M$.

Consider a different negative clause. Let ℓ' be its first literal. Similarly to above $e(\ell')$ must be matched, and $e(\ell')f(\ell') \notin M$. If $e(\ell')h(\ell') \in M$, then $b(\ell')d(\ell') \in M$, and $c(\ell')$ is unmatched, since M does not contain any consistency edges (Claim 2.4.4, part 2). We conclude that $c(\ell')e(\ell')$ is a $(+;+)$ edge incident to an unmatched node, a contradiction.

We now show that all unmatched nodes belong to the same bipartition class. From Claim 2.4.4, part 3, we know that for any positive clause $H(\ell)$, $b(\ell); c(\ell); e(\ell); d(\ell)$ are matched. This implies that $f(\ell)g(\ell) \in M$ for all positive gadgets. From what shown above, $b(\ell); c(\ell); d(\ell); e(\ell)$ are matched in all first literals for all negative clauses. In all other negative literals evicted edges are unmatched. This implies that $b(\ell); c(\ell); e(\ell); d(\ell)$ are matched in all gadgets and $f(\ell)g(\ell) \in M$ for all last literals ℓ of a negative clause. If $xw \in M$, then $vy \in M$; if $xw \notin M$, then $vw; xy \in M$. We conclude that $h(\ell)$ are the only unmatched nodes, and they belong to the same bipartition class, see Lemma 2.2.1. Hence, so there is no augmenting path. tu is a $(+;+)$ edge, so M is not stable. We conclude that M is dominant. □

Claim 2.4.6. *Let $M \in M_1$ of $(G_m; >)$. Then M is middle.*

Proof. From Claim 2.4.2 we know, that M is neither stable nor dominant. We will

show that there is a dominant matching of size larger than $|M|$. Consider a matching M' that contains the following edges:

$st; vy; wx$;

$a(\ell')c(\ell'); b(\ell')d(\ell'); e(\ell')h(\ell')$ for the first literal ℓ' in some negative clause such that such that $L(u; (c(\ell')) = 1$;

$f(\ell)g(\ell)$ for all gadgets;

all literals, except ℓ' (from above), are set to False: $b(\ell)d(\ell); c(\ell)e(\ell)$ for all positive gadgets $H(\ell)$, and $c(\ell)d(\ell); b(\ell)e(\ell)$ for all negative gadgets $H(\ell)$.

See Figure 2.8 for an example of such assignment. $vw; yx = (+; \ )$, and tu is the only $(+;+)$ edge. Vertices $h(\ell)$, for all negative gadgets $H(\ell)$, except $H(\ell')$, are the only unmatched nodes. M' is popular, since there is no alternating path between tu and $h(\ell)$. M' leaves unmatched only nodes from one set of the bipartition (nodes $h(\ell)$). On the other hand, M leaves s unmatched (which belongs to the other set of the bipartition, see Lemma 2.2.1), hence $|M| \quad |M'| \quad 1$. We conclude that M' is dominant, and $|M'| \quad |M| + 1$. □

Claim 2.4.7. *Let $M \ 2 \ M_3 \ [\ M_4$ of $(G_h; >)$. Then M is dominant and leaves unmatched exactly node s and nodes $h(\ell)$ for all literals ℓ.*

Proof. We first observe that $tt_1, uu_1 \ 2 \ M$. Suppose this is not the case. Then $t_1u_1 \ 2 \ M$ and $u; u_1; t_1; t$ is an alternating path containing two $(+;+)$ edges, a contradiction.

Consider first the first literal ℓ in some negative clause, omitting dependency on ℓ for nodes of $N(\ell)$. We claim that e is matched to either c or b. Suppose this is not the case. From Claim 2.4.4, part 2, we deduce that c is either unmatched, or matched to d. The former implies that e is incident to the $(+;+)$ edge ce, a contradiction. Hence the latter applies. Since u is matched to u_1, we deduce that b is unmatched and incident to the $(+;+)$ edge bd, a contradiction. Hence ec or $eb \ 2 \ M$. This implies:

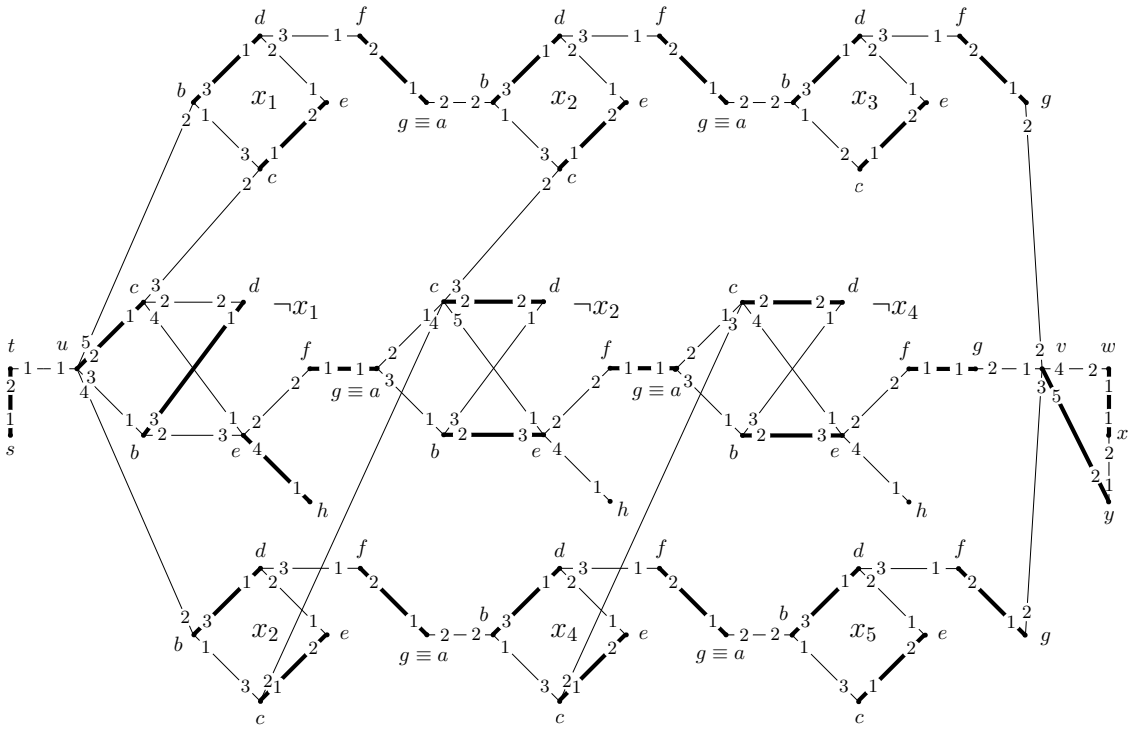

Figure 2.8: A dominant matching M' (bold edges) corresponding to an example from Fig. 2.4.

$eh; ef \in M$, hence $fg \in M$, thus (via Claim 2.4.4, part 4) $eh \in M$ for all literals ℓ' in the same clause as ℓ. We can therefore repeat the arguments above and deduce that, within each negative clause $N(\ell)$, nodes $b(\ell); c(\ell); d(\ell); e(\ell)$ are matched among themselves, and $f(\ell)g(\ell) \in M$. Because of Claim 2.4.4, part 3, the same apply to gadgets in positive clauses. This implies that v is not matched to any node $g(\ell)$.

Suppose first $wx \notin M$. Then $xy \in M$ (since x is the first choice of y), hence $vw \in M$. If conversely $wx \in M$, then $vy \in M$. We deduce that the only nodes left unmatched by M are s and $h(\ell)$, for all clauses ℓ. Note that nodes $h(\ell)$ belong to the same set of the bipartition (see Lemma 2.2.1). Moreover, the only nodes that can be reached from s via an M-alternating path in G_m are t and t_1, for $t_1; u_1$ is a $(\ ;\ )$ edge. Using Theorem 1.3.6, we deduce that M is dominant. □

Claim 2.4.8. *Let $M \in M_1$ of $(G_h; >)$. Then M is hidden.*

Proof. First, notice that all matchings from M_1 have the same size. This follows from the following. Since $tu \in M$ then $t_1u_1 \in M$. Similarly, $vw; xy \in M$. From Lemma 2.2.3, we conclude that s and $h(\grave{})$, for all $N(\grave{})$, are the only unmatched nodes in any $M \in M_1$.

We will show that a dominant matching has size $|M|$. Consider a matching M' such that contains the following edges:

$tt_1; uu_1; vy; wx$;

$f(\grave{})g(\grave{})$ for all gadgets;

All edges that correspond to False assignment of all literals: $b(\grave{})d(\grave{}); c(\grave{})e(\grave{})$ for all positive gadgets $H(\grave{})$, and $c(\grave{})d(\grave{}); b(\grave{})e(\grave{})$ for all negative gadgets $H(\grave{})$.

tu is the only $(+;+)$ edge. Vertices s and $h(\grave{})$ for all $\grave{}$ are the only unmatched vertices. Since $t_1u_1 = (\ ;\ )$ and $uu_1 \in M$, there is no alternating path from an unmatched node to $(+;+)$, we conclude that M' is popular. Notice that if M'' is popular, then $st \notin M''$. If $st \in M''$ then $tu \notin M''$ and $t_1u_1 \in M''$. In this case tt_1u_1u is an alternating path with $tt_1; uu_1 = (+;+)$. If $st \notin M''$, then either $tu; t_1u_1 \in M''$ or $tt_1; uu_1 \in M''$. Since all $h(\grave{})$ for all $\grave{}$ belong to the same bipartition class, Lemma 2.2.1, we conclude that M' is the largest popular matching. M' is dominant, since there is no augmenting path between s and $h(\grave{})$ for any $\grave{}$. $|M'| = |M|$. From Claim 2.4.2 we conclude that M is a hidden matching. □

Chapter 3

Popular matchings in graphs of bounded treewidth

In Chapter 2, we proved several hardness results on popular matchings. In particular, in the roommate case, even the problem of deciding if a popular matching exists is NP-Complete, see Theorem 2.1.5.

In this chapter, we show that polynomial-time solvability can be regained if we assume that the input graph has bounded treewidth. This applies with no other restriction on the input, i.e., we allow the graph to be non-bipartite. More precisely, we prove the following.

Theorem 3.0.1. *Let $!$ be a fixed constant. There exists a function $f(n) = O(n^{3!+7})$ such that, given as input a graph G with n nodes with strict preference lists $\prec$, treewidth at most $!$, and a cost vector c on the edges of $(G, \prec)$, in time $f(n)$ we can either conclude that $(G, \prec)$ has no popular matching, or find a popular matching of $(G, \prec)$ of minimum cost.*

3.1 Introduction

As discussed in Chapter 1, bounded treewidth is a classical assumption that often turns intractable problems into tractable ones. A typical example is Maximum Independent (Stable) Set (`MIS`), for which a polynomial-time algorithm exists in bounded treewidth graphs [9]. `MIS` enjoys two nice properties, often shared by problems for which the bounded treewidth approach is successful. The first is *monotonicity*: if S is an independent set in a graph $G(V, E)$ and G' is a subgraph of G, then the

solution induced by S on G' is also feasible. The second is *locality*: in order to check if S is an independent set, it suffices to verify, for each node of S, if any node of its neighborhood also belongs to S.

Interestingly, similar properties do not hold for popular matchings. Indeed, popularity is not a local condition, since it may depend on how nodes far away in the graph are matched. Moreover, if we take a graph G and a popular matching M, the subset of M contained in an induced subgraph of G may not be popular. Examples with both those features can be easily constructed by building on the characterization of popular matchings given by Theorem 1.3.5.

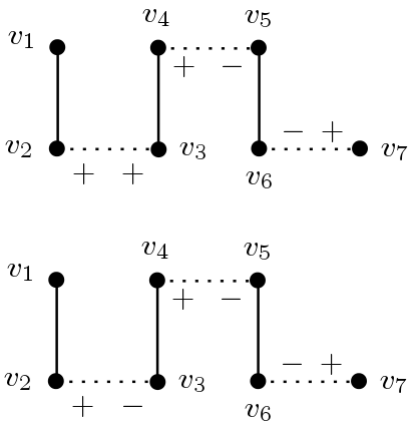

Figure 3.1: An example of popular vs locally-popular. In each graph, let M be the matching given by bold edges. *Top graph.* M is not popular, since there is an M-alternating path in G_M with a $(+;+)$ edge and an unmatched node. But if we take $X = \{v_1; v_2\}$ and $S = \{v_3; v_4\}$, matching $M_{X \cup S}$ is $(S; X)$-locally popular. *Bottom graph.* M is popular, hence it is $(X; S)$-locally popular for every choice of S and X. But if we restrict M to the subgraph G' of G induced by $v_1; v_2; v_3; v_4; v_5$, the matching we obtain is not popular in G', since there is a $(+;+)$ edge incident to the now unmatched node v_5.

Our technique to prove the tractability of the minimum cost popular matching problem in the bounded treewidth case is as follows. We assume wlog that all matchings have different costs. This can be achieved efficiently by standard perturbation techniques. Given a vertex separator S of G and a connected component X of $G[V \setminus S]$, we define the family of $(S; X)$-*locally popular* matchings. This contains all match-

ings that may potentially be extended to a popular matching in the whole graph by adding edges not incident to $X \cup S$ (a formal definition is given in Section 3.1). As $(S; X)$-locally popular matchings can be exponentially many even in graphs of bounded treewidth, we cannot store all of them. Instead, we divide them in classes (which we call *tipping points*), and show that, if a matching M of a class can be completed to a $(S'; X')$-locally popular matching for some sets with $X' \supseteq X$ and $X' \cup S' \supseteq X \cup S$ by adding some matching M' not incident to $X \cup S$, then all matchings of the same class can be extended via the same matching M' to a $(S'; X')$-locally popular matching. Hence, it is enough to keep only a representative for each class — the one of minimum cost, which we call the *leader*. Finally, we show how to iteratively construct tipping points and their leaders by building on a tree decomposition of the graph (see Algorithm 3). In particular, if the graph has bounded treewidth, a tree decomposition of bounded width can be computed in linear time [10], there will be only polynomially many tipping points and leaders, and they can be built in polynomial time, see Theorem 3.3.1.

Additional definitions

Recall that our input graph is $G = (V; E)$. For $S \subseteq V$, we denote by $G \setminus S$ the subgraph of G induced by $V \setminus S$. If $G \setminus S$ is disconnected, S is said to be a *vertex separator*. Given a matching M on G and a set $U \subseteq V$, the matching *induced* by M on U is given by $E[U] \cap M$, and it is denoted by $M[U]$.

Now also fix $U \subseteq V$. Given a matching M of G, we say that M is a *U-matching* if for all edges $(u; v) \in M$, we have $\{u; v\} \cap U \neq \emptyset$. Given a matching M of G, the *U-matching induced by M* is the set $M_U = \{(u; v) \in M : \{u; v\} \cap U \neq \emptyset\}$.

Again, since all graphs we deal with are simple, we represent paths and cycles as ordered set of nodes, with the first node of a cycle coinciding with the last. This also allows us to distinguish between the *first* and *last* node of a path. We will say

that *e is an edge of P* if it is an edge between two consecutive nodes of P. A U-path P is a path in G where either the first or the last node of P (or possibly both) belongs to U, and all other vertices of P do not lie in U (i.e., if $P = (v_1; : : : ; v_k)$, then $; \neq P \setminus U \quad \{v_1; v_k\}$).

Two paths $P = (v_1; : : : ; v_k)$, $P' = (v'_1; : : : ; v'_{k'})$ are called *U-disjoint* if $P \setminus P' \setminus U = ;$. P, P' are said to be *internally vertex-disjoint* if, for all $i \in [k]$ and $j \in [k']$, $v_i \neq v'_j$ with possibly the exception of $(i; j) \in \{(1; 1); (1; k'); (k; 1); (k; k')\}$. Consider paths $P_1; P_2; : : : ; P_q$ of G, $q \quad 2$, whose union contains at least two distinct vertices and with the following properties:

- $P_i \setminus P_j = ;$, unless $ji \quad jj \quad 1$ or $i; j \in \{1; q\}$;
- If $q \quad 3$, for $i = 1; : : : ; q \quad 1$, the last node of P_i is the first node of P_{i+1}, and $P_i; P_{i+1}$ are otherwise disjoint. Moreover, P_1 and P_q are disjoint, with possibly the exception of the last node of P_q coinciding with the first node of P_1;
- If $q = 2$, the last node of P_1 coincides with the first node of P_2, the last node of P_2 may coincide with the first node of P_1, and paths P_1 and P_2 are otherwise disjoint.

The *juxtaposition* of $P_1; : : : ; P_q$ is the path (if the first node of P_1 and the last of P_q are different) or cycle (otherwise) defined from $(P_1; P_2; : : : ; P_q)$ as above by removing consecutive repeated vertices.

Recall that we assume that no two matchings of G have the same cost.

Locally popular matchings, Configurations, and Tipping points

In this section, we fix a graph $G = (V; E)$ together with strict rankings of the neighbors of each node in V, a vertex separator S of G, and a connected component X of $G \setminus S$ (possibly $S = ;$ and $X = V$).

Let M be a $(X \cup S)$-matching of G. We will extensively work with graph $G_M[X \cup S]$ – that is, the subgraph of G_M induced by $X \cup S$. For a node v in S that is matched in M to a neighbor outside $X \cup S$, labels of $G_M[X \cup S]$ incident to v are a function of the edge $(v; M(v))$, however note that the edge $(v; M(v))$ is not in $G_M[X \cup S]$.

We say that M is $(S; X)$-*locally popular* if none of the structures (i), (ii), and (iii) from Theorem 1.3.5 is a subgraph of $G_M[X \cup S]$. We remark that a node that is not matched in $G_M[X \cup S]$ may still be M-covered (hence not M-exposed). If this happens, such a node cannot be the M-exposed node in the path (iii) from Theorem 1.3.5. The following simple proposition relates the definitions of popularity and $(S; X)$-local popularity.

Lemma 3.1.1. *Let $G; S; X$ be as above, S' be a vertex separator of G, and X' be one of the connected components of $G \setminus S'$ such that $X' \cup S' \supseteq X \cup S$. Then:*

1. *Let M be a matching in G. Then M is $(;; V)$-locally popular if and only if it is popular.*
2. *Let M' be a $(S'; X')$-locally popular matching. Let $M := M' \cup_{X \cup S}$. Then M is an $(S; X)$-locally popular matching.*

Proof. 1. It follows by definition and from the fact that $G_M[; \cup V] = G_M$.

2. Suppose M is not $(S; X)$-locally popular. Then, one of the structures from Theorem 1.3.5 is a subgraph of $G_M[X \cup S]$ — call it P. We claim that P is a forbidden structure in $G_{M'}[X' \cup S']$ (again, in the sense of Theorem 1.3.5), hence M' is not $(S'; X')$-locally popular, a contradiction.

Indeed, $X \cup S \supseteq X' \cup S'$, and none of the edges of $M' \setminus M$ is incident to $X \cup S$ by definition. We deduce $G_M[X \cup S] = G_{M'}[X \cup S]$ and a node of $X \cup S$ is M-exposed if and only if it is M'-exposed. This concludes the proof. □

We now introduce the concepts of *con guration* and *tipping point* in order to partition the family of $(S; X)$-locally popular matchings into a (small) number of

classes. These definitions are related to the existence of certain structures that are not forbidden in a popular matching, but restrict the capability of extending locally popular matchings to popular matchings of the whole graph.

From now on, we also fix M to be a $(X \cup S)$-matching of G. Let $\mathcal{P}_M$ be the set of M-alternating paths of $G_M[X \cup S]$. We associate to each $P \in \mathcal{P}_M$ a 2-dimensional *parity* vector , where the first component of is defined to be 0 if the first edge of P is a matching edge, 1 otherwise. Similarly, the second component of the parity vector takes values 0 or 1, depending on whether the last edge of the path is a matching edge or not. We also associate to P a two-dimensional *level* vector $\ell = (e, p)$, where $e \in \{0, 1, 2\}$ is the number of M-exposed nodes of P, and $p \in \{0, 1\}$ denotes the number of $(+,+)$ edges of P. If $p \geq 2$, then we say that P is of level *1*. Note that not all parity-level combinations are possible. Moreover, if P is of level *1*, then M is not (S, X)-locally popular. Let

$$U = ((u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)), \tag{3.1}$$

for some $k \in \mathbb{N}$, where $u_1, v_1, u_2, \ldots, v_k \in S \cup \{\}$, under the additional condition that $u_i \neq v_i$ for all $i \in [k]$, all pairs are different, and each node of S can appear at most twice in the collection. Moreover, let $L = (\ell_1, \ldots, \ell_k)$, $\Pi = (\pi_1, \pi_2, \ldots, \pi_k)$ and, for $i \in [k]$, $\ell_i \in \{0, 1, 2\} \times \{0, 1\}$ and $\pi_i \in \{0, 1\} \times \{0, 1\}$. The triple $C = (U, L, \Pi)$ is called an (S, X)-*con guration*.

We say that M is *active at* C if there exist pairwise X-disjoint S-paths $P^1, \ldots, P^k \in \mathcal{P}_M$, such that, for $i \in [k]$, P^i: is of level ℓ_i and parity π_i; starts at $u_i \in S$ if $u_i \neq$; and at some node of X otherwise; ends at $v_i \in S$ if $v_i \neq$; and at some node of X otherwise. We call those paths the *certi cate of* C *at* M[1].

[1]Note that, if M is active at C, it is also active at the con gurations e.g. obtained by permuting entries of U (and of L and accordingly). This causes some redundancy, yet this will not a ect our analysis, so we do not eliminate it.

Let $\{(U_i, L_i, \Pi_i)\}_{i=1,\ldots,q}$ be the collection of all (S, X)-configurations at which M is active. We call

$$(\{(U_i, L_i, \Pi_i)\}_{i=1,\ldots,q}, M_S)$$

the *(S, X)-tipping point of M*.

Lemma 3.1.2. *Let G, X, S be as above. Let M be an (S, X)-locally popular matching. The (S, X)-tipping point of M is uniquely defined. On the other hand, there exists a function $g : \mathbb{N} \to \mathbb{N}$ such that the number of (S, X)-configurations is bounded by $g(|S|)$, and the collection of (S, X)-tipping points of M, where M ranges over all (S, X)-locally popular matchings, has size at most $g(|S|)|V|^{|S|}$.*

Proof. The first statement follows by definition. For the second and third: the number of (S, X)-configurations (U, L, Π) is a function of the size of S only, since all pairs from U are different, and each node of S can appear in at most two pairs from U. Moreover, the number of S-matchings of G is upper bounded by $|V|^{|S|}$. □

Lemma 3.1.3. *Let G, X, S be as above. Let M be an (S, X)-locally popular matching and T be the (S, X)-tipping point of M. Let N be another (S, X)-locally popular matching whose (S, X)-tipping point is also T. Let $e \in \delta(v)$ for some $v \in S$. Then:*

1. *$e \in M$ if and only if $e \in N$.*
2. *Suppose $e \in M$ and let $? \in \{+, -\}$ be the label of e at v wrt M. Then $e \in N$ and $?$ is also the label of e at v wrt N.*

Proof. Both statements immediately follow from $M_S = N_S$, which holds by definition of tipping point. □

3.2 Leaders

Fix G, S, X, M as in the previous section, and let T be the (S, X)-tipping point of M. M is said to be the *T-leader* if it is the one of minimum cost among all (S, X)-locally

popular matchings whose $(S; X)$-tipping point is T. Due to the initial perturbation of costs, note that there is at most one T-leader for each tipping point T.

The following crucial lemma shows that, in order to find a min cost popular matching, it suffices to consider matchings that induce $(X \cup S)$-matchings that are T-leaders, for some $(S; X)$-tipping point T. The proof of Lemma 3.2.1 is given in Section 3.4.

Lemma 3.2.1. *Suppose we are given $G; X; S; M$ as above. Let T be the $(S; X)$-tipping point of M, and assume that M is not the T-leader. Let S' be a vertex separator of G, X' a connected component of $G \setminus S'$ with the property that $X' \subseteq X$ and $S' \cup X' \subseteq S \cup X$ (possibly $S' = \emptyset$ and $X' = V$). Let M' be a $(S'; X')$-locally popular matching such that $M'_{X \cup S} = M$. Let T' be the $(S'; X')$-tipping point of M'. Then M' is not a T'-leader.*

3.3 The algorithm

We now give our algorithm for computing a minimum weight popular matching, see Algorithm 3. Note that Algorithm 3 relies on the subroutine `Update` described in Algorithm 4, and the implementations of some other subroutines are not completely defined. We give a formal description of those together with a complexity analysis in the proof of Theorem 3.3.1.

Algorithm 3 takes as input a graph G with strict preference lists as usual, and a dichotomic directed tree decomposition $(T; B)$ of G (see Section 1.5). It iteratively constructs the sequence of closed subtrees T_B of T, with the first T_B corresponding to a leaf B of T, and the last to the root. For each T_B, let $S = B \setminus S(B)$ and $X = V(T_B) \setminus S$. The algorithm constructs and stores in L_B all the pairs $(M; T)$, where T is an $(S; X)$-tipping point and M is the T-leader. This set can be found by building on the corresponding sets for the (at most two) predecessors of B. Finally,

of all matchings M such that $(M;T)\ 2\ L_B$ – with B being the root – the one of minimum cost is output. If at the end of any iteration we have $L_B = ;$ for the current B, we deduce that G has no popular matching.

With a little abuse of notation, we will write $M\ 2\ L_B$ if $(M;T)\ 2\ L_B$ for some T. Similarly, we write $L_B = L_B\ [\ fMg$ to mean that $(M;T)$ is added to L_B for an appropriate T, and similarly for $L_B\ n\ fMg$.

Algorithm 3:

Require: A graph G, together with, for each node $v\ 2\ V$, a strict ranking of the neighbors of v. A dichotomic directed tree decomposition $(T;B)$ of G.
Ensure: A popular matching of minimum cost in G.

1: For all $B\ 2\ B$, label B as $un\ agged$.
2: Choose an unflagged bag B whose predecessors are flagged, and flag B.
3: Let T_B be the closed subtree of T whose head is B, and set $L_B = ;$.
4: Let $S = B \setminus S(B)$, $X = V(T_B)\ n\ S$.
5: **if** B has no predecessor in T **then**
6: **for** all B-matchings M of G **do**
7: **if** M is an $(S;X)$-locally popular matching **then**
8: Update$(M\ , L_B)$
9: **end if**
10: **end for**
11: **else**
12: Let $S_1 = B \setminus B_1$ and (possibly) $S_2 = B \setminus B_2$, where B_1 and (possibly) B_2 are the predecessors of B.
13: **for** all B-matchings M of G, all $M_1\ 2\ L_{B_1}$ and (possibly) $M_2\ 2\ L_{B_2}$ **do**
14: **if** $(M_1)_{S_1} = M_{S_1}$ and (possibly) $(M_2)_{S_2} = M_{S_2}$ **then**
15: Let $M = M\ [\ M_1\ [\ M_2$.
16: **if** M is an $(S;X)$-locally popular matching of G **then**
17: Update$(M\ , L_B)$
18: **end if**
19: **end if**
20: **end for**
21: **end if**
22: **if** $L_B = ;$ **then**
23: output: G has no popular matching.
24: **else if** there is a bag $B\ 2\ B$ that is unflagged **then**
25: Go to Step 2.
26: **end if**
27: Let B be the head of T. Output the matching of minimum cost from L_B.

Algorithm 4: `Update`

Require: M , L_B
  Let T be the $(S; X)$-tipping point of M .
  if there exists $M' \in L_B$ whose tipping point is T **then**
    if $c(M) < c(M')$ **then**
      Set $L_B = L_B \setminus \{M'\} \cup \{M\}$.
    end if
  else
    Set $L_B = L_B \cup \{M\}$.
  end if

Analysis

The goal of this section is to prove the following result.

Theorem 3.3.1. *Algorithm 3 is correct. If the treewidth of $G = (V; E)$ is upper bounded by a constant ℓ, then it can be implemented to run in time $O(|V|^{3\ell+7})$.*

We assume throughout the proof that every bag B that is not a leaf has two predecessors, as the (simpler) case where some B has only one predecessor follows in a similar fashion. For a bag B we write $S := B \setminus S(B)$ and $X := V(T_B) \cap S$; if B is not a leaf, we denote by B_1 and B_2 its predecessors, and write $S_i := B \setminus B_i$, $X_i = V(T_{B_i}) \cap S_i$ for $i = 1, 2$. We start by proving the correctness of Algorithm 3. We show that, at the end of the iteration where bag B is flagged,

( ) L_B contains exactly all T-leaders, for all $(S; X)$-tipping points T for which a T-leader exists.

Suppose ( ) is proved. Then, when B is the root, L_B contains only popular matchings (by Lemma 3.1.1), and the one of minimum cost among those is the popular matching of minimum cost. Again by Lemma 3.1.1, if $L_B = \emptyset$ at the end of the iteration where B is flagged, then G has no popular matching, and the output of Algorithm 3 is again correct.

The proof of ( ) is by induction on the number of nodes n_B of T_B. If $n_B = 1$, then the condition from the **if** statement in Step 5 is verified. In this case, the statement is immediate, since we enumerate over all possible B-matchings of G, check those that are $(S; X)$-locally popular, and for each $(S; X)$-tipping point T, keep the $(S; X)$-locally popular matching of minimum cost active at T.

Now suppose $n_B > 1$. By induction hypothesis, for $i = 1; 2$, all matchings that are stored in L_{B_i} are exactly all T-leaders, for all $(S_i; X_i)$-tipping points T for which a T-leader exists (i.e., there is at least a $(S_i; X_i)$-locally popular matching active at T). Now let T be an $(S; X)$-tipping point for which a T-leader $\hat{M}$ exists. It suffices to show that one of the matchings M constructed at Step 15 is indeed $\hat{M}$.

Since $\hat{M}$ is $(S; X)$-locally popular and $X_i \cap S_i \quad X \cap S$, matching $M_i := \hat{M}_{X_i \cap S_i}$ is also $(S_i; X_i)$-locally popular by Lemma 3.1.1. By Lemma 3.2.1, M_i is a $(S_i; X_i)$-leader. By induction, $M_i \in L_{B_i}$. On the other hand, $M := \hat{M}_B$ is a B-matching of G such that $M_{S_i} = (M_i)_{S_i}$ for $i = 1; 2$. Since we enumerate all B-matchings of G, as well as all matchings from L_{B_1} and L_{B_2}, matching $\hat{M}$ is eventually enumerated.

Running time analysis. We now bound the running time of Algorithm 3. We will use the following general fact proved in the claim below: if we are given an $(X \cap S)$-matching M in a graph H and $|X \cap S|$ is bounded, then one can check efficiently $(S; X)$-local popularity of M. If M is $(S; X)$-locally popular, then we can efficiently find the $(S; X)$-tipping point at which M is active.

Claim 3.3.1. *Let $H = (U; F)$ be a graph, S a vertex separator of H, X a connected component of $H \setminus S$. Let M be a $(X \cap S)$-matching of H. Assume $|X \cap S|$ is upper bounded by a constant. Then in $O(|F|)$ time one can:*

1. *check if M is $(S; X)$-locally popular in H and, if it is,*
2. *find the $(S; X)$-tipping point (in H) at which M is active.*

Proof. Building graph $H_M[X \cap S]$ takes $O(|F|)$ time. Since $|X \cap S|$ is upper bounded

by a constant, it takes constant time to enumerate all paths and cycles of $H_M[X \cup S]$, and to check if any of these violates the definition of $(S; X)$-local popularity. In case M is indeed $(S; X)$-locally popular, for any family of paths P in $H_M[X \cup S]$ and $(S; X)$-configurations C, one can check in constant time if P is a certificate for M at C. Since by Lemma 3.1.2 the number of $(S; X)$-configurations is upper bounded by $g(\ell + 1)$, which is a constant by hypothesis, the claim follows. □

Observe that $|B| \leq \ell + 1$, so enumerating all B-matchings takes time $O(|V|^{\ell+1})$. Because of Claim 3.3.1 and Step 7 (in Algorithm 3), finding the tipping point of any B-matching M can be performed in time $O(|E|) = O(|V|^2)$. Moreover, once the tipping point of M is computed, the `Update` function can be implemented to run in time $O(|V|^{\ell+1})$, since $\mathcal{L}_B$ at each step will contain at most one matching per tipping point, and by Lemma 3.1.2, there are at most $g(\ell + 1)|V|^{\ell+1}$ many tipping points.

We conclude that, when $n_B = 1$, the iteration when B is flagged runs in time $O(|V|^{2\ell+2})$. We now prove, by induction on n_B, that the iteration when any B is flagged can also be implemented to run in time $O(|V|^{3\ell+5})$, the base case having been just proved. Multiplying this by $O(|V|^2)$ (the number of bags of the tree decomposition), we obtain the desired bound.

Assume $n_B > 1$. Because of what was discussed above, given an $(S; X)$-configuration C and a matching $M = M \cup M_1 \cup M_2$, where M is a B-matching of G and for $i = 1, 2$ $M_i \in \mathcal{L}_{B_i}$, it is enough to give an upper bound on the time needed to decide if M is $(S; X)$-locally popular and if it is active at C. Recall that the number of B-matchings is $O(|V|^{\ell+1})$, while $|\mathcal{L}_{B_i}| = O(|V|^{\ell+1})$ and the number of $(S; X)$-configuration is at most $g(\ell + 1)$ by Lemma 3.1.2.

The condition from Step 14 can be verified in time $O(|E|)$. Hence, we assume that $(M_1)_{S_1} = M_{S_1}$ and $(M_2)_{S_2} = M_{S_2}$. We start with a simple claim.

Claim 3.3.2. *For $i = 1, 2$, let $C_i = (U_i; L_i; \Pi_i)$ be a configuration at which M_i is*

active, and let P_i be a certi cate of C_i at M_i. Then all paths from $P_1; P_2$ are pairwise internally vertex-disjoint.

Proof. By definition of certificate, paths from P_1 are pairwise internally vertex-disjoint, and similarly so are paths from P_2. Now let $P_1 \ 2 \ P_1$, $P_2 \ 2 \ P_2$. For $i = 1; 2$, $P_i \quad V(T_{B_i})$ and P_i is an S_i-path. Since $V(T_{B_1}) \setminus V(T_{B_2}) \quad S_1 \ [\ S_2$, the claim follows. □

Now fix an $(S_1; X_1)$-configuration $C_1 = (U_1; L_1; \Pi_1)$ in G at which M_1 is active, and an $(S_2; X_2)$-configuration $C_2 = (U_2; L_2; \Pi_2)$ in G at which M_2 is active. Consider the graph H and matching M_H obtained as follows. Start from $H = G_M[B]$ and the corresponding matching $M_H = M[B]$, and let $(u; v)$ be the first pair from U_1. Assume $u; v$ are matched by M_H, $`_1 = (0; 1)$, $_1 = (0; 0)$, the other cases following in a similar fashion. Add to H and M_H new nodes $u'; w'; z'; v'$, matching edges $(u'; w')$, $(z'; v')$, $(+; +)$ edge $(w'; z')$, and $(+; \ )$ edges $(u; u')$ and $(v; v')$. Note that $(u; u'; w'; z'; v'; v)$ is a M_H-alternating S_1-path of parity $_1$ and level $`_1$ starting at u and ending at v. We call it the *shortcut* of the path with endpoints $u; v$. Repeat this for all pairs from U_1 and U_2, adding at each time new nodes and an appropriate path. Note that this adds to H a bounded number of nodes. This means that the graph H and matching M_H can be constructed in time $O(jEj)$.

Claim 3.3.3. *Let M be an $(S \ [\ X)$-matching. M is not an $(S; X)$-locally popular matching in G if and only if there exist an $(S_i; X_i)$-con guration C_i at which M_i is active for $i = 1; 2$, such that, if we construct graph H and matching M_H as above, then M_H is not $(S; V(H) \ n \ S)$-locally popular in H.*

Proof. Consider the forbidden subgraph P (from Theorem 1.3.5) of G_M $[X \ [\ S]$, whose existence certifies that M is not $(S; X)$-locally popular in G, and take P that is minimal with this property. Using an immediate modification to Algorithm 5 (given in Section 3.4), we can write P as the juxtaposition of $(P_1; P_2; : : : ; P_k)$ where each P_j

is: either an S_i-path contained in $G[X_i \cup S_i]$ for $i \in \{1,2\}$, or a path contained in $G[B]$. Collect all such S_1-paths from $(P_1,\ldots,P_k)$ not contained in $G[B]$ in P_1, and all S_2-paths from $(P_1,\ldots,P_k)$ not contained in $G[B]$ in P_2. One easily checks that P_i form the certificate of a certain (S_i, X_i)-configuration C_i at M_i.

Now consider the graph H and matching M_H obtained from configurations C_1 and C_2. By replacing each path from P_1 and P_2 with the corresponding shortcut, we deduce that M_H is not an $(S, V(H) \setminus S)$-locally popular matching in H.

The opposite direction follows in a similar (inverse) fashion: start from a forbidden path in H that certifies that M_H is not $(S, V(H) \setminus S)$-locally popular in H, write it as $(P_1,\ldots,\ldots,P_k)$, replace each of those subpaths with the appropriate certificate paths so as to obtain an M -alternating forbidden path by Claim 3.3.2. □

The proof of the following claim follows in a similar fashion to the proof of Claim 3.3.3.

Claim 3.3.4. *Let M be an (S, X)-locally popular matching in G and let C be an (S, X)-configuration. Then M is active at C if and only if there exist a (S_i, X_i)-configuration C_i at which M_i is active for $i = 1, 2$ such that if we construct graph H and matching M_H, then M_H is active at C (here we interpret C as an $(S, V(H) \setminus S)$-configuration in H, which is allowed since $S \subseteq V(H)$).*

Because of Claim 3.3.3, we can check if M is an (S, X)-locally popular matching in G by checking, for each pair of (S_i, X_i)-configurations C_i at which M_i is active for $i = 1, 2$, if the corresponding M_H is $(S, V(H) \setminus S)$-locally popular in H. For $i = 1, 2$, the number of (S_i, X_i)-configurations is at most $g(k+1)$. Since the number of nodes of H is bounded (we start with a bounded set of nodes and we add a bounded number of new nodes), testing if M_H is $(S, V(H) \setminus S)$-locally popular in H can be done in time $O(|E|) = O(|V|^2)$ by Claim 3.3.1. Similarly, if M is (S, X)-locally popular in G, we can find its tipping point in time $O(|V|^2)$ by repeatedly applying Claims 3.3.4 and

3.3.1. Hence, the iteration when B is flagged can be performed in time $O(|V|^{3!+5})$. This concludes the proof of Theorem 3.3.1.

3.4 Proof of Lemma 3.2.1

Let N be the $(S; X)$-locally popular matching that is the T-leader. This implies that $M_S = N_S$, and N is a $(X \cup S)$-matching. Define $N' := N \oplus (M' \ominus M)$, and notice this is a disjoint union. Hence, $N'_{X \cup S} = N$. We will now show that N' is a $(S'; X')$-locally popular matching whose $(S' \cup X')$-tipping point is T'. We then have:

$$c(N') = c(N) + c(N' \ominus N) < c(M) + c(M' \ominus M) = c(M');$$

concluding the proof. We start with some claims, the first two of which immediately follow from construction.

Claim 3.4.1. *$G_{M'}[X \cup S] = G_M[X \cup S]$ and $G_{N'}[X \cup S] = G_N[X \cup S]$.*

Claim 3.4.2. *$M' = M \oplus (N' \ominus N)$, and the latter is a disjoint union.*

Claim 3.4.3. *Let $e \in E$ not be incident to a node of X.*

a) $e \in N'$ if and only if $e \in M'$.

b) Let $e \in N'$. Then $e \in E(G_{N'})$ if and only if $e \in E(G_{M'})$. Moreover, if $e \in E(G_{N'})$, then it has the same labels in $G_{N'}$ and in $G_{M'}$.

Proof. a) Suppose $e \in N'$. If $e \in N$, then $e \in N_S = M_S \subseteq M \subseteq M'$, where equality holds by hypothesis and inclusions by definition. Else, $e \in (N' \ominus N) \subseteq M'$. The argument can be reversed to show the opposite direction.

b) Suppose $e = (u; v) \in N'$. By part a), $e \in M'$. If $u \in S$, then $N'(u) = M'(u)$ by part a). We deduce that the label of e at u coincides in M' and N'. If instead $u \notin S$,

then the label of e at u in G_M and G_N coincide by Lemma 3.1.3. The claim follows since $M'_S = M_S = N_S = N'_S$. □

Claim 3.4.4. *N' is an $(X' \cup S')$-matching of G.*

Proof. Let us first show that it is a matching of G. Recall that $M'_{X \cup S} = M$. Hence, $M' \triangle M$ has no edge incident to $X \cup S$. Since by hypothesis N is an $(X \cup S)$-matching, N has no edge incident to nodes other than $X \cup S$. Hence N' is the union of two node-disjoint matchings of G, hence a matching.

Now by hypothesis, $S \cup X \subseteq S' \cup X'$. Hence, N has no edge in $G - (S' \cup X')$. Moreover, M' is by hypothesis an $(S'; X')$-locally popular matching, so it is an $(S' \cup X')$-matching. Hence, N' is an $(S' \cup X')$-matching. □

Claim 3.4.5. *Let $k \in \mathbb{N}$ and $P^1, \ldots, P^k \in \mathcal{P}_{M'}$ be pairwise internally vertex-disjoint and X-disjoint. Assume that, for $j \in [k]$, path P^j is of level $\ell_j \in \{0,1,2\} \setminus \{0,1\}$ and parity $\Pi_j \in \{0,1\} \setminus \{0,1\}$, that every node appears at most twice in $P^1, \ldots, P^k$, no pair of paths have exactly the same endpoints, and no path is contained in X. Then there exist pairwise internally vertex-disjoint and X-disjoint paths $Q^1, \ldots, Q^k \in \mathcal{P}_{N'}$ with the following properties:*

1. *For $j \in [k]$, Q^j is of level ℓ_j and parity Π_j, and $P^j \setminus S' = Q^j \setminus S'$.*
2. *For $j \in [k]$, let u_j (resp. v_j) be the first (resp. last) node of P^j. If u_j (resp. v_j) $\notin X$, then u_j (resp. v_j) is the first (resp. last) node of Q^j. If u_j (resp. v_j) $\in X$, the first (resp. last) node of Q^j also belongs to X.*

Moreover,

3. *1,2 also hold if we switch the roles of M' and N'.*

Proof. The proof of the claim will only assume that M and N are $(S; X)$-locally popular matchings with the same $(S; X)$-tipping point. Recall that $N' = N \cup (M' \triangle M)$, $M' = M \cup (N' \triangle N)$ (by Claim 3.4.2), and that those unions are disjoint. Therefore, the

roles of M and N (hence those of $M^{\emptyset}$ and $N^{\emptyset}$) can be exchanged and the conclusions preserved. This proves 3 (assuming 1,2).

Let us consider any path P from $P^1;:::;P^k$, where we omit the superscript for the sake of readability. If $P \setminus X = ;$, we write $P_1 := P$. Else, we write P as the juxtaposition of $P_0;P_1;P_2;:::;P_q;P_1$, defined through the procedure described in Algorithm 5. Note that some of those paths may consist of a single node. We call them *trivial*. Note that a trivial path can be removed from the juxtaposition without changing the resulting P. Hence, in the following, we assume that all paths are non-trivial (without changing the subscripts of the remaining non-trivial paths), and we let q be the subscript of the last non-trivial path (other than 1).

The following fact immediately holds by construction.

Claim 3.4.6. *Let $fP_ig_{i2[q][f0;1g}$ be the output of Algorithm 5 on input $P = P^j$ for some j 2 $[k]$. Then: (i) P is the juxtaposition of $P_0;P_1;:::;P_q;P_1$. (ii) For all i 2 $[q] [f0; 1g$, P_i 2 $P_{M^{\emptyset}}$. (iii) For i 2 N, i 2, the first and last nodes of P_i belong to S. (iv) For i 2 N, i odd, P_i is an S-path and P_i $X [S$. (v) For i even, $P_i \setminus X = ;$. (vi) Assume q 1. Then P_1 ends at a node of S, and starts either at a node of X (if the* **if** *condition in Step 2 of Algorithm 5 is satisfied), or at a node of S (otherwise). (vii) If $P_1 \setminus X$ 6= ;, then P_1 is an S-path of $G_{M^{\emptyset}}[X [S]$ ending at some node of X.*

Let us call a subpath P_i of P *hidden* if i 6 1 mod 2 or $i = 1$ and $P_1 \setminus X$ 6= ; (i.e., condition (vii) above is verified). Now consider the collection of paths $fP_i^j g$ defined above, for j 2 $[k]$[2].

Fact 3.4.1. *Consider the family of paths $fP_i^j g$ with j 2 $[k]$. Then:*

[2] We introduced the superscripts back, to distinguish the paths produced by applying Algorithm 5 to $P^1, \ldots, P^k$.

Algorithm 5:

Let v be the first vertex of P.
if $v \in X$ **then**
  Let $P_0 = \{v\}$ and $u = v$.
else
  Let u' be the first vertex of P that is contained in X, and u the predecessor of u' in P. Note that $u \in S$. Let P_0 be the subpath of P between v and u (possibly $u = v$ and $P_0 = \{u\}$).
end if
Let $i = 1$.
Starting from u, traverse P until the first node $v \in S$, $v \neq u$ is encountered.
if such v does not exist **then**
  Set $q = i - 1$ and go to Step 21.
else
  Let P_i be the subpath between u and v. Set $i = i + 1$ and $u = v$.
  Starting from u, traverse P until the first node $u' \in X$ is encountered.
  if such u' does not exist **then**
    Set $q = i - 1$ and go to Step 21.
  else
    Let P_i be the subpath of P between u and the predecessor v of u' in P. Note that $v \in S$ and possibly $v = u$.
    Set $u = v$, $i = i + 1$, and go to Step 8.
  end if
end if
Let P_{-1} be the subpath of P between u and the last node of P.
Output $P_0, P_1, \ldots, P_q, P_{-1}$.

1. *$\{P_i^j\}$ is a collection of pairwise internally vertex-disjoint paths and each node appears at most twice in the collection.*
2. *The restriction of the collection $\{P_i^j\}$ to hidden paths is a family of pairwise X-disjoint S-paths from P_M, and the level of each of those path is not -1.*
3. *If $P_i^j \setminus P_{i'}^{j'} \neq \emptyset$, then either (a) $j = j'$ and $(ji - i'j - 1$ or $i, i' \in \{q_j, -1\})$, or (b) i (resp. i') is the first or last path of the juxtaposition leading to P^j (resp. $P^{j'}$).*

Proof. Part 1 follows from hypothesis, the fact that paths P^j, $j \in [k]$ form a collection of pairwise internally vertex-disjoint paths and Fact 3.4.6.

We now prove part 2. Restrict the collection $\{P_i^j\}$ to hidden paths. We already argued in Fact 3.4.6 that those are S-paths. As they are subpaths of M'-alternating paths, they are also M'-alternating. Using again Fact 3.4.6, they are contained in $X \cup S$. Using Claim 3.4.1, we deduce they are paths from P_M. The fact that they are X-disjoint follows by hypothesis for $j \neq j'$, and by Fact 3.4.6 for $j = j'$. The fact that M' is (S', X')-locally popular implies that they cannot be of level 1. This proves part 2.

We now prove part 3. Let $P_i^j \setminus P_{i'}^{j'} \neq \emptyset$. If $j = j'$, by Fact 3.4.6 the only possibility is that P_i^j, $P_{i'}^j$ are consecutive paths in the juxtaposition leading to P. This is case (a) in 3. Else, $j \neq j'$ and case (b) follows from the fact that P^j and $P^{j'}$ are internally vertex-disjoint, concluding the proof of 3. □

Consider the collection of hidden paths $\{P_i^j\}$ for $j \in [k]$, and let u_i^j (resp. v_i^j) be the first (resp. last) node of each of those paths. Consider $C = (U, L, \Pi)$ defined as follows: U is the ordered collection of pairs:

(u_1^j, v_1^j) if the first node of P_1^j belongs to S, and $(\cdot, v_1^j)$ otherwise.
(u_i^j, v_i^j) for $i \geq 3$ odd and
$(u_1^j, \cdot)$ if P_1^j is hidden.

L is the ordered collection of levels ℓ_i^j of the hidden paths P_i^j, while Π is the ordered collection of their parities. Using Claim 3.4.6, Claim 3.4.1 and the hypothesis, one easily verifies the following.

Claim 3.4.7. *C is an (S, X)-configuration. M is active at C, and the collection of hidden paths $\{P_i^j\}$ for $j \in [k]$ is a certificate of C at M.*

By hypothesis, N is also active at C. Hence, for all pairs (i, j) such that P_i^j is hidden, we can find pairwise X-disjoint S-paths $Q_i^j \in P_N$, with each P_i^j starting (resp. ending) at node u_i^j (resp. v_i^j) if this belongs to S, and at a node of X otherwise, of the appropriate level and parity.

For $j \in [k]$, recall that P^j is the juxtaposition of $(P_0^j, P_1^j, \ldots, P_{q_j}^j, P_1^j)$, from which we removed the trivial paths. Consider the path Q^j obtained by replacing, in the juxtaposition above, each hidden path P_i^j with the corresponding Q_i^j. We conclude the proof of the claim by showing that the collection of Q^j, $j \in [k]$, satisfies the claim.

Fix $j \in [k]$. Let us first argue that Q^j is a path in G that satisfies property 2 from the statement of the claim. Note that, by construction, for all hidden P_i^j, the first (resp. last) vertex of P_i^j coincides with the first (resp. last) vertex of Q_i^j, with possibly the exception of the first vertex of P_0^j (resp. last vertex of P_1^j) if it belongs to X. Moreover, since all Q_i^j are pairwise X-disjoint S-paths with no edge in $G \setminus X$, each Q^j is a path in G. The statement follows.

Let us now argue that Q^j is N^0-alternating. Since $N^0[(X^0 \cup S^0) \setminus X] = M^0[(X^0 \cup S^0) \setminus X]$ and all P_i^j that are not hidden are M^0-alternating and do not intersect X, we conclude that all such P_i^j are also N^0-alternating. For each hidden P_i^j, Q_i^j is an N-alternating path in $G_N[X \cup S]$ by construction. Since $G_N[X \cup S] = G_{N^0}[X \cup S]$ by Claim 3.4.1, each Q_j^i is also N^0-alternating. By construction, the parity of each hidden P_i^j coincides with that of the corresponding Q_i^j. We conclude that each Q^j is N^0-alternating.

Let us now show that $Q^j \in P_{N^0}$, and it is of the same level and parity of P^j. Let P_i^j be non-hidden. By Fact 3.4.6, it is contained in $G[X^0 \cup S^0] \setminus X$. By Claim 3.4.3, $P_i^j \setminus N^0 = P_i^j \setminus M^0$, and each edge from $P_i^j \setminus N^0$ has the same labels in $G_{M^0}[X^0 \cup S^0]$ and $G_{N^0}[X^0 \cup S^0]$. Now consider a hidden path P_i^j. By construction, Q_i^j is a path of $G_N[X \cup S] = G_{N^0}[X \cup S]$ of the same parity and level of P_i^j. The statement follows.

Now observe that property 1 of the claim holds, since the hypothesis $X \subseteq X^0$ implies $S^0 \setminus X = \emptyset$, hence we have:

$$Q^j \setminus S^0 = (Q^j \setminus (S^0 \setminus S)) \cup (Q^j \setminus (S^0 \setminus S)) = (P^j \setminus (S^0 \setminus S)) \cup (P^j \setminus (S^0 \setminus S)) = P^j \setminus S^0.$$

Finally, let us observe that $Q^1;\dots;Q^k$ are internally vertex-disjoint and X-disjoint. The X-disjointness follows from the fact that all Q_i^j corresponding to hidden P_i^j are X-disjoint, while other P_i^j do not intersect X by Fact 3.4.6. The fact that $Q^1;\dots;Q^k$ are pairwise internally vertex-disjoint follows from Fact 3.4.1. □

In order to conclude the proof of the lemma, we need to prove that N' is a $(S';X')$-locally popular matching whose tipping point is T'.

$(S';X')$-local popularity. Let us first show that N' is $(S';X')$-locally popular. Suppose it is not, then there exists a forbidden path or cycle P as in item (i), (ii), or (iii) from Theorem 1.3.5 that is contained in $G_{N'}[S' \cup X']$. Since (by Claim 3.4.1) $G_N[X \cup S] = G_{N'}[X \cup S]$ and $v \in X \cup S$ is N-exposed if and only if it is N'-exposed, we deduce $P \not\subseteq X \cup S$, else N would not be $(S;X)$-locally popular, a contradiction.

Suppose P is a path: take one that is inclusionwise minimal among all paths that certify that N' is not $(S';X')$-locally popular. We can then write P as the juxtaposition of $(P^1;P^2)$ of levels $\ell_1;\ell_2 \neq 1$, so that the first node of P^2 (which coincides with the last node of P^1) does not belong to X. Clearly $P^1;P^2$ satisfy the hypothesis of Claim 3.4.5 with the roles of N' and M' exchanged and $k = 2$. Let $Q^1;Q^2$ be the paths whose existence is guaranteed by the claim. We claim that the juxtaposition of $(Q^1;Q^2)$ is a forbidden path in $G_{M'}[X' \cup S']$, a contradiction.

First, observe that the last node of Q^1 coincides with the last node of P^1 (since the latter does not belong to X) which coincides with the first node of P^2, which coincides with the first node of Q^2 (again, since it does not belong to X). We now argue that P^1 and P^2 have no other node in common. We know they are internally vertex-disjoint and X-disjoint. Hence, if the first node of P^1 or the last node of P^2 belongs to X, we are done. Suppose not. Then, by Claim 3.4.5, the first node of Q^1 is the first node of P^1, which is different by hypothesis (P is a path) from the last node of P^2, which coincides, again by Claim 3.4.5, with the last node of Q^2. Hence, the juxtaposition of $(Q^1;Q^2)$ is a path of G, that we denote by Q. Again by Claim

3.4.5, Q^1, Q^2 2 P_{M^0}. Moreover, the parity and level of Q^1 (resp. Q^2) coincide with the parity and level of P^1 (resp. P^2). Hence Q 2 P_{M^0} is the required forbidden path, obtaining the desired contradiction.

The argument when P is a cycle follows in a similar fashion, writing P as the juxtaposition of $(P^1;P^2;P^3)$ with $P^1 \setminus X = P^2 \setminus X = ;$, and the endpoints of P_3 lying in S (note that we can assume that P contains exactly one $(+;+)$ edge, else it also contains a path with two $(+;+)$ edges and we are back to the previous case).

Tipping point. Let us now show that T^0 is the tipping point of N^0. First, observe that X X^0 implies $S^0 \setminus X = ;$. For v 2 $S^0 \setminus S$, we know that $N^0(v) = N(v) = M(v) = M^0(v)$. For v 2 $S^0 \cap S$, we have $N^0(v) = M^0(v)$ by construction. Hence $N^0_{S^0} = M^0_{S^0}$. Now let $C = \{U; L; \Pi\}$ be a $(S^0; X^0)$-configuration at which M^0 is active, with U as in (3.1). We will show that N^0 is also active at C. A symmetric argument shows that, if N^0 is active at C, then so is M^0, concluding the proof.

Take paths $P^1;::::;P^k$ that are the certificate of C at M^0. We claim that those paths satisfy the hypothesis of Claim 3.4.5. They are, by construction, S^0-paths that are pairwise X^0-disjoint. This implies that they are pairwise internally vertex-disjoint and X-disjoint (using again $X \setminus S^0 = ;$). The X^0-disjointness implies that each node from X^0 appears at most once. Moreover, again by construction, each node from S^0 appears at most twice. Hence, every node appears at most twice in $P^1;::::;P^k$. Moreover, by definition of U, no two paths have the same endpoints.

Let $Q^1;::::;Q^k$ 2 P_{N^0} be the paths whose existence is guaranteed by Claim 3.4.5. We show that $Q^1;::::;Q^k$ are the certificate of C at N^0, concluding the proof. Fix j 2 $[k]$. The parity and level of Q^j is the same as that of P^j. Hence, the parity of Q^j is Π_j, and its level $\grave{}_j$. Recall that Q^j is an S^0-path. By property 2 of Claim 3.4.5, Q^j starts (resp. ends) at the same node as P^j when this belongs to S^0, and at some node of X^0 otherwise. Using property 1 of Claim 3.4.5, we deduce that Q^j is a S^0-path that starts (resp. ends) at u_j (resp. v_j) when u_j $\not=$ $;$ (resp. v_j $\not=$ $;$), and at a node

$x \in X'$ otherwise.

We are left to show that paths $\{Q^j\}_{j=1,\ldots,k}$ are X'-disjoint. Suppose not, then there exists $v \in X'$, $j \neq j'$ such that $v \in Q^j \setminus Q^{j'}$. Note that v cannot be an internal node of Q^j or $Q^{j'}$, since those paths are pairwise internally vertex-disjoint by construction. On the other hand, if v is an endpoint of both Q^j and $Q^{j'}$, then the statement either follows from the X-disjointness of Q^j and $Q^{j'}$, or by property 2 of Claim 3.4.5. □

Chapter 4

The distance of stable and popular matchings from the Pareto-optimal frontier

4.1 Introduction

As we have seen in Chapter 1, stability is the pristine concept used in two-sided matching markets. However, in many contexts, one side of the market (typically schools, when the other side of the bipartition are students) is seen as just a good to be distributed to the other side. Using this point of view, preferences of schools play a minor role, hence stability is not an appropriate solution concept anymore. The focus is therefore moved towards *Pareto-optimality* for students. Pareto-optimal matchings can be obtained by *serial dictatorship* but are, in general, less tractable than stable matchings. For instance, unless P=NP, a Pareto-optimal matching of minimum size cannot be found efficiently, though a Pareto-optimal matching of maximum size can be found in polynomial time [4].

The goal of much of the modern literature on economic aspects of two-sided markets is to somehow resolve this tension between stability and Pareto-optimality. Of particular interest are therefore instances where there is a matching that is both stable and Pareto-optimal. This situation has been studied in [3]. Because of the lattice structure of stable matchings, the only candidate to lie in this set is the man-optimal stable matching. Since this matching can be found efficiently, and Pareto-optimality can also be checked efficiently [4], we conclude the following fact.

Theorem 4.1.1. *Given a marriage instance, one can decide in polynomial time if there is a stable matching that is Pareto-optimal.*

Unfortunately, in typical instances, a matching that is both Pareto-optimal and stable does not exist. However, this fact alone does not convey how contradicting the properties of Pareto-optimality and stability are for a given instance. Moreover, as the families of both stable and Pareto-optimal matchings can be very large, a market designer is often left with the dilemma of which matching to chose. A stable matching that is "close" (in some well-defined sense) to being Pareto-optimal could be favoured over one that is "far away" (and similarly exchanging Pareto-optimality and stability). But do such matchings exist and, if they do, can they be computed efficiently?

In this section, we introduce and study parameters measuring the distance between the sets of stable/popular and Pareto-optimal matchings. As we will see, those parameter can be interpreted in two ways. First, as a structural property of the instance, describing in a natural way how far stable/popular matchings are from the Pareto-optimal frontier. Then, as lower bounds on the number of "changes" certain well-known algorithms need to perform when moving from a stable to a Pareto-optimal matchings.

Δ, Λ, Γ, and our results.

In order to introduce the first parameter, fix a marriage instance $(G; <)$ with bipartition $(A; B)$ and for $M; M^0 \ 2 \ M(G)$ define

$$\Delta(M; M^0) := jfm \ 2 \ A : M^0(m) \ 6= \ M(m)gj:$$

Hence, for each pair of matchings M and M^0, we have $\Delta(M; M^0) \ 2 \ f0; 1; ::; jAjg$, with the extreme values achieved when $M = M^0$ and when $M \ \setminus \ M^0 = ;$, respectively. For

an instance $(G; <)$, we let:

$$\Delta(G; <) := \min_{M \text{ stable}, M' \text{ Pareto opt}} \Delta(M; M').$$

Hence, $\Delta(G; <)$ resembles somehow the classical Hamming distance between binary vectors. It measures the minimum number of men that have to change partner, in order to move from a stable to a Pareto-optimal matching, and is a natural measure of the distance between the set of Pareto-optimal and stable matchings. We will omit the dependency on $(G; <)$ in Δ when the underlying graph is clear from the context.

It will be useful to work with matching-dependent functions as well. For a stable matching M, we let

$$\Delta(M) := \min_{M' \text{ Pareto-optimal}} \Delta(M; M').$$

For a Pareto-optimal matching M', we let

$$\Delta(M') := \min_{M \text{ stable}} \Delta(M; M').$$

The notation is well-defined, since when M is both Pareto-optimal and stable, we have $\Delta(M) = 0$.

Many algorithm studied in theory and used in practice, such as Kesten's EADAM [37] and the Top Trading Cycle [50], take as input an initial (stable) matching M, and produce a Pareto-Optimal matching M' that every man weakly prefers to M (i.e., M' Pareto-dominates M). This is motivated by the fact that no man would accept a "globally better solution", if this solution is less competitive for him. Since departing from stability is often countered by protests of agents, it is therefore important to understand how to achieve a Pareto-optimal matching with a minimum number of changes.

To capture this setting, we define therefore $PO(M)$ to be the set of Pareto-

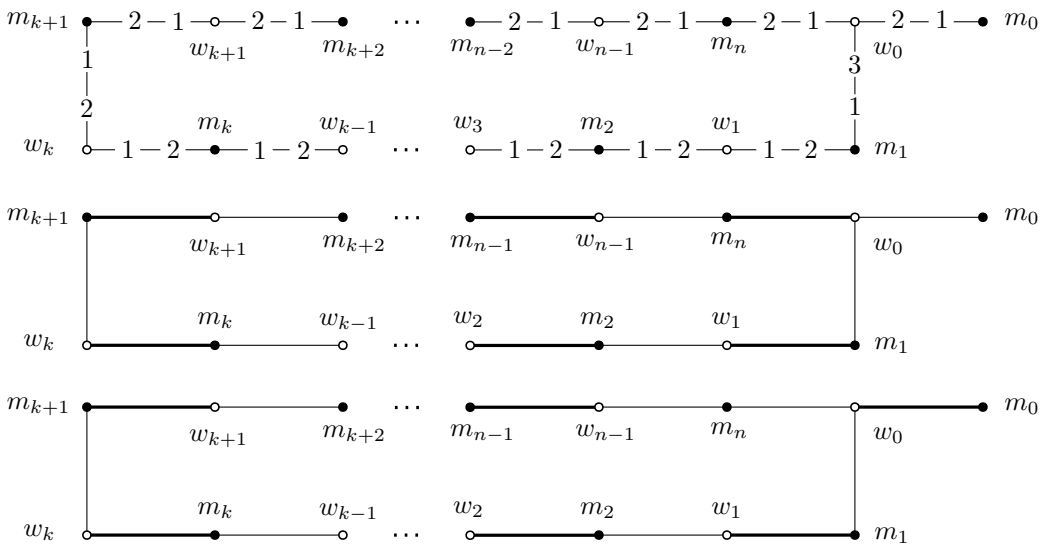

Figure 4.1: On the top: a marriage instance $(G; <)$. In the middle: the only stable matching M of the instance. It contains exactly one coalition cycle C. One can easily check (or resort to Lemma 4.2.2) that the only matching in $PO(M)$ is $M \quad C$. Hence, $\Lambda(M) = k + 2$. On the bottom: the matching realizing $\Delta(M) = 1$.

optimal matchings that Pareto-dominate M. Note that, for $M' \in PO(M)$, we can alternatively compute $\Delta(M; M')$ as follows.

Observation 4.1.2. *For a matching M and a Pareto-optimal matching $M' \in PO(M)$, we have:*

$$\Delta(M; M') = |\{m \in A : M'(m) \neq M(m)\}| = |\{m \in A : M'(m) >_m M(m)\}|. \quad (4.1)$$

We define therefore

$$\Lambda(G; <) := \min_{M \in S; M' \in PO(M)} \Delta(M; M').$$

Again, the associated local function is

$$\Lambda(M) := \min_{M' \in PO(M)} \Delta(M; M')$$

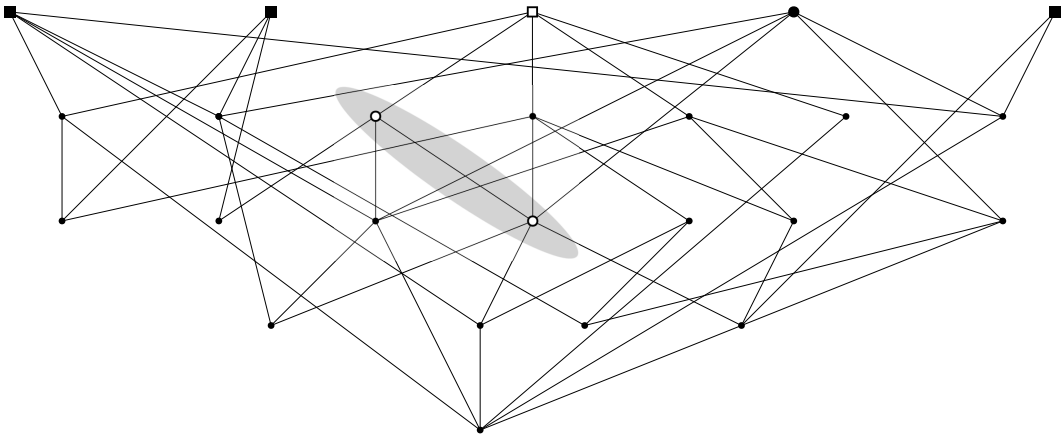

Figure 4.2: The Hasse diagram of the poset of matchings of a marriage instance with 4 students and schools. Stable matchings are represented by unfilled circles, while Pareto-optimal matchings correspond to elements of the poset without predecessors. There is exactly one Pareto-optimal matching $M_0^\emptyset$ (unfilled square) that dominates the student-optimal stable matching M_0. Hence $\Gamma(M_0) = \Gamma(M_0; M_0^\emptyset)$. There is a second Pareto-optimal matching $M_1^\emptyset$ (filled large circle) that dominates the other stable matching M_1. Hence $\Gamma(M_1) = \min\{\Gamma(M_1; M_0^\emptyset); \Gamma(M_1; M_1^\emptyset)\}$ and $\Gamma(G; <) = \min\{\Gamma(M_0; M_0^\emptyset); \Gamma(M_1; M_0^\emptyset); \Gamma(M_1; M_1^\emptyset)\}$. On the other hand, in order to compute $\Delta(G; <)$, we must also consider the other Pareto-optimal matchings (filled squares).

Clearly, one has $\Delta = 0$ if and only if $\Lambda = 0$, and $\Lambda \quad \Delta$. It is maybe counterintuitive that the latter inequality can be strict – that is, the matching realizing $\Delta(M)$ may not Pareto-dominate M. Actually, it is easy to construct examples where the distance between Λ and Δ can be arbitrarily large, see Fig. 4.1. See Fig. 4.2 for a pictorial representation of Λ vs. Δ.

Many algorithms produce a Pareto-optimal matching $M^\emptyset$ by starting from an already existing matching M, and then iteratively take the symmetric difference of M with *Pareto-improving* (or *coalition*) cycles, see Section 1.3 for definitions. Recall that these are M-alternating cycles where every man prefers his neighbor not in M to the one in M. We call such an Algorithm a *Pareto-improving* algorithm. At each step, multiple choices for C are possible. As we argue in Lemma 4.2.3, when M is stable (and, more generally, popular) $\Lambda(M)$ is exactly the minimum number of men

that need to be in some Pareto-improving cycle when going from M to $M' \in PO(M)$. Hence, it gives a lower bound to the number of changes in the matching that any such algorithm must perform.

Recall that the existence of a stable matching that is Pareto-optimal (equivalently, that $\Delta = \Lambda = 0$) can be verified efficiently by building on the lattice structure of the set of stable matchings.

However, the general problem of computing or even approximating Δ is very hard. We show in fact the following.

Theorem 4.1.3. *There exists a universal constant c such that the following holds. Unless P=NP, for each $\varepsilon > 0$, there is no polynomial-time algorithm that, given a marriage instance $(G(A \cup B; E); \prec)$, accepts if $\Delta(G) \leq \varepsilon|A|$, rejects if $\Delta(G) \geq c\varepsilon|A|$, and is free to accept or reject if none of these cases apply. In particular, computing $\Delta(G)$ is NP-hard.*

The proof of this theorem and of related hardness results is presented in Section 4.5. We counterbalance this results by showing that deciding whether Δ or Λ is a constant can be decided in polynomial time. The proof of next result appears in Section 4.4. We denote by PO the set of Pareto-optimal matchings.

Theorem 4.1.4. *Let k be a constant. There exists a polynomial-time algorithm that, given a marriage instance $(G; \prec)$, either concludes $\Delta \leq k$ (resp. $\Lambda \leq k$) and outputs $M \in S$, $M' \in PO$ (resp. $M \in S$, $M' \in PO(M)$) with $\Delta(M; M') \leq k$, or correctly concludes that $\Delta > k$ (resp. $\Lambda > k$).*

A corollary of Theorem 4.1.3 is that there exist instances where the distance of any stable matching from the Pareto-optimal frontier is very large. One could therefore think of relaxing stability and hopefully moving "closer" to the set of Pareto-optimal

matchings. Hence, we define the parameter $\Gamma(G; <)$, that is the analogous of $\Delta(G; <)$ when the set of stable matchings in replaced by that of popular matchings.

Formally, for a marriage instance $(G; <)$, we let:

$$\Gamma(G; <) := \min_{M \text{ popular}, M' \text{ Pareto opt}} \Delta(M; M').$$

It will also be useful to define, for a marriage instance $(G; <)$ and a Pareto-Optimal matching M'

$$\Gamma(M') := \min_{M \text{ popular}} \Delta(M; M');$$

and similarly for a popular matching M

$$\Gamma(M) := \min_{M' \text{ Pareto-optimal}} \Delta(M; M').$$

The notation is well-defined, since when M is both Pareto-optimal and popular, we have $\Gamma(M) = 0$.

As one can imagine, it can in fact be the case that Γ is much smaller than Δ. In Section 4.3, we give an example of an instance where $\Gamma = 0$ and $\Delta = |A| \quad 2$. Maybe not surprisingly (given complexity results on popular matching presented in this thesis), when we move from stability to popularity, even deciding whether there is a matching that is both popular and Pareto-optimal becomes hard.

Theorem 4.1.5. *Let $(G; <)$ be a marriage instance. It is NP-Complete to decide if $\Gamma(G) = 0$.*

On the other hand one could expect that, by relaxing stability to popularity, it would be easier to find a matching that is reasonably close to the Pareto-optimal frontier. Our next results shows this is not the case.

Theorem 4.1.6. *There exists a universal constant c such that the following holds. For each > 0, unless P=NP, there is no polynomial-time algorithm that, given a preference system (G(A [B; E); <), accepts if* $\Gamma = 0$*, rejects if* Γ *cjAj, and is free to accept or reject if none of these cases apply.*

The techniques

Recall the general approach developed in Chapter 2 to prove hardness results on stable/popular matchings. Starting from an instance of Monotone 3-SAT, a marriage instance (*G; <*) is constructed, with the property that is satisfiable iff (*G; <*) contains the required stable or popular matching.

In order to prove results from this section we push this connection further. We illustrate the general idea for Theorem 4.1.3, which is also followed by most other hardness results in this section (although technical details are quite different). The reduction between and (*G; <*) is defined so that there is a bijection between stable matchings in *G* and *consistent* literal assignments in (see Section 1.4 for definitions). We then show that one can define a map *q* from *feasible* assignments to Pareto-optimal matchings, in such a way that the closer a feasible literal assignment is to be consistent (in terms of number of satisfied in close), the closer (in terms of Δ) the Pareto-optimal matching *q*( ) is to be stable. If we could show that *q* is surjective over the set of Pareto-optimal matchings, then we would be done by Theorem 1.4.2 on the hardness of approximation of Monotone 3-SAT. However, this is not the case: there are Pareto-optimal matchings that are not in the codomain of *q*. However, we show that none of those Pareto-optimal matchings is the closest (again, in terms of Δ) to any stable matching, concluding the proof.

For the polynomial-time results, we heavily build on known structural and algorithmic facts on stable matchings, introduced in Chapter 1.

Organization of the chapter. We start with settling notation and basic facts in 4.2. In Section 4.3, we present a number of examples which illustrate that the parameters introduced can be very different from each other, and that more generally show the complexity of the problem we investigate. Our polynomiality results are presented in Section 4.4. Theorem 4.5 and related results are proved in Section 4.5. Theorem 4.1.5 and Theorem 4.1.6 are proved in section 4.5.

4.2 Further notation and basic facts

If H is a subgraph of a graph G and $M; M'$ are matchings on G, we let $\Delta_H(M; M') := \Delta(M_H; M'_H)$, where for a matching N and a subgraph H of G, N_H is the restriction of N to nodes of H.

Lemma 4.2.1. *Let M be a popular matching and $M' \in PO(M)$. Then $V(M) = V(M')$ and $M \oplus M'$ is a collection of vertex-disjoint coalition cycles for M.*

Proof. We first observe $V(M) = V(M')$. Since M' is Pareto-optimal, we have

$$A(M') \supseteq A(M). \tag{4.2}$$

Suppose first by contradiction that there exists $b \in B(M') \cap B(M)$. Ask all agents to vote between M and M'. Let A_M (resp. $A_{M'}$, A_0) be the number of men voting for M (resp. M', abstaining). Define B_M, $B_{M'}$, B_0 similarly. By definition of Pareto-dominance,

$$A_M = 0; A_{M'} = |\{a \in A : M(a) \neq M'(a)\}|.$$

Clearly b contributes to $B_{M'}$ (and not to B_M). Hence we have:

$$\begin{aligned} A_{M'} + B_{M'} &\quad j f a \ 2 \ A : M(a) \ 6= M'(a) g j + 1 \\ &= j f b \ 2 \ B : M(b) \ 6= M'(b) g j + 1 \\ &> B_M = A_M + B_M ; \end{aligned} \tag{4.3}$$

contradicting the popularity of M. This shows

$$B(M') \quad B(M): \tag{4.4}$$

Now assume there exists $a \ 2 \ A(M') \ n \ A(M)$. Because of (4.2), we deduce that there exists $b \ 2 \ B(M') \ n \ B(M)$, and we are back to the previous case, which we know cannot happen. Together with (4.2), this implies $A(M') = A(M)$. Hence $jB(M)j = jB(M')j$, and (4.4) implies $B(M) = B(M')$. We conclude $V(M) = V(M')$.

Hence, the symmetric difference of M and M' is given by a collection of vertex-disjoint cycles. Since $M' \ 2 \ PO(M)$, those cycles must be coalition cycles for M, and the thesis follows. □

A *Pareto-improving algorithm* A starts from a matching M that is maximal and trade-in free, chooses a coalition cycle C of M, and replaces M with $M \ 4 C$. Note that $M \ 4 C$ is maximal and trade-in free. When M has no coalition cycle, then the algorithm outputs M which by Lemma 1.3.4 is Pareto-optimal. The *number of displaced men* $d(A; M)$ is the number of men for which the partner in the input matching M and output matching differ.

Lemma 4.2.2. *Let M be a popular matching and $M' \ 2 \ PO(M)$. Then there is Pareto-improving algorithm A that on input M, outputs M'. Conversely, any Pareto-improving algorithm A on input M outputs a matching from $PO(M)$.*

Proof. Let M be a popular matching and $M' \ 2 \ PO(M)$. Since a popular matching is

maximal and trade-in free, we deduce that M is a well-defined input to any Pareto-improving algorithm. By Lemma 4.2.1, there is a collection of vertex-disjoint cycles $C_1;:::;C_k$ such that $M \quad M' = fC_1;:::;C_kg$. Let now $i \ 2 \ [k]$. Since $C_1;:::;C_i$ are vertex disjoint, $C_{i+1};:::;C_k$ are coalition cycles in $((M \quad C_1) \quad :::) \quad C_i$. Hence, there is a Pareto-improving algorithm that, for $i \ 2 \ [k]$, at step i selects C_i, and this algorithm outputs M'.

The converse statement follows from the fact that, for any matching M and coalition cycle C, $M \quad C \quad M$. □

Lemma 4.2.3. *Let M be a popular matching. Then*

$$\Lambda(M) = \min d(A; M);$$

where the minimum is taken over all Pareto-improving algorithms A.

Proof. Let M' realize $\Lambda(M)$. Then $M' \ 2 \ PO(M)$ and, by Lemma 4.2.2 and its proof, there exists a Pareto-Improving algorithm A that outputs M by selecting, at each step, one of the k coalition cycles from $M \quad M'$. Hence, using Observation 4.1.2, we have:

$$\min d(A; M) \quad d(A \ ; M) = jfa \ 2 \ A : M(a) \ 6= \ M'(a)gj = \Lambda(M);$$

showing one side of the inequality.

Conversely, consider the execution of a Pareto-improving algorithm A on input M, and let M' be the matching it outputs on input M. From Lemma 4.2.2, $M' \ 2 \ PO(M)$. Hence, using again Observation 4.1.2, we deduce:

$$\Lambda(M) \quad \Delta(M; M') = jfa \ 2 \ A : M(a) \ 6= \ M'(a)gj = d(A; M);$$

showing the other side of the inequality and concluding the proof. □

Lemma 4.2.4. *Let M be a popular matching that contains k vertex-disjoint coalition cycles. Then $\Lambda(M) \quad \Delta(M) \quad k$.*

Proof. $\Lambda(M) \quad \Delta(M)$ follows by definition. Let M^0 be a Pareto-optimal matching that realizes $\Delta(M)$. Consider a coalition cycle C of M. Since M^0 is Pareto-optimal, then $C \not\supseteq M^0$. This implies that at least one man in C has changed his partner. Since there are k vertex-disjoint coalition cycles, and each cycle has at least one man that changed his partner, then $\Delta(M) \quad k$. □

Lemma 4.2.5. *For matchings M, M^0, we have:*

$$\frac{1}{2}jM \quad M^0j \quad \Delta(M;M^0) \quad jM \quad M^0j:$$

Proof. Assume there are k men, $m_1;m_2;:::;m_k$, that change partners between M and M^0. Then

$$M \quad M^0 \quad fm_1M(m_1);m_1M^0(m_1);m_2M(m_2);m_2M^0(m_2);::::;m_kM(m_k);m_kM^0(m_k)g:$$

If m_i, $i \in 1:::k$ is unmatched in M (or M^0), then $m_iM(m_i) = ;$ (or $m_iM^0(m_i) = ;$). We conclude that $jM \quad M^0j \quad 2k$, or $\frac{1}{2}jM \quad M^0j \quad k = \Delta(M;M^0)$.

On the other hand for each $i = 1;:::;k$ at least one of $M(m_i)$ and $M^0(m_i)$ is non-empty, call it $M^{00}(m_i)$. This implies that

$$M \quad M^0 \quad fm_1M^{00}(m_1);m_1M^{00}(m_1);::::;m_kM^{00}(m_k)g:$$

We conclude that $jM \quad M^0j \quad k = \Delta(M;M^0)$. □

Lemma 4.2.6. *For each marriage instance $(G(A \cup B);\prec)$ we have $\Gamma(G) \quad \Delta(G) \quad jAj \quad 1$, and there are infinitely many instances with $\Gamma(G) \quad jAj \quad 2$ (resp. $\Delta(G) = jAj \quad 1$).*

Proof. Kesten's EADAM algorithm [37] produces a Pareto-optimal matching that agrees with the man-optimal stable matching in at least one edge. This shows Γ Δ jAj 1. The example with $\Delta = jAj$ 1 is presented in Section 4.3. The example with Γ jAj 2 is given in Section 4.3. □

4.3 Phenomenology

In this section, we provide some examples. The goal is twofold: First, we want to illustrate the complexity of the problems investigated in this chapter. Second, we want to show some extreme scenarios, showing that Δ, Λ, and Γ can be very different, and can essentially cover all the spectrum, from being 0 to linear in jAj.

$\Delta = jAj$ 1 and $\Gamma = 0$

In this section we provide an infinite family of instances, such that $\Gamma = 0$ and Δ jAj 2. This construction is based on a modification of Irving and Leather's instances [29] given by Faenza and Zhang [15].

Algorithm 6 and Algorithm 7 construct matrices A_n and B_n that represent complete preferences lists for agents from A and B respectively, with $jAj = jBj = 2^n + 1$. An element c_{ij} of a matrix indicates that entity c_{ij} is in position j of ith preference list. We denote the resulting marriage instance by $(G_n; <_n)$. Let $I(i)$ be a matrix of size i by i with all elements being 1. Let $A(i : j; k : l)$ be a submatrix of A with all entries of A at intersection of rows $i; i+1; : : : ; j$ and columns $k; k+1; : : : ; l$. $A(i;)$ indicates the ith row of matrix A.

Algorithm 7: Construction of B_n

$B_n(1 : 2^n; 1) = A_n(1 : 2^n; 2^n)$;
$B_n(1 : 2^n; 2) = (2^n + 1; : : : ; 2^n + 1)$;
for ($i = 3 : : : 2^n + 1$) f
  | $B_n(1 : 2^n; i) = A_n(1 : 2^n; 2^n + 2 \quad i)$;
g
$B_n(2^n + 1;) = (2^n + 1; 1; 2; 3; : : : ; 2^n)$;

Algorithm 6: Construction of A_n

initialization $C_1 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$;
for ($i = 2;\dots;n$) *f*
  $C'_{i\ 1} = C_{i\ 1} + 2^{i\ 1} I(2^{i\ 1})$;
  $C_i = \begin{pmatrix} C_{i\ 1} & C'_{i\ 1} \\ C'_{i\ 1} & C_{i\ 1} \end{pmatrix}$;
g
$A_n(1 : 2^n; 1 : 2^n) = C_n$;
$A_n(2^n + 1;\ ) = (1; 2;\dots;2^n + 1)$;
$A_n(\ ; 2^n + 1) = (2^n + 1;\dots;2^n + 1)$

Note that, in the instance $(G_n; <_n)$, for $i\ 2\ [2^n]$, man i's top preference is i and his second-to-least favorite woman is $2^n \quad i + 1$. Every man's least favorite woman is $2^n + 1$. Conversely, for $i\ 2\ [2^n]$, woman i's favorite partner is man $2^n \quad i + 1$, her least favorite is man i. Woman $2^n + 1$'s favorite partner is man $2^n + 1$.

Table 4.1 shows an example of the construction for $n = 3$.

1	1	2	3	4	5	6	7	**8**	9
2	2	1	4	3	6	5	8	**7**	9
3	3	4	1	2	7	8	5	**6**	9
4	4	3	2	1	8	7	6	**5**	9
5	5	6	7	8	1	2	3	**4**	9
6	6	5	8	7	2	1	4	**3**	9
7	7	8	5	6	3	4	1	**2**	9
8	8	7	6	5	4	3	2	**1**	9
9	1	2	3	4	5	6	7	8	**9**

1	**8**	9	7	6	5	4	3	2	1
2	**7**	9	8	5	6	3	4	1	2
3	**6**	9	5	8	7	2	1	4	3
4	**5**	9	6	7	8	1	2	3	4
5	**4**	9	3	2	1	8	7	6	5
6	**3**	9	4	1	2	7	8	5	6
7	**2**	9	1	4	3	6	5	8	7
8	**1**	9	2	3	4	5	6	7	8
9	**9**	1	2	3	4	5	6	7	8

Table 4.1: An example of the construction from Algorithm 6 (left) and Algorithm 7 (right) for $n = 3$. The unique stable matching is circled.

In the following, we fix $n\ 2\ \mathbb{N}$ and the instance $(G_n; <_n)$. We denote a matching of $(G_n; <_n)$ by an ordered tuple with $2^n + 1$ elements, where man i is matched to the woman corresponding to the i-th woman in the tuple. We let $S_n := (2^n; 2^n\ 1;\dots;1; 2^n + 1)$ and $P_n := (1; 2;\dots;2^n + 1)$.

Lemma 4.3.1. *S_n is the only stable matching.*

Proof. One readily verifies that Gale-Shapley algorithm with men proposing outputs S_n. On the other hand, as each woman is matched to her favorite partner in S_n, it is also the woman-optimal stable matching. Hence, $(G_n; <_n)$ only has one stable matching. □

Lemma 4.3.2. *P_n is Pareto-optimal and popular.*

Proof. P_n is obtained by serial dictatorship algorithm using the following order of men: $\{1; 2; \ldots; 2^n + 1\}$. This implies that P_n is Pareto-optimal by Lemma 1.3.9.

We now consider graph G_{P_n} obtained by labelling and removing edges as discussed in Chapter 1. Notice that:

$(i; j) = (\ \ ; +)$ for $i = 1; \ldots; 2^n$ and all possible j;
$(2^n + 1; j) = (+; +)$ for $j = 1; \ldots; 2^n$;
$(k; 2^n + 1) = (\ \ ; \ \ )$ for $k = 1; \ldots; 2^n$.

This implies that all $(+; +)$ edges are adjacent. We conclude that there is no alternating path with two $(+; +)$ edges. All adjacent edges to $2^n + 1$ 2 B are $(\ \ ; \ \ )$, this implies that there is no alternating cycle that contains $(+; +)$. Since all vertices are matched, using Theorem 1.3.4 we conclude that P_n is popular. □

Lemma 4.3.3. *$jS_n \setminus Mj \quad 1$ for all Pareto-optimal matchings M.*

Proof. Let M be a Pareto-optimal matching of $(G_n; <_n)$. Assume first that the $2^n + 1$-th men and women are matched to each other in M. Then $M \quad P_n$, else P_n Pareto-dominates M, a contradiction. The statement then follows since $jS_n \setminus P_n j = 1$.

Hence, we can assume $M(2^n+1) \neq 2^n+1$ and, by contradiction, that $jM \setminus S_n j \quad 2$. Then we have $M(i) = 2^n \quad i + 1$ and $M(j) = 2^n \quad j + 1$ for some $i \neq j \ 2 \ [2^n]$. Construct matching M' starting from M swapping the partner of men i and j – that is, $M'(i) = 2^n \quad j + 1$ and $M'(j) = 2^n \quad i + 1$. Clearly $M' \quad M$, hence M is not Pareto-optimal. □

Lemma 4.3.2 implies that $\Gamma = 0$. Lemma 4.3.1 and 4.3.3, together with the fact that preference lists of $(G_n; <_n)$ are complete, imply that $\Delta = jAj \quad 1$.

$\Gamma = 0$, $\Delta(M) = \Theta(jAj)$ for any dominant or stable matching M

It is a natural question whether Γ can be computed or well-approximated just by restricting the search over the set of stable or dominant matchings. We have seen in Section 1 that those are well-behaved classes of popular matchings and in Section 2.3 that the best polynomial-time approximation to a popular matching of maximum weight (with nonnegative weights) can be computed by picking the stable or dominant matchings of maximum weights.

In this section, we show this is not the case. In fact, We construct an infinite family of instances, such that $\Delta = 0$ is achieved on a *middle* matching (i.e., neither stable nor dominant), and $\Delta = \Omega(n)$ for any other non middle matching, where n is the number of men.

Our example of an instance $(G(A \ [\ B; E); <)$ is constructed from the same kind of blocks (or gadgets) that are connected with additional edges. One block i consists of five men $m_i^0; m_i^1; m_i^2; m_i^3; m_i^4$ and four women $w_i^0; w_i^1; w_i^2; w_i^3$, Fig. 4.3. Their preferences lists are the following:

$$
\begin{array}{llll}
m_i^0: & w_i^0 & w_i^0: & m_i^1; m_i^0 \\
m_i^1: & w_i^0; w_i^1 & w_i^1: & m_i^1; m_i^2 \\
m_i^2: & w_i^3; w_i^1; w_i^2 & w_i^2: & m_i^2 \\
m_i^3: & w_i^3 & w_i^3: & m_i^2; m_i^3; m_i^4 \\
m_i^4: & w_i^3 & &
\end{array}
$$

Consider an instance I with $k \quad 2$ blocks with $5k$ men and $4k$ women. Add $2k(k \quad 1)$ edges: $(m_i^1; w_j^0)$ and $(m_i^3; w_j^3)$ for any $i; j = 1; : : : ; k$, $i \neq j$. For each man m_i^1 and m_i^3 add the $k \quad 1$ women to the beginning of his preference list in arbitrary

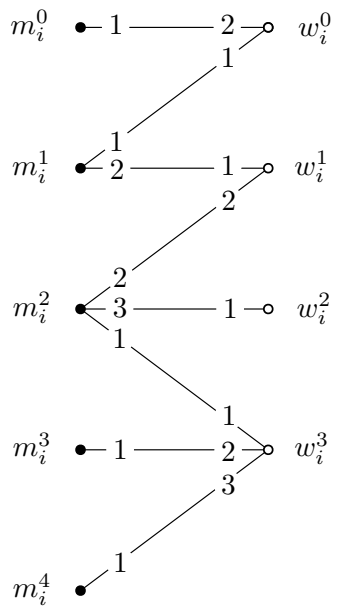

Figure 4.3: One block i consists of five men $m_i^0; m_i^1; m_i^2; m_i^3; m_i^4$ and four women $w_i^0; w_i^1; w_i^2; w_i^3$ with their preferences labeled on the edges.

order. For each women w_i^0 and w_i^3 add k 1 men to the end of her preferences list in arbitrary order.

Lemma 4.3.4. *Edges $(m_i^1; w_j^0)$, $(m_i^3; w_j^3)$, $(m_i^4; w_i^3)$ and $(m_i^2; w_i^1)$ are not contained in any popular matching for any $i; j = 1; : : : ; k$, $i \neq j$.*

Proof. If one of the edges $(m_i^1; w_j^0)$, $i; j = 1; : : : ; k$, $i \neq j$ belongs to a matching M then $(m_j^0; w_j^0) = (+; +)$ is adjacent to an unmatched node m_j^0. This implies that M cannot be popular.

Similarly, if one of the edges $(m_i^3; w_j^3)$, $i; j = 1; : : : ; k$, $i \neq j$ belongs to a matching M then $(m_j^4; w_j^3) = (+; +)$ is adjacent to an unmatched node m_j^4. This implies that M cannot be popular.

Now we will prove that no popular matching contains one of the edges $(m_i^4; w_i^3)$, $i = 1; : : : ; k$. By contradiction, assume there is a popular matching M that contains an edge $(m_i^4; w_i^3)$, for some $i \in \{1; : : : ; k\}$. From above $(m_i^3; w_j^3)$ is unmatched in any popular matching for $i; j = 1; : : : ; k$, $i \neq j$. This implies that $(m_i^3; w_i^3) = (+; +)$ is

adjacent to an unmatched node m_i^3 and contradicts popularity of M.

In the last step we show that no popular matching contains one of the edges $(m_i^2; w_i^1)$, $i = 1; : : : ; k$. By contradiction, assume there is a popular matching M that contains an edge $(m_i^1; w_i^1)$, for some $i \in \{1; : : : ; k\}$. From above $(m_i^4; w_i^3) \notin M$, for any $i = 1; : : : ; k$. Notice that $(m_i^2; w_i^3) = (+; +)$, and $(m_i^3; w_i^3) \in M$, otherwise $(m_i^2; w_i^3) = (+; +)$ is adjacent to an unmatched node w_i^3. Node m_i^1 has to be matched otherwise there is a path $m_i^1; w_i^1; m_i^2; w_i^3$ that contains an unmatched node m_i^1 and an edge $(m_i^2; w_i^3) = (+; +)$. We arrive to a contradiction that M is popular, since a path $m_i^0; w_i^0; m_i^1; w_i^1; m_i^2; w_i^3$ contains an unmatched node m_i^0 and an edge $(m_i^2; w_i^3) = (+; +)$. □

Lemma 4.3.5. *There is exactly one dominant matching* $M_0 = \{(m_i^0; w_i^0); (m_i^1; w_i^1); (m_i^2; w_i^2); (m_i^3; w_i^3) : i = 1 : : : k\}$ *in the instance* I.

Proof. First, we show that matching M_0 is popular. Notice that $(m_i^1; w_i^0)$ and $(m_i^2; w_i^3)$ are the only $(+; +)$ edges, and m_i^4 are the only unmatched nodes for $i = 1; : : : ; k$. Edges $(m_i^1; w_i^0)$ and $(m_i^2; w_i^3)$ do not belong to any alternating path within any block i. Similarly, node m_i^4 does not belong to any alternating path with a $(+; +)$ edge within any block i. Since the edge $(m_i^1; w_j^0)$ is unmatched for any $i; j = 1; : : : ; k$, $i \neq j$, then there is no alternating path that contains two $(m^1; w^0)$ edges from different blocks $\grave{} = i; j$. We apply the same argument for edges $(m^3; w^3)$ edges from different blocks $\grave{} = i; j$. Since there is no alternating path that connects edges $(m_i^1; w_i^0)$ and $(m_i^2; w_i^3)$ within one block, we argue that there is no alternating path between any two blocks that contains edges $(m_i^1; w_i^0)$ and $(m_j^2; w_j^3)$ from different blocks i and j. Similarly, we show that there is no alternating path that contains any $(+; +)$ edge from one block and unmatched node from another block.

Since nodes m_i^0 and w_i^2 have degree one for all blocks $i = 1; : : : ; k$, and edge $(m_i^4; w_i^3)$ is unmatched in any popular matching for all $i = 1; : : : ; k$, and dominant matching has the largest number of matched nodes, we conclude that $M_0 =$

$f(m_i^0; w_i^0); (m_i^1; w_i^1); (m_i^2; w_i^2); (m_i^3; w_i^3) : i = 1:::kg$ is the only popular matching with the largest number $8k$ matched nodes, and, therefore, is the only dominant matching. □

Lemma 4.3.6. *There is exactly one stable matching $M_1 = f(m_i^1; w_i^0); (m_i^2; w_i^3) : i = 1:::kg$ in the instance I.*

Proof. For any block $i = 1;::::;k$ edges $(m_i^0; w_i^0); (m_i^1; w_i^1); (m_i^2; w_i^1); (m_i^3; w_i^3); (m_i^4; w_i^3)$ are either $(+;\ )$ or $(\ ;+)$. All edges between the blocks $(m_i^1; w_j^0); (m_i^3; w_j^3)$ are $(+;\ )$ for $i; j = 1;::::;k$, $i \neq j$. We conclude that there are no $(+;+)$ edges, and M_1 is stable. Similarly to Lemma 4.3.4, we argue that $(m_i^1; w_j^0)$, $(m_i^3; w_j^3)$ are not contained in any stable matching for any $i; j = 1;::::;k$, $i \neq j$. Recall that all stable matchings have the same people matched. We conclude that there are no any other options to match $m_i^1; w_i^0; m_i^2; w_i^3$ for $i = 1;::::;k$, $i \neq j$, and M_1 is the only stable matching. □

Lemma 4.3.7. *Matching $M_2 = f(m_i^0; w_i^0); (m_i^1; w_i^1); (m_i^2; w_i^3) : i = 1:::kg$ is middle and Pareto-optimal.*

Proof. First, we show that M_2 is popular. Notice that there is exactly one $(m_i^1; w_i^0) = (+;+)$ edge within any block $i = 1;::::;k$. Since edge $(m_i^1; w_j^0)$ is unmatched for $i; j = 1;::::;k$, $i \neq j$, we conclude that there is no alternating path with two $(+;+)$ edges. Similarly, since $(m_i^2; w_i^1) = (\ ;\ )$ for all $i = 1;::::;k$, there is no alternating path that contains unmatched node m_i^4 and $(m_i^1; w_i^0) = (+;+)$ edge for any $i = 1;::::;k$. This implies that M_2 is popular. Since $jM_1j < jM_2j < jM_0j$, we conclude that M_2 is a middle matching.

Second, since nodes $m_i^4; m_i^3$ for all $i = 1;::::;k$ are unmatched, they cannot be part of any coalition cycle. Since m_i^2 is matched to its top choice for all $i = 1;::::;k$, and nodes m_i^0, $i = 1;::::;k$ have degree one, they cannot be in any coalition cycle. Consider nodes m_i^1, $i = 1;::::;k$. Assume there is $i \ 2 \ 1;::::;k$ such that m_i^1 is in

a coalition cycle C. Since $(m_i^1; w_i^0); (m_j^1; w_i^0)$ are unmatched for all $j = 1; : : : ; k$, we conclude that $(m_i^1; w_j^0); (m_i^1; w_i^1)$ 2 C, and consequently $(m_i^2; w_i^1)$ 2 C. Since $(m_i^2; w_i^1) = (\ ;\ )$, C is not a coalition cycle. This implies that M_2 is Pareto-optimal.

□

Lemma 4.3.8. *Any Pareto-optimal matching M contains at most one edge from $f(m_i^1; w_i^0) : i = 1 : : : kg$ and at most one edge from $f(m_i^3; w_i^3) : i = 1 : : : kg$.*

Proof. Assume by contradiction there is a Pareto-optimal matching M that contains two edges $(m_i^1; w_i^0); (m_j^1; w_j^0)$ from $f(m_i^1; w_i^0) : i = 1 : : : kg$. Since w_i^0 and w_j^0 are the most preferred choices for m_j^1 and m_i^1 correspondingly, then there is a coalition cycle $m_i^1; w_i^0; m_j^1; w_j^1$. This contradicts the Pareto-optimality of M.

Similarly, assume by contradiction there is a Pareto-optimal matching M that contains two edges $(m_i^3; w_i^3); (m_j^3; w_j^3)$ from $f(m_i^3; w_i^3) : i = 1 : : : kg$. Since w_i^3 and w_j^3 are the most preferred choices for m_i^3 and m_j^3 correspondingly, then there is a coalition cycle $m_i^3; w_i^3; m_j^3; w_j^3$. This contradicts the Pareto-optimality of M. □

Lemma 4.3.7 implies that $\Gamma = 0$ achieved on a middle matching M_2. From Lemma 4.3.6 and Lemma 4.3.8 we conclude that $\Delta(M_1)$ is linear in k, which is linear in the number of men n. Similarly, from Lemma 4.3.5 and Lemma 4.3.8 we conclude that $\Delta(M_0)$ is linear in n. Since M_1 and M_0 are the only stable and dominant matchings, we conclude that they are the only not middle matchings.

Γ jAj 2

We now construct an infinite family of instances, each with exactly one popular matching M_0, and such that M_0 has at most two common edges with any Pareto-optimal matching. Hence, for this instance, Γ jAj 2.

Our example will be constructed from the same kind of blocks (or gadgets) that are connected with additional edges. One block i consists of three men $m_i^0; m_i^1; m_i^2$ and two women $w_i^0; w_i^1$, Fig. 4.4. Their preferences lists are the following:

$m_i^0 : \quad w_i^1; w_i^0$ $\qquad\qquad$ $w_i^0 : \quad m_i^0; m_i^1; m_i^2$

$m_i^1 : \quad w_i^0; w_i^1$ $\qquad\qquad$ $w_i^1 : \quad m_i^1; m_i^2; m_i^0$

$m_i^2 : \quad w_i^1; w_i^0$

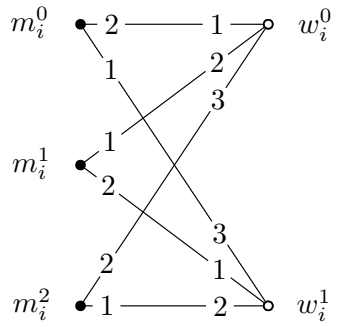

Figure 4.4: One block i consists of three men $m_i^0; m_i^1; m_i^2$ and two women $w_i^0; w_i^1$ with their preferences labeled on the edges.

Consider an instance I with k 2 blocks with $3k$ men and $2k$ women. Add $2k(k$ 1) edges: $(m_i^0; w_j^0)$ and $(m_i^1; w_j^1)$ for any $i; j = 1; : : : ; k$, $i \neq j$. For each man m_i^0 and m_i^1 add the k 1 women to his preference lists between his first and second choices in the gadget, in arbitrary order. For each women w_i^0 and w_i^1 add k 1 men to the end of her preferences lists in arbitrary order.

Lemma 4.3.9. *There is exactly one popular matching $M_0 = f(m_i^0; w_i^0); (m_i^1; w_i^1) : i = 1 : : : kg$ in the instance I.*

Proof. First, we will prove that no popular matching contains one of the edges $(m_i^0; w_j^0)$ or $(m_i^1; w_j^1)$, $i \neq j$ 2 $[k]$.

By contradiction, assume there is a popular matching M that contains an edge $(m_i^1; w_j^1)$, $i \neq j$, for some $i; j$ 2 $f1; : : : ; kg$. Since all new edges during the construction were added in the end of w_j^1 list, then w_j^1 prefers m_j^0, m_j^1 and m_j^2. We

conclude that $(m_j^2; w_j^0) \in M$, otherwise $(m_j^2; w_j^1) = (+;+)$ is adjacent to an unmatched node m_j^2, which would contradict popularity. A path $m_j^1; w_j^0; m_j^2; w_j^1$ contains $(m_j^1; w_j^0); (m_j^2; w_j^1) = (+;+)$. This contradicts popularity of M and implies that $(m_i^1; w_j^1) \notin M$, for $i \neq j \in [k]$ and all popular matchings M.

Similarly, by contradiction, assume there is a popular matching M that contains edge $(m_i^0; w_j^0)$, for some $i \neq j \in [k]$. Since all new edges during the construction were added in the end of w_j^0 list, then w_j^0 prefers m_j^0, m_j^1 and m_j^2 to m_j^0. We conclude that $(m_j^2; w_j^1) \in M$, otherwise $(m_j^2; w_j^0) = (+;+)$ is adjacent to an unmatched node m_j^2, which would contradict popularity. Consider an edge $(m_j^1; w_j^1)$. In the first part we concluded that $(m_j^1; w_\ell^1) \notin M$ for all $\ell \neq j$. This implies that $(m_j^1; w_j^0) = (+;+)$ is adjacent to an unmatched node m_j^1, which contradicts popularity of M and implies that $(m_i^0; w_j^0) \notin M$, for $i \neq j \in [k]$ and all popular matchings M.

We next show that there is no popular matching that contains an edge $(m_i^0; w_i^1)$ or $(m_i^2; w_i^1)$ for any $i \in [k]$.

By contradiction, assume there is a popular matching M that contains an edge $(m_i^0; w_i^1)$, for some $i \in [k]$. We conclude that $(m_i^2; w_i^0) \in M$, otherwise $(m_i^2; w_i^1) = (+;+)$ is adjacent to an unmatched node m_i^2, which would contradict popularity. Observe that $(m_i^0; w_i^0)$ is a $(\ ;+)$ edge. Cycle $m_i^2; w_i^0; m_i^0; w_i^1$ contains $(m_i^2; w_i^1) = (+;+)$, which contradicts popularity.

Similarly, by contradiction, assume there is a popular matching M that contains an edge $(m_i^2; w_i^1)$, for some $i \in \{1; \ldots; k\}$. We conclude that $(m_i^1; w_i^0) \in M$, otherwise $(m_i^1; w_i^1) = (+;+)$ is adjacent to an unmatched node m_i^1, which would contradict popularity. Notice that $(m_i^0; w_i^0) = (+;+)$ and m_i^0 is unmatched, since we proved earlier that $(m_i^0; w_\ell^0) \notin M$, for all $\ell \neq i$. This contradicts popularity. We conclude that $(m_i^0; w_i^1); (m_i^2; w_i^1) \notin M$ for any $i \in \{1; \ldots; k\}$ and any popular matching M.

Notice that w_i^1 is matched in any popular matching M for all $i \in \{1; \ldots; k\}$, otherwise $(m_i^0; w_i^1) = (+;+)$ is adjacent to unmatched node w_i^1, which would contradict

popularity. From above $(m_j^1; w_i^1) \notin M$ for all $i; j \in \{1; \ldots; k\}$ and all popular matchings M, and $(m_i^2; w_j^1) \notin M$, we conclude that $(m_i^1; w_i^1) \in M$ for all $i \in \{1; \ldots; k\}$. Similarly, $(m_i^0; w_i^0) \in M$ for all $i \in \{1; \ldots; k\}$, otherwise $(m_i^0; w_i^0) = (+; +)$ is adjacent to unmatched node m_i^0.

We conclude that edges $(m_i^0; w_i^0); (m_i^1; w_i^1); i = 1 \ldots k$ belong to all popular matchings. Since there is no any other combinations for matched edges, and popular matchings exist in every bipartite instance, there is only one popular matching M_0. This concludes the proof. □

Lemma 4.3.10. *Any Pareto-optimal matching M_1 contains at most one edge from $f(m_i^0; w_i^0) : i = 1 \ldots k$ and at most one edge from $f(m_i^1; w_i^1) : i = 1 \ldots k$.*

Proof. Assume by contradiction there is a Pareto-optimal matching M that contains two edges $(m_i^0; w_i^0); (m_j^0; w_j^0)$ from $f(m_i^0; w_i^0) : i = 1 \ldots k$. Since w_i^0 and w_j^0 are the least preferred choice for m_i^0 and m_j^0 correspondingly, then there is a coalition cycle $m_i^0; w_i^0; m_j^0; w_j^0$. This contradicts to Pareto-optimality of M.

Similarly, assume by contradiction there is a Pareto-optimal matching M that contains two edges $(m_i^1; w_i^1); (m_j^1; w_j^1)$ from $f(m_i^0; w_i^0) : i = 1 \ldots k$. Since w_i^1 and w_j^1 are the least preferred choice for m_i^1 and m_j^1 correspondingly, then there is a coalition cycle $m_i^1; w_i^1; m_j^1; w_j^1$. This contradicts to Pareto-optimality of M. □

Lemma 4.3.9 and 4.3.10 implies that the popular matching M_0 has at most two common edges with any Pareto-optimal matching. Since all men are matched in M_0, we conclude that $\Gamma \quad jAj \quad 2$.

4.4 Two easy cases

In this Theorem, we show polynomial-time results. We start with the proof of Theorem 4.1.4.

Proof. We first deal with the computation of Δ. Consider the following algorithm:

1. For $t; \grave{} \ 2 \ [k]$:
   a) Guess two disjoint sets of edges, say E^{in} and E^{out}, of size respectively t and $\grave{}$, such that the number of men incident to $E^{\text{in}} \ [\ E^{\text{out}}$ is at most k;
   b) Compute the man-optimal stable matching M in $S(E^{\text{in}}; E^{\text{out}})$, see Lemma 1.3.1 and Corollary 1.3.1.
   c) if $M^{\emptyset} := (M \ n \ E^{\text{in}}) \ [\ E^{\text{out}}$ is a Pareto-optimal matching, output $(M; M^{\emptyset})$ and stop.
2. output $\Delta(G) > k$ and stop.

We now show that the algorithm above is correct and runs in time $jA \ [\ Bj^{O(k^2)}$.

The running time follows from the fact that there are $jA \ [\ Bj^{O(k)}$ sets of edges of size at most k.

For the correctness, suppose first $\Delta(G) \quad k$. Then, there exists a stable matching M and a Pareto-optimal matching $M^{\emptyset}$ such that $\Delta(M; M^{\emptyset}) \quad k$. Now let $t := jM \ n \ M^{\emptyset}j \quad k$ and $\grave{} = jM^{\emptyset} \ n \ Mj \quad k$ and let $E^{\text{in}} := M \ n \ M^{\emptyset}$, $E^{\text{out}} := M^{\emptyset} \ n \ M$. Note that $t; \grave{}; E^{\text{in}}; E^{\text{out}}$ is a valid guess of the algorithm.

We claim that M is a man-optimal stable matching in $S(E^{\text{in}}; E^{\text{out}})$. Suppose not, then, by Lemma 1.3.1, there exists a matching $\overline{M} \ 2 \ S(E^{\text{in}}; E^{\text{out}})$ with $\overline{M} > M$. Because of Theorem 1.3.1, graph $G(A \ [\ B; \overline{M} \quad M)$ is a collection of singletones and cycles whose edges alternate between M and $\overline{M}$. Take one such cycle C. Note that C does not contain nodes adjacent to edges of $E^{\text{in}} \ [\ E^{\text{out}}$. Since $\overline{M} > M$, $\overline{M}(m) > M(m)$ for every $m \ 2 \ V(C) \setminus A$. Hence, C is a coalition cycle for M, and, as it contains no node adjacent to edges of $E^{\text{in}} \ [\ E^{\text{out}}$, it is also a coalition cycle for $M^{\emptyset}$, contradicting the fact that $M^{\emptyset}$ is Pareto-optimal.

Therefore, M is the man-optimal stable matching in $S(E^{\text{in}}; E^{\text{out}})$. Hence, the pair $M; M^{\emptyset}$ is constructed during some iteration of the algorithm.

Conversely, suppose the algorithm outputs a pair $(M; M')$. Then by construction M is stable, while M' is Pareto-optimal since the algorithm verifies so. We have:

$$\Delta(M; M') = \#\text{ men that have a different partner in } M \text{ and } M' \quad k$$

by construction, hence the output of the algorithm is correct.

Now for the computation of Λ, repeat the algorithm above after modifying (c) as follows.

(c') if $M' := (M \; n \; E^{\text{in}}) \; [\; E^{\text{out}} \; 2 \; PO(M)$, output $(M; M')$ and stop.

The correctness of the algorithm follows as above, after observing that, if $\Delta(M; M') = \Lambda$, then M is again a man-optimal stable matching in $S(E^{\text{in}}; E^{\text{out}})$. □

We next show that, for a fixed Pareto-optimal matching M, $\Delta(M)$ can be computed efficiently.

Lemma 4.4.1. *There is a polynomial-time algorithm that, given a marriage instance $(G; <)$ and a Pareto-optimal matching M, computes $\Delta(M)$ and the stable matching achieving it.*

Proof. Consider a preference system $(G(A \; [\; B; E); <)$ and a Pareto-optimal matching M. Assign weight 0 to all edges that belong to M, and weight 1 to all other edges. We now claim that

$$\arg \min_{M' 2 S} \Delta(M; M') = \arg \min_{M' 2 S} w(M'): \tag{4.5}$$

Indeed, let $M' \; 2 \; S$. Note that:

$$\Delta(M; M') = \#\text{ men that have a different partner in } M \text{ and } M' = w(M') + jA(M n M')j:$$

As by Theorem 1.3.1, $|A(M \oplus M')|$ does not depend on the choice of M', and the claim follows. The thesis follows from the fact that a stable matching of minimum weight can be computed in polynomial time, see Theorem 1.3.3. □

4.5 Proof of Theorem 4.1.3

In this section, we prove Theorem 4.1.3. For the reader's convenience, the theorem is restated below.

Theorem 4.1.3. *There exists a universal constant c such that the following holds. Unless P=NP, for each $\varepsilon > 0$, there is no polynomial-time algorithm that, given a marriage instance $(G(A \cup B, E), <)$, accepts if $\Delta(G) \leq |A|$, rejects if $\Delta(G) \geq c|A|$, and is free to accept or reject if none of these cases apply. In particular, computing $\Delta(G)$ is NP-hard.*

The main idea of the proof is to construct a map from instances of `Monotone 3-SAT` to marriage instance $(G(A \cup B, E), <)$ such that if is satisfiable then Δ is small, but if is not satisfiable, there are many disjoint coalition cycles that results in Δ being (relatively) large.

For a fixed , the instance of the marriage problem is constructed from gadgets corresponding to literals and some edges that connect them. The following is the description of a gadget that corresponds to one literal. Positive and negative literals have the same associated gadgets, but the true and false assignments correspond to different stable matchings in those gadgets. Nodes of the gadget that correspond to literal ℓ_i are labeled as $m_j(\ell_i)$ or $w_k(\ell_i)$, for integers j and k. Sometime we omit the dependency on ℓ_i if it is clear to which gadget the nodes belong.

One gadget consists of $2l + 8$ men and $l + 6$ women (for some l to be fixed later on), see Fig. 4.5. Fig. 4.6 and Fig. 4.7 provide an example of the construction.

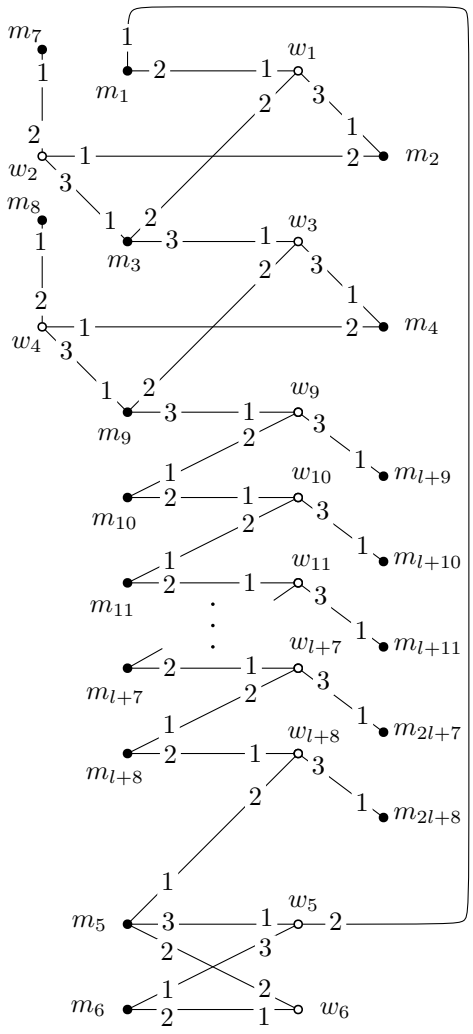

Figure 4.5: One gadget.

If ℓ_i and ℓ_j are literals that correspond to a variable and its negation respectively, then the corresponding gadgets are connected by extra edges $m_1(\ell_i)w_5(\ell_j)$. For man $m_1(\ell_i)$, add woman $w_5(\ell_j)$ to the beginning of his preference list, and for a woman $w_5(\ell_j)$, add man $m_1(\ell_i)$ after her the first choice $m_5(\ell_j)$.

For a clause with exactly two positive literals $\ell_1;\ell_2$, add edges $m_{i+1}(\ell_1)w_i(\ell_2)$ and $m_{i+1}(\ell_2)w_i(\ell_1)$ for $i = 9; 10; :::; l + 7$. For men $m_{i+1}(\ell_1); m_{i+1}(\ell_2)$ add women $w_i(\ell_1); w_i(\ell_2)$, $i = 9; 10; :::; l + 7$, to the top of their preferences list respectively. For women $w_i(\ell_1); w_i(\ell_2)$ add men $m_{i+1}(\ell_1); m_{i+1}(\ell_2)$, $i = 9; 10; :::; l + 7$, to the end of their preferences list respectively. If the clause has three positive literals $\ell_1;\ell_2;\ell_3$,

add edges $m_{i+1}(\ell_1)w_i(\ell_3)$, $m_{i+1}(\ell_2)w_i(\ell_1)$ and $m_{i+1}(\ell_3)w_i(\ell_2)$ for $i = 9, 10, \ldots, l+7$. The preference lists are modified similarly to the case with two literals in a clause.

For a clause with exactly two negative literals ℓ_1, ℓ_2, add edges $m_i(\ell_1)w_i(\ell_2)$ and $m_i(\ell_2)w_i(\ell_1)$ for $i = 9, 10, \ldots, l+8$. For men $m_i(\ell_1), m_i(\ell_2)$ add women $w_i(\ell_1), w_i(\ell_2)$, $i = 10, \ldots, l+8$, after their top choice in the preference lists respectively. For women $w_i(\ell_1), w_i(\ell_2)$ add men $m_i(\ell_1), m_i(\ell_2)$, $i = 9, 10, \ldots, l+8$, to the end of their preferences list respectively. If the clause has three negative literals ℓ_1, ℓ_2, ℓ_3, add edges $m_i(\ell_1)w_i(\ell_3)$, $m_i(\ell_2)w_i(\ell_1)$ and $m_i(\ell_3)w_i(\ell_2)$ for $i = 9, 10, \ldots, l+8$. The preference lists are modified similarly to the case with two literals in a clause.

Lemma 4.5.1. *Any stable matching M of $(G(A \cup B, E), <)$ does not contain any edges between the gadgets. Moreover, the set of stable nodes of G in each gadget is $\bigcup_{i=1,\ldots,6,9,\ldots,l+8}\{m_i, w_i\}$.*

Proof. First, we prove that edges between the gadgets in the same clause are not stable. Assume there is an edge $m_i(\ell_j)w_i(\ell_k) \in M$ between two negative literals ℓ_j and ℓ_k. This implies that $m_{l+i}(\ell_k)w_i(\ell_k) = (+,+)$ and M is not stable. Similarly, we argue that edges between the gadgets in a positive clause are not stable.

Notice, that $M^{\emptyset} = \{m_i w_i : i = 1, 2, \ldots 6, 9, \ldots, l+8\}$ for all gadgets is stable, since every woman is matched to her favorite partner. We conclude that the set of stable nodes in each gadget is $\bigcup_{i=1,\ldots,6,9,\ldots,l+8}\{m_i, w_i\}$. Edges m_3w_2 and m_9w_4 are not stable in all gadgets, since otherwise $m_7w_2 = (+,+)$ and $m_8w_2 = (+,+)$.

Second, we prove that edges connecting two literals that correspond to a variable and its negation are not stable. Assume by contradiction that there are two literals ℓ_1 and ℓ_2 that correspond to a variable and its negation respectively, and $m_1(\ell_1)w_5(\ell_2) \in M$. Since nodes $m_6(\ell_2), m_5(\ell_2), m_{l+8}(\ell_2), \ldots, m_9(\ell_2), m_3(\ell_2)$ are matched in $M^{\emptyset}$, and edges between the gadgets and m_3w_2, m_9w_4 are not stable, then $m_5(\ell_2)\, w_{l+8}(\ell_2), m_{l+8}(\ell_2)w_{l+7}(\ell_2), \ldots, m_{10}(\ell_2)w_9(\ell_2), m_9(\ell_2)w_3(\ell_2), m_3(\ell_2)w_1(\ell_2) \in M$.

This leaves node $m_1(\grave{}_2)$ unmatched, a contradiction, since $m_1(\grave{}_2)$ is matched in $M^{\emptyset}$, and all stable matchings match the same set of nodes, see Theorem 1.3.1. □

Lemma 4.5.2. *Let M be a stable matching of $(G(A \cup B; E); <)$. Then in each positive gadget H (the one that corresponds to a positive literal), exactly one of the following is true (dependency on $\grave{}$ is omitted):*

$$T(H) := \{m_i w_i : i = 1; 2; :::6; 9; :::; l+8\}, \text{ or}$$
$$F(H) := \{m_6 w_6; m_5 w_{l+8}; m_{l+8} w_{l+7}; :::; m_{10} w_9; m_9 w_3; m_3 w_1; m_1 w_5; m_2 w_2; m_4 w_4\}.$$

Similarly, in each negative gadget N (the one that corresponds to a negative literal), exactly one of the following is true (dependency on $\grave{}$ is omitted):

$$F(N) := \{m_i w_i : i = 1; 2; :::6; 9; :::; l+8\}, \text{ or}$$
$$T(N) := \{m_6 w_6; m_5 w_{l+8}; m_{l+8} w_{l+7}; :::; m_{10} w_9; m_9 w_3; m_3 w_1; m_1 w_5; m_2 w_2; m_4 w_4\}.$$

Proof. In each gadget the set of stable nodes is $\bigcup_{i=1;:::;6;9;:::;l+8} \{m_i; w_i\}$, Lemma 4.5.1. There are two options for matching node w_5 reagrdles the type of the gadget: $m_5 w_5$ and $m_1; w_5$. The corresponding stable matchings are $\{m_i w_i : i = 1; 2; :::6; 9; :::; l+8\}$ and $\{m_6 w_6; m_5 w_{l+8}; m_{l+8} w_{l+7}; :::; m_{10} w_9,\ m_9 w_3; m_3 w_1; m_1 w_5; m_2 w_2; m_4 w_4\}$. □

Lemma 4.5.3. *Let M be a stable matching of $(G(A \cup B; E); <)$. Let $\grave{}_1$ be a positive and $\grave{}_2$ a negative occurrence of the same variable. Then exactly one of the following is true:*

(a) $T(H(\grave{}_1)) \quad M$;

(b) $T(N(\grave{}_1)) \quad M$.

Conversely, each maximal matching M that satisfy exactly one of (a)-(b) above for each pair of positive $\grave{}_1$ and negative $\grave{}_2$ occurrence of the same variable is stable.

Proof. Proof by contradiction. Assume $T(H(\grave{}_1)) \quad M$ and $T(N(\grave{}_1)) \quad M$. This implies that $m_7(\grave{}_1) w_5(\grave{}_2) = (+; +)$, contradicting the stability of M.

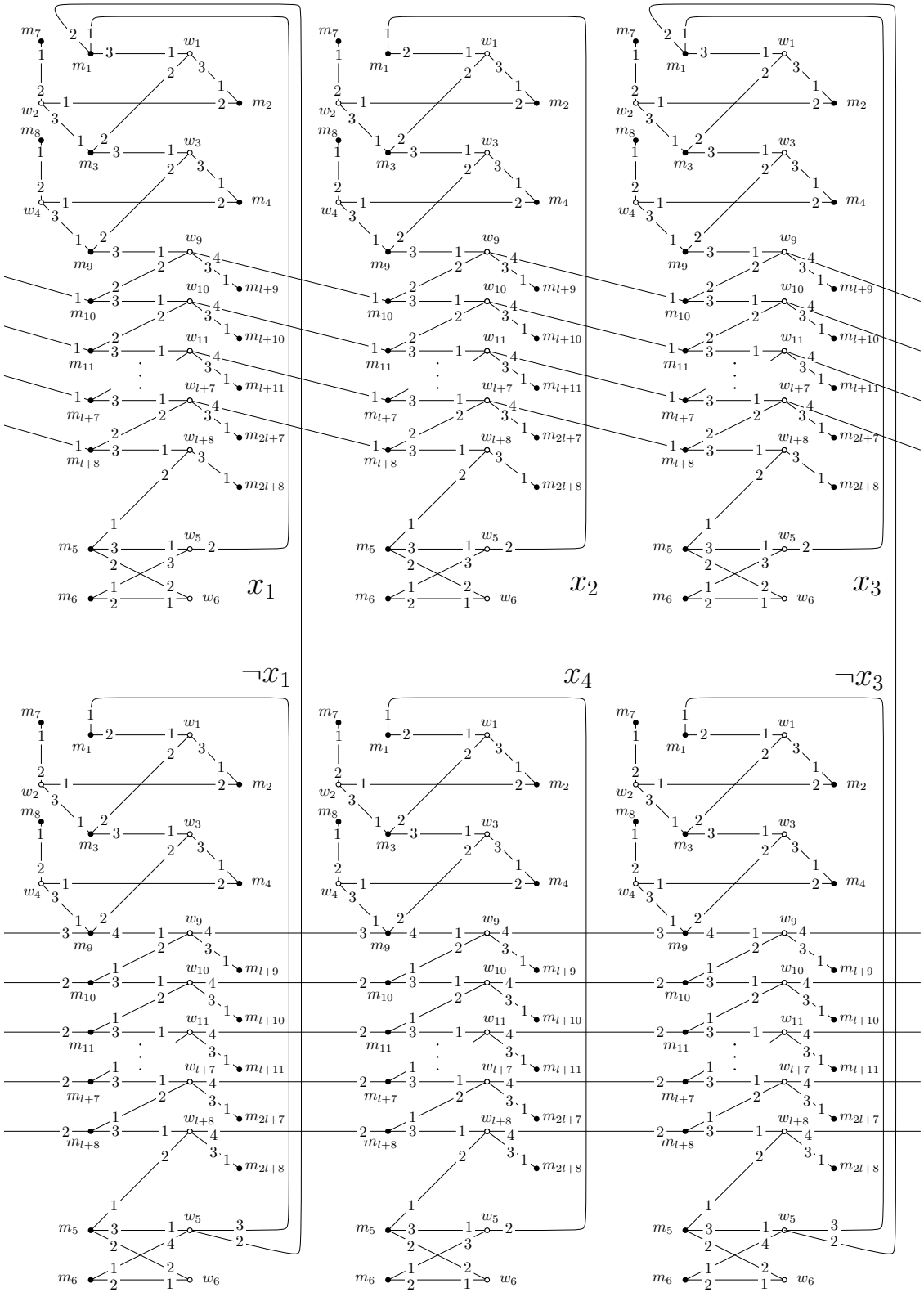

Figure 4.6: Corresponding graph for $\quad = (x_1 \_ x_2 \_ x_3) \wedge (\neg x_1 \_ \neg x_4 \_ \neg x_3)$.

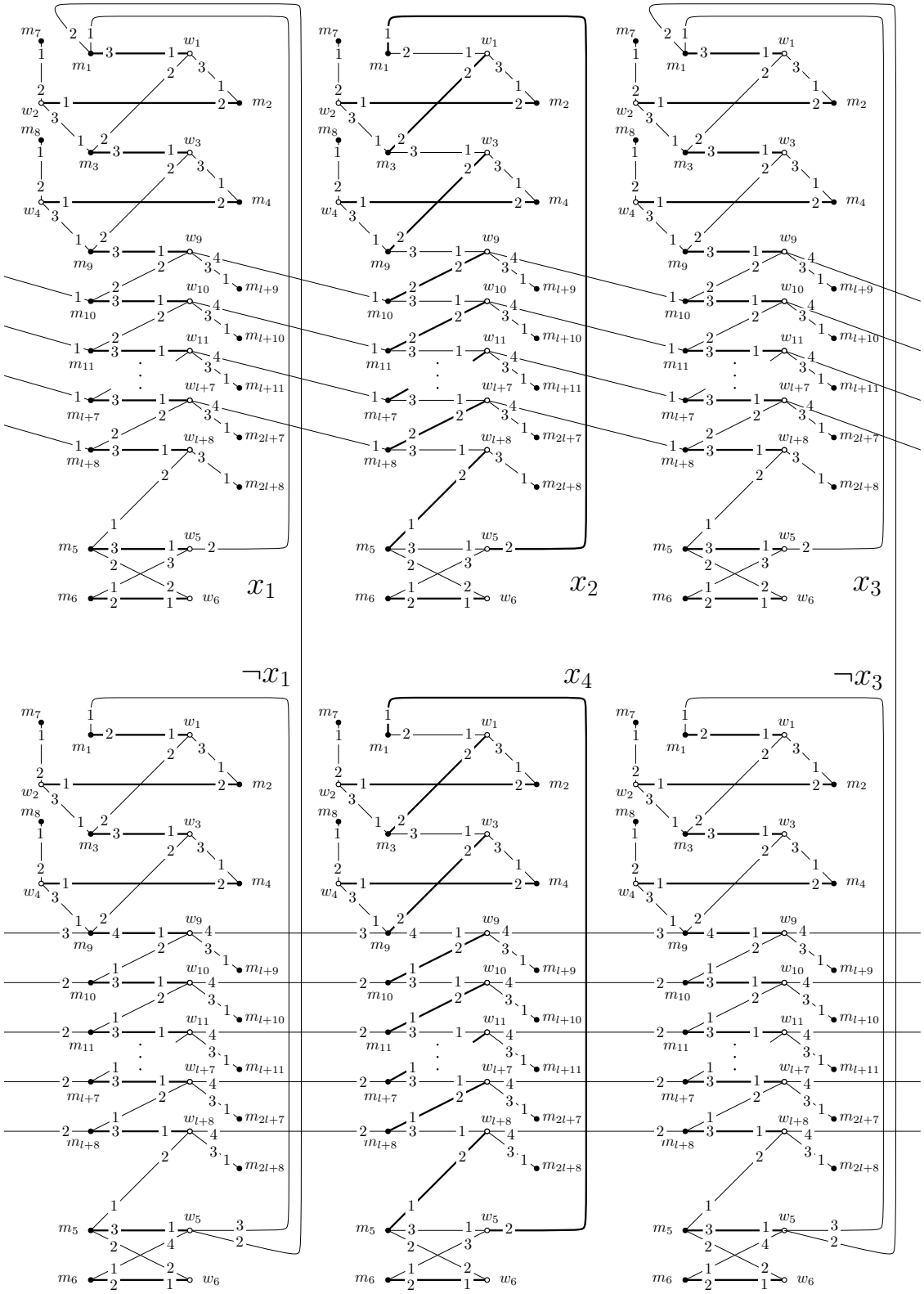

Figure 4.7: Corresponding stable matching (bold edges) for = (x_1 _ x_2 _ x_3) ^ (: x_1 _ : x_4 _ : x_3), where $x_1; x_3 = T; x_2; x_4 = F$.

Let M be a maximal matching that satisfy exactly one of (a)-(b) above for each pair of positive $`_1$ and negative $`_2$ occurrence of the same variable. We show that there no $(+;+)$ edges in this case. Assume there is a blocking edge. For any $`$, $T(H(`))$ and $F(H(`))$ do not contain $(+;+)$ edges. For all gadgets $w_9; w_{10}; :::; w_{l+8}$ prefer the least men from other gadgets. We conclude that only edge $m_1(`_1)w_5(`2)$ could be $(+;+)$, when both $`_1$ and $`_2$ are assigned to true. This contradicts to the statement that $M$ satisfy exactly one of (a)-(b) for each pair of positive $`_1$ and negative $`_2$ occurrence of the same variable. □

To a consistent assignment , we associate a matching $M = M(\ )$ defined as follows: $T(H(`))$ $M$ if literal $`$ is set to true, $F(H(`))$ $M$ otherwise; similarly, $T(N(`))$ M if literal $`$ is set to true, $F(N(`))$ M otherwise. Conversely, to a stable matching M, because of Lemma 4.5.3, we can associate an assignment where a positive (resp. negative) literal $`$ is set to true if and only if $T(H(`)$ M (resp. $T(N(`)$ M).

Corollary 4.5.1. *Let be a consistent assignment, then the corresponding matching M is stable. Conversely, let M be a stable matching of $(G(A [B; E); <)$, then the corresponding assignment is consistent.*

Associate to an assignment the set $M^{\emptyset}(\ ) = M^{\emptyset} = [\cdot M^{\emptyset}$, where:

If variable x is set to true, then for each literal $`$ (both positive and negated) corresponding to x, set

$$M^{\emptyset} := f m_i w_i : i = 1; 2; :::4; 9; :::; l + 8; m_5 w_6; m_6 w_5 g$$

(dependence on $`$ is omitted from the nodes);

If variable x is set to false, the for each literal ℓ (both positive and negated) corresponding to x, set

$$M^{\ell} := \{m_6w_6, m_5w_{l+8}, m_{l+8}w_{l+7}, \ldots, m_{10}w_9, m_9w_3, m_3w_1, m_1w_5, m_7w_2, m_8w_4\}$$

(dependence on ℓ is omitted from the nodes).

One easily verifies that M^{ℓ} is a matching of G and it is maximal.

Lemma 4.5.4. *Let be a satisfying assignment. Then M^{ℓ}( ) is Pareto-optimal.*

Proof. First, we prove that there are no coalition cycles that are contained in one gadget. Consider a matching M^{ℓ} that corresponds to a variable x that is set to false. In this case the statement follows from the fact that there are only two edges $m_9w_4, m_6w_5 = (+,)$ that are adjacent to a matched men m_9 and m_6 that prefer w_4 and w_5 to their partners respectively. There is also one edge $m_2w_2 = (+,+)$ adjacent to unmatched node m_2, so it can not be a part of any coalition cycles. Suppose by contradiction that m_9w_4 is part of a coalition cycle C. Then also $m_8 \in C$, which is impossible. Similarly, if m_6w_6 is part of a coalition cycle C, then also $m_5 \in C$, also impossible. Consider a matching M^{ℓ} that corresponds to a variable x that is set to true. Similarly to above, there are only six edges $m_1w_5, m_3w_2, m_3w_1, m_9w_4, m_9w_3, m_5w_{l+8} = (+,); (+,+)$ that can be used to form a coalition cycle, but there is no cycle that these edges form.

Now we prove that edges between two literals in different clauses cannot be in any coalition cycle. This follows from the fact that in a matching M^{ℓ} that corresponds to a variable x that is set to false in positive clause, $m_1w_5 = (,+)$, and in a matching M^{ℓ_0} that corresponds to a variable x that is set to true in negative clause (ℓ_0 is set to false), $m_6w_6 = (,+)$.

Next, we prove that there are no coalition cycles that are contained in gadgets corresponding to one clause. Consider a positive clause with three literals ℓ_1, ℓ_2, ℓ_3

in this order. Without loss of generality assume $\grave{}_1$ is assigned to false, and $\grave{}_2; \grave{}_3$ are assigned to true. Assume there is a coalition cycle C and $m_i(\grave{}_1)w_{i\ 1}(\grave{}_1)\ 2\ C$. It follows that $m_i(\grave{}_1)w_{i\ 1}(\grave{}_3); m_i(\grave{}_2)w_{i\ 1}(\grave{}_1); m_{i\ 1}(\grave{}_3)w_{i\ 1}(\grave{}_3); m_i(\grave{}_2)w_i(\grave{}_2)\ 2\ C$. Since there is no other path between $m_{i\ 1}(\grave{}_3)$ and $w_i(\grave{}_2)$, we obtain a contradiction to C being a cycle. The same argument is applied if there are two literals assigned to false, or if there are two literals in total. Consider a positive clause with three literals $\grave{}_1; \grave{}_2; \grave{}_3$. Since all men in a gadget that corresponds to true assignment $m_9; m_10; :::; m_{l+8}$ are matched to their top choice, there could not be a coalition cycle that includes edges between the gadgets.

Since all women are matched, we conclude that $M^0(\ )$ is trade-in-free and maximal. From Lemma 1.3.4 it follows that $M^0(\ )$ is Pareto-optimal. □

Lemma 4.5.5. *Let be a satisfying instance. Then* $\Delta(G)$ $4p_2 + 6p_3$, *where* p_2 *and* p_3 *are the number of clauses with two and three literals respectively.*

Proof. Consider the Pareto-Optimal matching M^0 and stable matching M corresponding to a satisfying assignment , as defined above. Now fix a gadget H. By construction, $\Delta_H(M; M^0) = 2$. Hence, $\Delta_C(M; M^0) = 4$ for each clause C with two literals and $\Delta_C(M; M^0) = 6$ for each clause C with three literals. Therefore:

$$\Delta(G) \quad \Delta(M) \quad \Delta(M; M^0) = 4p_2 + 6p_3:$$

□

Lemma 4.5.6. *Let* p_2 *and* p_3 *be the number of clauses with two and three literals respectively. If any consistent assignment to has at least* k 6 0 *unsatis ed clauses, then* $\Delta(G)$ $k(l + 8)$.

Proof. Based on Corollary 4.5.1 there is a bijection between stable matchings and consistent assignments. Consider any stable matching M and its corresponding as-

signment . Since leaves k unsatisfied clauses, then there are k clauses that has all literals assigned to false. Consider an unsatisfied positive clause C with three literals ℓ_1, ℓ_2, ℓ_3. Edges $m_i(\ell_1)w_{i\ 1}(\ell_1), m_i(\ell_2)w_{i\ 1}(\ell_2), m_i(\ell_3)w_{i\ 1}(\ell_3)$, for $i = 10, 11, \ldots, l+8$, form $l+8 \quad 10+1 = l \quad 1$ disjoint coalition cycles. From Lemma 4.2.4, this implies $\Delta_C(M) \quad l \quad 1$.

Consider now an unsatisfied negative clause with three literals ℓ_1, ℓ_2, ℓ_3. Edges $m_i(\ell_1)w_i(\ell_1), m_i(\ell_2)w_i(\ell_2), m_i(\ell_3)w_i(\ell_3)$, for $i = 9, 11, \ldots, l+8$, form $l+8 \quad 10+1 = l \quad 1$ disjoint coalition cycles, and again using Lemma 4.2.4 we can conclude $\Delta_C(M) \quad l \quad 1$. A similar argument holds if C has 2 literals instead of 3. Hence, $\Delta_C(M) \quad l \quad 1$ for any stable matching M and any clause C that is not satisfied in the consistent assignment corresponding to M. Since any consistent assignment leaves at least k clauses unsatisfied, the thesis follows. □

Using the hardness of approximation for MONOTONE 3-SAT, we can deduce the proof of Theorem 4.1.3.

Proof. Let be an instance of the MONOTONE 3-SAT with n variables and $m = O(n)$ clauses, and consider the construction from this section, taking $\ell = \Theta(n\ )$ with $= \frac{1}{}$. Hence, $|A| = \Theta(n^{\ +1})$.

If is satisfiable, then from Lemma 4.5.5 we have:

$$\Delta(G) \quad 4p_2 + 6p_3 < c'|A|$$

for an appropriate constant $c' > 0$.

Conversely, if every consistent assignment leaves at least $k = (\frac{1}{8} + \ )m$ clauses of unsatisfied, we deduce:

$$\Delta(G) \quad k(l+8) > c|A|$$

for an appropriate constant $c > 0$.

Now suppose that an algorithm A as in the hypothesis of the theorem holds. We construct an algorithm A^0 contradicting the hardness result from Theorem 1.4.2.

Indeed, suppose algorithm A accepts. This implies $\Delta(G) < cjAj$, hence, we can deduce that there exists an assignment that satisfies at least $(\frac{7}{8} + \ )m$ clauses from

and we can make A^0 accept. Conversely, if A rejects, then $\Delta(G) \quad c^0jAj$ hence is not satisfiable, and we can make A^0 reject. □

Computing $\Delta(M)$ for a stable matching M is NP-Hard

We showed in Section 4.4 that $\Delta(M)$ for a Pareto-optimal matching M can be computed in polynomial time. We show here that the "symmetric" problem, that is, computing $\Delta(M)$ for a stable matching M is NP-Hard.

Theorem 4.5.1. *Given a marriage instance $(G; <)$ and a stable matching M, computing $\Delta(M)$ is NP-hard.*

In this section we prove Theorem 4.5.1 by reduction from the Maximum independent set problem.

Consider the `Maximum independent set` problem for a graph G. We construct a corresponding marriage instance in the following way. For each vertex i of G there is an instance $(G_i; \ )$ that we will call a block. One block i consists of three men $m_i^0; m_i^1; m_i^2$ and three women $w_i^0; w_i^1; w_i^2$, Fig. 4.8. Their preferences lists are the following:

$$m_i^0 : \quad w_i^2 \qquad\qquad w_i^0 : \quad m_i^2$$

$$m_i^1 : \quad w_i^2; w_i^1 \qquad\qquad w_i^1 : \quad m_i^2; m_i^1$$

$$m_i^2 : \quad w_i^0; w_i^2; w_i^1 \qquad\qquad w_i^1 : \quad m_i^2; m_i^1; m_i^0$$

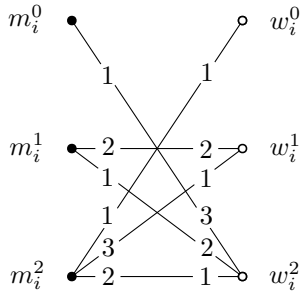

Figure 4.8: One block i consists of three men $m_i^0; m_i^1; m_i^2$ and and three women $w_i^0; w_i^1; w_i^2$ with their preferences labeled on the edges.

If two vertices i and j from G are adjacent, then we add the following extra edges to the marriage instance: $(m_i^1; w_j^2)$ and $(m_j^1; w_i^2)$. Correspondingly, for each man m_i^1 and m_j^1 add w_j^2 and w_i^2 to the beginning of his preference list respectively. For each women w_i^2 and w_j^2 add m_j^1 and m_i^1 to the end of her preferences list respectively.

Lemma 4.5.7. *There is exactly one stable matching $M_0 = f(m_i^1; w_i^2); (m_i^2; w_i^0) : i = 1:::jG(V)jg$ in the instance I.*

Proof. Notice that edges $(m_i^1; w_j^2)$ **2** M for any i **6=** j, otherwise $(m_j^0; w_j^2) = (+; +)$. We conclude that $(m_i^2; w_i^0)$**;** **2** M, since w_i^0 is the top choice for m_i^2. Edge $m_i^0 w_i^2$ **2** M, otherwise $m_i^1 w_i^2 = (+; +)$. Since w_i^2 has to be matched, then $(m_i^1; w_i^2)$**;** **2** M. □

If there are no extra edges, or all vertices in graph G disconnected,then $M_0 = f(m_i^1; w_i^2); (m_i^2; w_i^0) : i = 1:::jG(V)jg$ is also Pareto-optimal, since there is no coalition cycle. We call a block with edges $(m_i^1; w_i^2); (m_i^2; w_i^0)$ a *stable block*.

Lemma 4.5.8. *If two blocks are connected by extra edges, then at most one of them can be a stable block in a Pareto-optimal matching M_0.*

Proof. Assume by contradiction there is a Pareto-optimal matching M that contains two stable blocks i and j that are connected. Since w_j^2 and w_i^2 are the top choices for m_i^1 and m_j^1 correspondingly, then there is a coalition cycle $m_i^2; w_j^2; m_j^1; w_i^2$. This contradicts to Pareto-optimality of M. □

Let S be an independent set. Let I_S be the set of blocks that corresponds to S, and $I_{V \setminus S}$ be the set of blocks that corresponds to $V \setminus S$. Assign $(m_i^1; w_i^2); (m_i^2; w_i^0) \in M_1$ for each block from I_S, and $(m_i^0; w_i^2); (m_i^2; w_i^2); (m_i^2; w_i^0) \in M_1$ for each block from $I_{V \setminus S}$. Notice that M_1 is a Pareto-optimal matching.

Lemma 4.5.9. *M_1 achieves $\Delta(M_0) = 3|V| - |S|$ if and only if S is the largest independent set.*

Proof. Assume that S is the largest independent set, but there is a Pareto-optimal matching M_2 such that $\Delta(M_0; M_2) < \Delta(M_0; M_1) = 3|V| - |S|$. This would imply that M_2 has more stable blocks and this would correspond to a larger stable set S', contradiction.

Assume M_1 achieves $\Delta(M_0) = 3|V| - |S|$, but S' is the largest independent set such that $|S'| > |S|$, then S' corresponds to a Pareto-optimal matching M_2 that has more stable blocks than M_1 and would achieve smaller $\Delta(M_0)$ correspondingly. □

4.6 Proofs of Theorem 4.1.5 and Theorem 4.1.6

In this section, we prove Theorem 4.1.6, which in particular implies Theorem 4.1.5. For the reader's convenience, the theorem is restated below.

Theorem 4.1.6. *There exists a universal constant c such that the following holds. For each $\varepsilon > 0$, unless P=NP, there is no polynomial-time algorithm that, given a preference system $(G(A \cup B; E); \prec)$, accepts if $\Gamma = 0$, rejects if $\Gamma \geq c|A|$, and is free to accept or reject if none of these cases apply.*

Because of Lemma 4.2.5, it suffices to show the statement with Δ replaced by Γ.

For a fixed ε, the instance of the marriage problem is constructed from gadgets corresponding to literals and some edges that connect them. The following is the description of a gadget that corresponds to one literal. Positive and negative literals

have the same associated gadgets H, but the true and false assignments correspond to different popular matchings in those gadgets. Nodes of the gadget that correspond to literal ℓ_i are labeled as $a(\ell_i)$, $b(\ell_i); c(\ell_i)$ and $d(\ell_i)$. Sometime we omit the dependency on ℓ_i if it is clear to which gadget the nodes belong.

One gadget consists of two men $b; d$ and two women $a; c$. Fig. 4.9 and Fig. 4.10 provide an example of the construction.

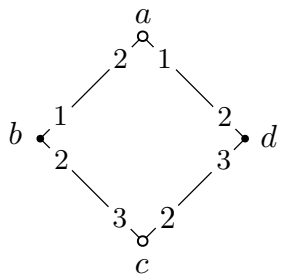

Figure 4.9: One gadget.

If ℓ_i and ℓ_j are literals that correspond to a variable and its negation respectively, then the corresponding gadgets are connected by extra edges $c(\ell_i)b(\ell_j)$ (consistency edges). For woman $c(\ell_i)$, add man $b(\ell_j)$ to the beginning of her preference list, and for a man $b(\ell_j)$, add woman $c(\ell_i)$ to the end of his preference list. If c only has two adjacent edges, then d is the top choice, and b is the second (last) choice in the preference list of c.

For a clause with exactly two literals $\ell_1; \ell_2$, add connectivity edges $d(\ell_1)a(\ell_2)$ and $d(\ell_2)a(\ell_1)$. For men $d(\ell_1); d(\ell_2)$ add women $a(\ell_1); a(\ell_2)$ to the top of their preferences list respectively. For women $a(\ell_1); a(\ell_2)$ add men $d(\ell_1); d(\ell_2)$ to the end of their preferences list respectively. If the clause has three positive literals $\ell_1; \ell_2; \ell_3$, add edges $d(\ell_1)a(\ell_2)$, $d(\ell_2)a(\ell_3)$ and $d(\ell_3)a(\ell_1)$. The preference lists are modified similarly to the case with two literals in a clause. We define graph $G(\ )$ as follows. First, construct disjoint graphs $G(\ell)$ for each literal ℓ, then add edges that connect literals in the same clause, and edges that connect literals that correspond to a variable and its negation,

as described above.

Lemma 4.6.1. *$G(\ )$ is bipartite.*

Proof. We assign each vertex from $V := V(G(\ ))$ to exactly one of two sets A or B, and prove that if $u; v \in A$ (or $u; v \in B$) then $uv \notin E := E(G(\ ))$. The proof can be followed in Fig. 4.10, where nodes are colored according to the set of the bipartition they belong to. For each literal ℓ assign $b; d$ to A and $a; c$ to B. One easily checks that edges in all gadgets connect nodes in different sets of the bipartition. Now consider $b(\ell)c(\ell')$, where by construction ℓ (resp. ℓ') is a negative (resp. positive) literal. Hence $b(\ell) \in A$, $c(\ell') \in B$, as required. □

To a consistent assignment $\ $, we associate a matching $M = M(\ )$ defined as follows: $T(H(\ell)) \subseteq M$ if literal ℓ is set to true, $F(H(\ell)) \subseteq M$ otherwise, where (dependency on ℓ is omitted):

$$T(H) := \{ab; cd\},$$
$$F(H) := \{ad; bc\}.$$

Claim 4.6.1. *An assignment to $\ $ is consistent if and only if it corresponds to a popular matching.*

Proof. Assume we are given a consistent assignment that corresponds to a matching M. By the construction all nodes are matched and in any gadget for a literal ℓ either $a(\ell)b(\ell); c(\ell)d(\ell) \in M$ or $a(\ell)d(\ell); b(\ell)c(\ell) \in M$. This implies that $d(\ell)a(\ell') = (+; \ )$ for all $\ell; \ell'$. Similarly, $c(\ell)a(\ell') = (+; \ )$ for all $\ell; \ell'$.

There is no M-alternating path from an unmatched vertex to a $(+;+)$ edge, since there are no unmatched vertices in M. Note that the only $(+;+)$ edges are all and only those of the form $a(\ell)d(\ell)$ for all ℓ such that ℓ is set to true. Consider any maximal M-alternating path P which contains a certain $(+;+)$ edge $a(\ell)d(\ell)$. P must also contain the subpath $b(\ell)a(\ell)d(\ell)c(\ell)$.

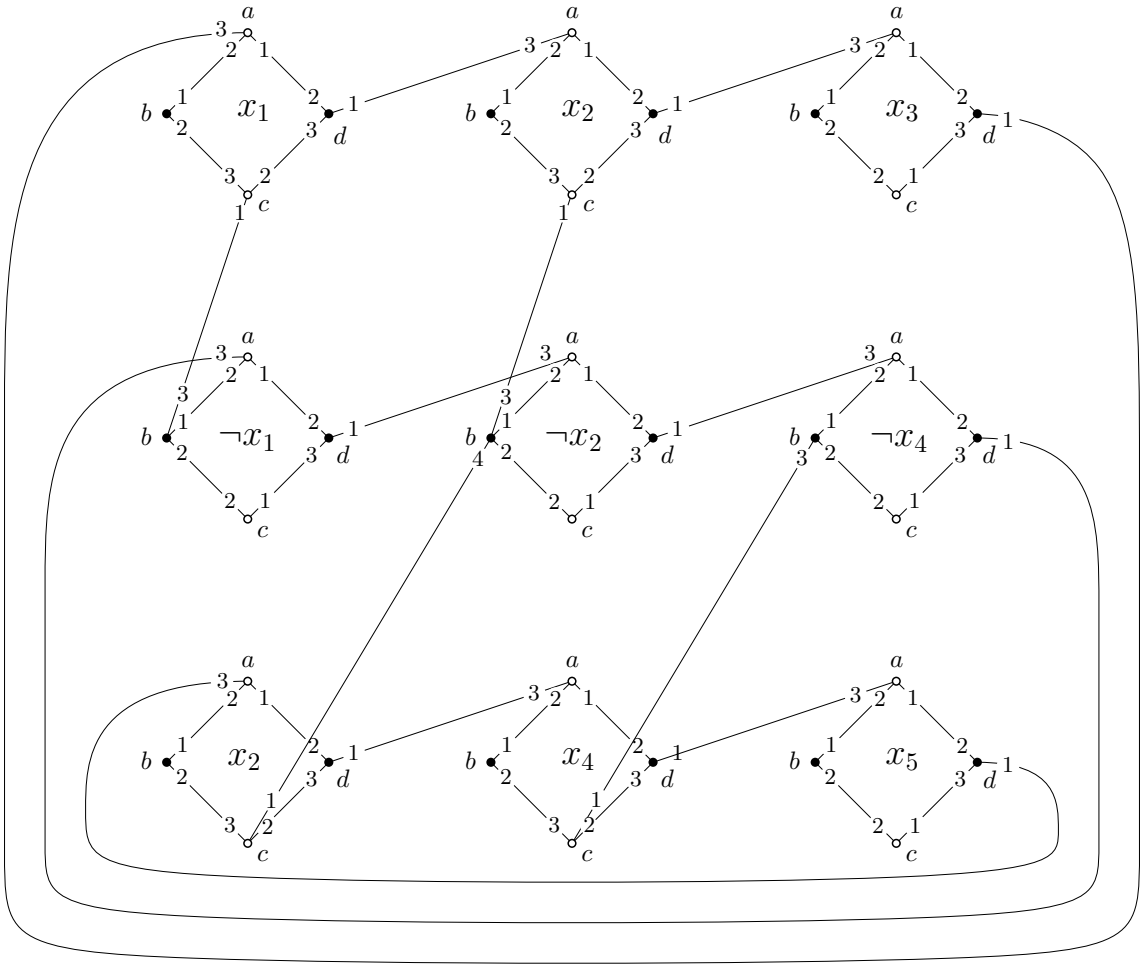

Figure 4.10: A monotone 3-SAT instance $= (x_1 \_ x_2 \_ x_3) \wedge (:x_1 \_ :x_2 \_ :x_4) \wedge (x_2 \_ x_4 \_ x_5)$ with the corresponding preference system $(G(\ ); <)$.

Suppose first that $`$ is a positive literal. For positive literals, $b(`)$ has degree one in G_M, implying that there is no M-alternating cycle in G_M containing $a(`)d(`)$. The only way to continue the M-alternating path P is by extending it through a consistency edge incident to $c(`)$. Suppose therefore that $P$ contains the consistency edge $c(`)b(`_1)$, and that $`_1$ is in the clause formed by literals $`_1; `_2; `_3$ in this order. Since $`$ is set to true and M is consistent, $`_1$ is set to false. This implies that $P$ has as subpath $b(`)a(`)d(`)c(`)b(`_1)c(`_1)d(`_1)a(`_1)d(`_3)$. Now we have two possibilities:

1. $`_3$ is set to true;
2. $`_3$ is set to false, $`_2$ is set to true;

3. ℓ_3 is set to false, ℓ_2 is set to false;

In the first case, P continues along $d(\ell_3)c(\ell_3)$ where we cannot continue since $c(\ell_3)$ has degree one in G_M. In the second case there are two possibilities, either P continues along $d(\ell_3)a(\ell_3)b(\ell_3)c(\ell_3)$, or along $d(\ell_3)a(\ell_3)d(\ell_2)c(\ell_2)$. Similar to the first case, the path stops afterwards, without traversing a second $(+,+)$ edge. In the third case there is one additional possibility $d(\ell_3)a(\ell_3)d(\ell_2)a(\ell_2)b(\ell_2)c(\ell_2)$, since $d(\ell_1) \not\in P$ and $c(\ell_2)$ has degree one in G_M.

Now suppose that ℓ is a negative literal. Here, $c(\ell)$ has degree one in G_M which implies that there is no M-alternating cycle in G_M containing $a(\ell)d(\ell)$. Similar to above, the only way to continue the path P is by extending it through a consistency edge leaving at $b(\ell)$. We may suppose that P contains the consistency edge to the gadget $H(\ell_1)$ and that ℓ_1 is in the clause containing literals ℓ_1, ℓ_2, ℓ_3 in this order. Since ℓ is a negative literal which is set to true, we know that ℓ_1, ℓ_2, ℓ_3 are positive literals and ℓ_1 is set to false, since M is consistent. This implies that P has as subpath $c(\ell)d(\ell)a(\ell)b(\ell)c(\ell_1)b(\ell_1)a(\ell_1)d(\ell_1)a(\ell_2)$. Again we have two possibilities:

1. ℓ_2 is set to true;
2. ℓ_2 is set to false.

In the first case, P continues along $a(\ell_2)b(\ell_2)$ where we cannot continue since $b(\ell_2)$ has degree one in G_M. In the second case there are two possibilities, either P continues along $a(\ell_2)d(\ell_2)c(\ell_2)b(\ell_2)$, $a(\ell_2)d(\ell_2)a(\ell_3)b(\ell_3)$ or $a(\ell_2)d(\ell_2)a(\ell_3)d(\ell_3)c(\ell_3)b(\ell_3)$ where the path stops since $a(\ell_1) \not\in P$.
So there is no M-alternating path containing two $(+,+)$ edges and no M-alternating cycle containing a $(+,+)$ edge in G_M. We conclude that M is popular.

Assume we are given a popular matching M. We will show that the corresponding assignment is consistent. Assume that the assignment is not consistent, meaning there are two literals ℓ and ℓ' in a positive and negative gadgets correspondingly that are

both assigned to true. In this case there is a path $a(\grave{})d(\grave{})c(\grave{})b(\grave{}^{0})a(\grave{}^{0})d(\grave{}^{0})$ with two $a(\grave{})d(\grave{}); a(\grave{}^{0})d(\grave{}^{0}) = (+;+)$. This contradicts popularity. □

Claim 4.6.2. *All feasible assignments to correspond to Pareto-optimal matchings.*

Proof. Assume we are given a feasible assignment that corresponds to a matching M. We need to show that there is no coalition cycle in M to get Pareto-optimality. We can easily check that there is no coalition cycle inside of a gadget. Since b prefer its possible partners in their own gadget to the one at the other end of any consistency edge, the consistency edge is never used in a coalition cycle of M. Assume there is a coalition cycle. It does not contain consistency edges, hence it is contained in the union of gadgets of literals that belong to exactly one clause, and since there are no coalition cycles inside a gadget, it must use all connectivity edges in that clause. But if $\grave{}$ is set to true, it is not possible to obtain an M-alternating path in G_M containing both connectivity edges incident to the $H(\grave{})$, since $b(\grave{})c(\grave{}) = (\ ;\ )$. Since there is a true literal in every clause, we conclude that there are no coalition cycles in G_M, the thesis follows. □

Claim 4.6.3. *Let M be a popular matching that corresponds to a consistent assignment to with k unsatisfied clauses. Then* $\min_{M^{0}\ \text{Pareto opt}} jM \quad M^{0}j = 4k$.

Proof. Let M^{0} be any Pareto-optimal matching. Let c be a clause that is not satisfied by , and consider the matching M_c (resp. M_c^{0}) obtained by restricting matching M (resp. M^{0}) to the subgraph of G corresponding to gadgets of c. Let $\grave{}_1; \grave{}_2$ (and possibly $\grave{}_3$) be the literals from c. Next we prove that $jM_c \quad M_c^{0}j \quad 4$.

Assume all literals in the clause C correspond to all negated variables. Similar proof holds for the case when all literals correspond to positive variables. Notice, that if $a(\grave{}_i)d(\grave{}_i) \ \ 2 \ M_c$ for all $i = 1;2;3$, then there is a coalition cycle $a(\grave{}_1)d(\grave{}_1)a(\grave{}_2)d(\grave{}_2)a(\grave{}_3)d(\grave{}_3)$. We conclude that there is at least one edge

$a(\grave{}_i)d(\grave{}_i) \in M_c^{\emptyset}$ for some i. Assume without loss of generality that $i = 1$. Next, we will consider all possible Pareto-optimal matchings by analyzing the following cases: $c(\grave{}_1)d(\grave{}_1); d(\grave{}_1)a(\grave{}_2) \in M_c^{\emptyset}$, $c(\grave{}_1)d(\grave{}_1) \notin M_c^{\emptyset}$ and $d(\grave{}_1)a(\grave{}_2) \notin M_c^{\emptyset}$.

If $c(\grave{}_1)d(\grave{}_1); d(\grave{}_1)a(\grave{}_2) \in M_c^{\emptyset}$, then $b(\grave{}_1)c(\grave{}_1) \notin M_c^{\emptyset}$, otherwise node $d(\grave{}_1)$ prefers unmatched node $c(\grave{}_1)$, which is a contradiction to Pareto-optimality. Similarly, we argue that $a(\grave{}_1)d(\grave{}_3) \notin M_c^{\emptyset}$, and either $a(\grave{}_3)b(\grave{}_3) \notin M_c^{\emptyset}$ or $a(\grave{}_3)d(\grave{}_2) \notin M_c^{\emptyset}$. In both cases $|M_c \oplus M_c^{\emptyset}| > 4$, since $a(\grave{}_1)d(\grave{}_1); a(\grave{}_3)d(\grave{}_3)$, $b(\grave{}_3)c(\grave{}_3) \in M_c^{\emptyset}$ and $a(\grave{}_1)d(\grave{}_3)$, $a(\grave{}_3)b(\grave{}_3) \notin M_c^{\emptyset}$.

If $c(\grave{}_1)d(\grave{}_1) \notin M_c^{\emptyset}$, then either $b(\grave{}_1)a(\grave{}_1) \notin M_c^{\emptyset}$ or $a(\grave{}_1)d(\grave{}_3) \notin M_c^{\emptyset}$, since otherwise node $b(\grave{}_1)$ prefers unmatched node $a(\grave{}_1)$, which is a contradiction to Pareto-optimality. In the first case $|M_c \oplus M_c^{\emptyset}| \geq 4$, since $a(\grave{}_1)d(\grave{}_1); b(\grave{}_1)c(\grave{}_1) \in M_c^{\emptyset}$ and $a(\grave{}_1)b(\grave{}_1); c(\grave{}_1)d(\grave{}_1) \notin M_c^{\emptyset}$. In the second case $|M_c \oplus M_c^{\emptyset}| > 4$, since $a(\grave{}_1)d(\grave{}_1); b(\grave{}_1)c(\grave{}_1)$, $a(\grave{}_3)d(\grave{}_3) \in M_c^{\emptyset}$ and $c(\grave{}_1)d(\grave{}_1); a(\grave{}_1)d(\grave{}_3) \notin M_c^{\emptyset}$.

If $d(\grave{}_1)a(\grave{}_2) \notin M_c^{\emptyset}$, then, similarly to the argument above, either $a(\grave{}_1)b(\grave{}_1) \notin M_c^{\emptyset}$ or $b(\grave{}_1)c(\grave{}_1)$, $a(\grave{}_1)d(\grave{}_3) \notin M_c^{\emptyset}$. $|M_c \oplus M_c^{\emptyset}| \geq 4$, since $a(\grave{}_1)d(\grave{}_1); b(\grave{}_1)c(\grave{}_1)$, $a(\grave{}_2)d(\grave{}_2) \in M_c^{\emptyset}$ and $a(\grave{}_1)b(\grave{}_1)$, $d(\grave{}_1)a(\grave{}_2) \notin M_c^{\emptyset}$, or $a(\grave{}_1)d(\grave{}_1)$, $a(\grave{}_3)d(\grave{}_3) \in M_c^{\emptyset}$ and $d(\grave{}_1)a(\grave{}_2)$, $a(\grave{}_1)d(\grave{}_3) \notin M_c^{\emptyset}$

In all cases the arguments hold even if there are only two literals per clause.

This implies that for all unsatisfied clauses there must be at least four edges changed from either being matched to unmatched or vice versa, in order to obtain a Pareto-optimal matching. Notice that $\Delta(M_c; M_c^{\emptyset}) = 4$, when the corresponding assignment of the literal changes parity, so the corresponding clause becomes satisfied. If for each unsatisfied clause of we change parity for exactly one literal, the obtained assignment is feasible, since all clauses are satisfied (but not necessarily consistent). According to Claim 4.6.2, this feasible assignment corresponds to a Pareto-optimal matching $M^{\emptyset}$, and $|M \oplus M^{\emptyset}| = 4k$. □

Claim 4.6.4. *Let $M; M'$ achieve the minimum in $\min_{M \text{ popular},\ M' \text{ Pareto opt}} jM \quad M'j$. Then every consistent assignment has at least $k = \frac{\min_{M' \text{ Pareto opt}} jM \quad M'j}{4}$ unsatis ed clauses.*

Proof. By contradiction. Assume that there is a consistent assignment such that the number of unsatisfied clauses is less than k. By Claim 4.6.1, this assignment corresponds to a popular matching M . Change the parity for exactly one literal in each unsatisfied clause as to obtain a feasible (possibly not consistent) assignment, and let M' be the corresponding matching. According to Claim 4.6.2 M' is Pareto-optimal. Then $\Delta(M ; M') < 4 \ k = \min_{M' \text{ Pareto opt}} jM \quad M'j = \min_{M \text{ popular},\ M' \text{ Pareto opt}} jM$ $M'j$, a contradiction. □

Take $c = \frac{1}{12}$, and suppose that an algorithm A'' as in the hypothesis of the theorem exists. We show that we can provide an algorithm A' as in the hypothesis of Lemma 1.4.2, a contradiction (assuming P$\neq$NP). Feed $(G(A \cup B; E); <)$ to A''. If A'' accepts, then $\min_{M \text{ popular},\ M' \text{ Pareto opt}} jM \quad M'j < (c \quad)jAj$. By Claim 4.6.3, there is a consistent assignment that satisfies

$$m \quad \frac{\min_{M \text{ popular},\ M' \text{ Pareto opt}} jM \quad M'j}{4} \quad m \quad \frac{c}{4} jAj \quad (1 \quad \frac{3}{2}c + \frac{3}{2})m > (1 \quad \frac{3}{2}c +)m =$$

$$= (\frac{7}{8} +)m$$

clauses. We can then make A' accept. Suppose conversely that A'' rejects. Then $\min_{M \text{ popular},\ M' \text{ Pareto opt}} jM \quad M'j > 0$, hence is not satisfiable (by Claim 4.6.4), and we can make A' reject.

Chapter 5

Combinatorial Scheduling for Adaptive Machine Learning in Cybersecurity

5.1 Introduction

Cyber-security is an important problem in today's world. Enormous amounts of computations across different industries rely on secure operation of networked computing devices. These systems contain information that can be used for malicious intentions such as identity theft, damage of reputation and business operations, loss of valuable resources, loss of intellectual property and even compromise national security. With the advances of cryptography algorithms the cyber-attacks become more elaborate as well. These attacks exploit the smallest flaws in the system and are guided by intelligent and adaptive adversaries with resources. They can last for long periods of time while remaining undetected and progress slowly by learning the system and its design.

Corporations and governments spend tremendous amounts of resources to avoid data breaches and system infiltration to protect their information, as it's compromise can lead to serious and costly consequences. The development of modern cryptography algorithms reflects the need of design that incorporates the possibility of data being partially compromised. Moving target defense (MTD) strategies are one of the techniques that attempt to mitigate complicated attacks on a system.

In [51] authors introduce game *FlipIt* to model attacks that compromise a system or take control of a resource completely, but are not necessarily revealed immediately.

In this model there are two players, the attacker and the defender. They are both interested in gaining control of a single resource that in real-life could be a password, design of a system, cryptographic key etc. Players can take a move at any given time that allows them to immediately gain control of the resource. Attacker and defender are not aware of their opponents moves and who controls the resource except at the time when they make a move. Each move has a cost. Players are interested in maximizing their utility which is the amount of time they controlled the resource minus the total costs associated with the moves. This motivates the game's second name, *game of stealthy takeover*. There are multiple papers ([39], [38], [44]) that explore FlipIt game and its extensions.

In Fig. 5.1 it is shown a schematic diagram of an instance of FlipIt game.

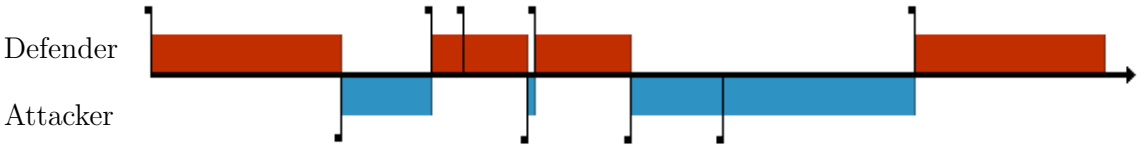

Figure 5.1: An example of FlipIt game. Defender controls the resource in the beginning, then the attacker makes a move and the defender looses the control. Flags indicate the moves of the players. Red and blue colors show time when the defender and attacker control the resource respectively.

Moving target defense (MTD) strategies allow to mitigate sophisticated attacks by incorporating randomness or making a static part of a system dynamic. For example, certain parts of the system might be required to restart to a known, not compromised state, such as change of a password. The additional cost required to obtain needed information from such systems might prevent the adversary to start the attack. In [42] authors provide survey on MTD strategies. In [30] authors develop a strategic game, PLADD, which is an abstract model for MTD. PLADD stands for Probabilistic Learning Attacker, Dynamic Defender. This model incorporates key characteristics that are shared among different categories of MTD strategies [30]:

1. There exists some information that provides competitive advantage to the attacker when its in his/her possession.
2. The defender can take this information from the attacker at least temporary.

The inspiration for PLADD comes from FlipIt game, but there are several important differences [30]:

1. The attacker does not gain the control of the resource immediately after the attack. There is a random time-to-success wait period.
2. Time-to-success incorporates the ability of the attacker to learn. Its distribution changes over time.
3. The defender can make two types of moves that are accomplished without delays. The first move "take" allows to gain the control of the resource. The second move "morph" allows to gain the control of the resource and take information away from the attacker (stop the ongoing attack). This allows the attacker to detect both defender's moves when the attacker controls the resource (the attacker knows when they lose control) and to detect a morph move even if the defender controls the resource, since morph move restarts the system. After the morph the first attacker move comes from f_{base} distribution, and the following attacks are drawn from f_{learned} distribution untill the next morph move. A take move does not restart the system, so it does not reset the distribution to f_{base}. At any time only one attack can be in progress. Fig. 5.2 shows an instance of PLADD game.

The attacker's and defender's goal is to maximize their utility which is total time in control minus total cost associated with moves. The defender has two costs c_{take} and c_{morph} associated with the corresponding moves. The attacker has a fixed cost to start an attack and a variable cost during the attack. The variable cost is proportional the duration of the attack.

The analysis of PLADD consists of deriving strategies of when and what type of moves the defender should make.

In [30] authors analyzed PLADD, derived closed-form solutions in multiple game variants and developed simulation of the game that verified their analytical solution. They also formulated a stochastic programming model and proposed another approach for analyzing PLADD, a new combinatorial scheduling problem. In this work we analyze the proposed latter approach.

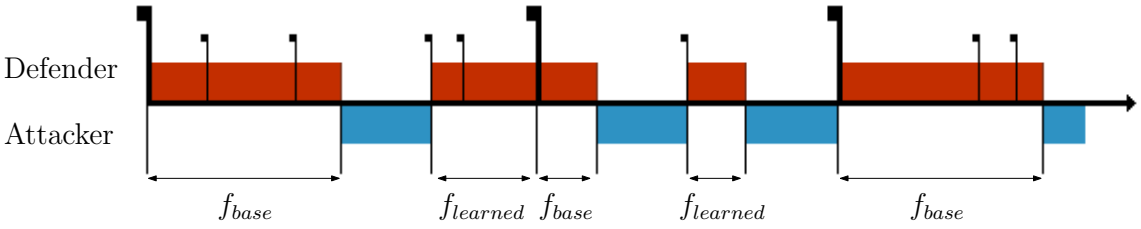

Figure 5.2: This diagram illustrates an example of PLADD game. Small flags indicate take moves and large flags indicate morph moves. The defender starts the game with a morph move. At the same time the attacker starts the attack that does not succeed immediately but have time-to-success wait time drawn from f_{base} distribution. During this wait time the defender makes two take moves that are wasted and do not have any influence on the system. After that the attacker gains the control of the resource until the defender places a take and regains the control of the system. The attacker knows that he lost control and immediately starts a new attack. This time the wait-to-success time is drawn from $f_{learned}$ distribution, since the attacker already acquired some knowledge about the system from the previous attach. The defender places a take that does not influence the game. The next defender's move, morph, cancels the attack and reset the distribution of wait-to-success time to base. The game resets. The attacker lost acquired knowledge and starts a new attack from f_{base} distribution.

5.2 The PLADD game and a new scheduling problem

In this section we present a new scheduling problem for PLADD analysis [30]. Since a morhp move completely resets the system by setting the time-to-success distribution to f_{base} (all prior attacker's knowledge about the system is deleted), we will analyse

the game between two successive morph moves with fixed number of takes. Let T be the game's time horizon that equals to the time between two morph moves. Assume there are m scenarios, and each scenario is a list of t_{k+1} draws such that the first draw is from f_{base} distribution, and the consecutive k draws are from $f_{learned}$ distribution. The goal is to schedule k takes withing T time such that the attacker control time is minimized over all scenarios. This approach allows to find a strategy for a defender that minimizes on average the expected attacker control time.

Each scenario can be modelled as a machine or a server. Each such machine has a list of jobs with fixed length that must be processed in order. At time $t_0 = 0$ all machines start their first jobs. If there is a start time (a take in a PLADD scenario), a machine can only start the next job if it finished its current job (the machine was idle), otherwise the machine continues processing its current job. The problem is to find a schedule of start times, or synchronization points, $t_i; i = 1;:::;k$, that minimizes total idle time on all machines, where $k + 1$ is the length of the longest list on a machine. Without loss of generality, we assume that the total processing time of all jobs on any machine is at most T. In order to achieve this we can truncate the last job or drop jobs from the end of the list. This implies that on any machine either the total processing time of all jobs equals to T, or number of jobs is $k + 1$, or both. Fig. 5.3 provides an example of a scheduling problem with three machines.

If there is one machine, then the optimal solution is a schedule where each job starts immediately the previous job finishes, or all start times are placed at the end of jobs. For multiple machines there is an optimal solution such that all synchronization points are scheduled at the end of some jobs that do not necessary belong to one machine, [30]. This statement follow from the observation that if there is a schedule such that all machines idle t time units before a start time, then scheduling this synchronization point and all following it t time units earlier might allow to schedule more job at the end, or provide additional processing time for jobs that were not

finished.

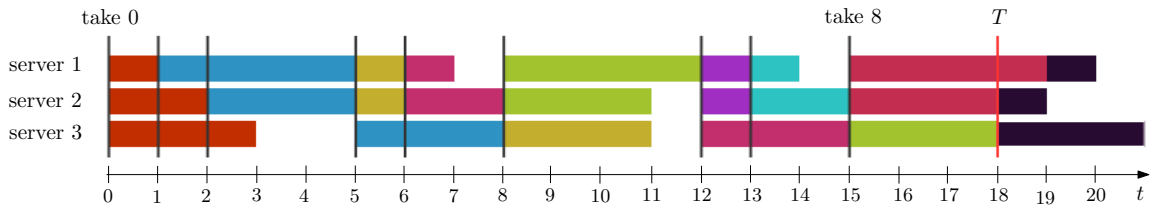

Figure 5.3: This diagram illustrates an example of PLADD game. There are three servers. Different jobs on one machine are colored differently for visualization purpose. On all servers jth jobs are the same color. For example, first and second jobs on all servers have red and blue colors respectively. All job lengths are measured in units of time that is needed to process them. The first server has 9 jobs of lengths 1, 4, 1, 1, 4, 1, 1, 4, 1. The second server has 9 jobs of lengths 2, 3, 1, 2, 3, 1, 2, 3, 1. The third server has 6 jobs, each of length 3. There are 9 takes (vertical black lines) and $T = 18$ (vertical red line). The first take t_0 is scheduled at time $t = 0$. The rest of the takes are scheduled at times 1, 2, 5, 6, 8, 12, 13, 15. The idle time is represented by white color between jobs on one server. Jobs after time T shown in dark color to illustrate that they were not processed. The diagram shows the optimal schedule for this instance with the total idle time of 6 units.

5.3 Integer Programming Formulation

In this section we present an integer programming formulation (IP) for the problem. The time scale is discretized as the set $f0;:::;Tg$. Any job that does not start in $f0;:::;T \quad 1g$ is permitted to start at time T without additional take and without satisfying precedence constraints, since we are only concerned about the idle time before time T and are not interested about the schedule after this time horizon. All n_m jobs for each machine $m \ 2 \ f1;:::;Mg$ are indexed in order they must be processed. The length of job j on machine m is $`_{m;j}$ for $j \ 2 \ f1;:::;n_m g$ and $m \ 2 \ f1;:::;Mg$. Let $L(m;i;j) = \sum_{k2[i;j]} `_{m;k}$ be the total sum of lengths of all jobs with indices $i;:::;j$ on machine m. $L_m = L(m;1;n_m)$ is the total length of all the jobs on machine m. Let $\Delta_{j;m}(t) = fs \ j \ t \ 2 \ [s;s+`_{m;j})g$ be a set of start times s for job j on a machine m such that if j starts at time s, then it still runs at time t. The earliest possible start

time for job j on machine m is $ET(m;j) := L(m;1;j \quad 1)$. The binary variables for the formulation are the following:

$x_{m;j;t} = 1$ indicates that job j starts at time t on the machine m and 0 otherwise,

$y_t = 1$ indicates that jobs are allowed to start at time t, i.e. a take places at time t,

$z_{m;t} = 1$ indicates that the machine m is idle at time t.

The variables $x_{m;j;t}$ are defined for $j \ 2 \ [1;n_m]$, $m \ 2 \ [1;M]$ and $t \ 2 \ [ET(m;j);T]$. The variables y_t and $z_{m;t}$ are defined for $t \ 2 \ [0;T)$. We are given k takes (not including the first take at start time $t = 0$). The objective is to minimize idle time:

$$\min \sum_{m2[1;M]} \sum_{t2[0;T)} z_{m;t}:$$

We require that every job is scheduled:

$$\sum_{t2[ET(m;j);T]} x_{m;j;t} = 1; \ 8m;j: \tag{5.1}$$

The first jobs on all machines start at time $t = 0$:

$$x_{m;1;0} = 1; \ 8m: \tag{5.2}$$

We also require that at most one job is in process at any time $[0;T)$. Recall that at time T multiple jobs are allowed to be scheduled, and time T is not included in measuring idle time. If all $x_{m;j;s}$ in the summation below are zeros, then $z_{m;t}$ must be 1, indicating that the server is idle.

$$z_{m;t} + \sum_{j} \sum_{s2 \ m;j(t)} x_{m;j;s} = 1; \ 8t < T \text{ and } 8m: \tag{5.3}$$

We impose precedence constraints to ensure that for a job i, all jobs with a greater index j must start after all jobs between it and i have completed. For an IP formulation the precedence constraints only for $j = i + 1$ are needed. However, in Section 5.4 we provide an example that illustrates that including the constraints for $j > i + 1$ result in a stronger LP relaxation.

$$x_{m;j;t} \quad \sum_{s2[t+l_{m;j};T]} x_{m;j+1;s}; \ 8m; j; t : ET(m; j) \quad t < T: \tag{5.4}$$

Precedence constraints (5.4) with constraints for $j > i + 1$:

$$x_{m;i;t} \quad \sum_{s2[t+L(m;i;j \quad 1);T]} x_{m;j;s}; \ 8m; i; j; t : ET(m; j) \quad t < T; j > i: \tag{5.5}$$

The following constraints are complementary to (5.4) and (5.5) respectively. They are not necessary for an IP formulation. They ensure that for a job i, each job with a smaller index j must end before i starts and leave enough time for processing jobs between j and i. Similarly to above, for an IP formulation the precedence constraints only for for $j = i \quad 1$ are needed.

$$x_{m;j+1;t} \quad \sum_{s2[ET(m;j);t \quad l(m;j)]]} x_{m;j;s}; \ 8m; j; t : ET(m; j) \quad t < T: \tag{5.6}$$

Precedence constraints (5.6) with constraints for $j < i + 1$:

$$x_{m;i;t} \quad \sum_{s2[ET(m;j);t \quad L(m;j;i \quad 1)]} x_{m;j;s}; \ 8m; i; j; t : ET(m; j) \quad t < T; j < i: \tag{5.7}$$

Constraints (5.4) - (5.7) can be replaced by the following constraints (5.8) - (5.11). We replace the left-hand side with an appropriate sum. This results in a stronger LP relaxation as shown in Section 5.4.

$$\sum_{r2[t;T]} x_{m;i;r} \quad \sum_{s2[t+l_{m;j};T]} x_{m;j+1;s};\ 8m; j; t : ET(m; j) \quad t < T: \tag{5.8}$$

$$\sum_{r2[t;T]} x_{m;i;r} \quad \sum_{s2[t+L(m;i;j \quad 1);T]} x_{m;j;t};\ 8m; i; j; r : ET(m; j) \quad t < T; j > i: \tag{5.9}$$

$$\sum_{r2[ET(j+1);s]} x_{m;j+1;r} \quad \sum_{s2[ET(m;j);t \quad l(m;j)]]} x_{m;j;s};\ 8m; j; t : ET(m; j) \quad t < T: \tag{5.10}$$

$$\sum_{r2[ET(i);s]} x_{m;i;r} \quad \sum_{s2[ET(m;j);t \quad L(m;j;i \quad 1)]} x_{m;j;s};\ 8m; i; j; t : ET(m; j) \quad t < T; j < i: \tag{5.11}$$

We ensure that jobs may only start if a take is placed. Number of takes is budgeted:

$$\sum_{j} x_{m;j;t} \quad y_t;\ 8m; t : ET(m; j) \quad t < T; \tag{5.12}$$

$$\sum_{t2[0;T)} y_t \quad k + 1: \tag{5.13}$$

At time $t = 0$ all machines start their first jobs. This implies the first take is always placed at $t = 0$.

$$y_0 = 1: \tag{5.14}$$

5.4 Linear Programming Relaxation

In this section we explain the meaning of variables in linear programming relaxation (LP) and give some examples that illustrate how redundant constraints for IP

strengthen LP. The solutions to IP and LP were found using Python implementation with Gurobi Optimizer. For each example we provide input data and the output for variables in IP or LP formulations. The input data consists of list *length*, time horizon T and number of takes k (without the take at time $t = 0$). List *length* contains lists that correspond to servers and contain lengths of jobs in order. The output data contains variables z, x and y. We will slightly change notation of variables to resemble that of Python. Variables $z[m][t]$ are presented in a table where the rows correspond to servers and columns correspond to time line $t = [0;:::;T)$. Variable $x[m][j][t]$ are presented for each server separately in a table, where each row corresponds to a job in order $0;:::;n_m \quad 1$ and columns correspond to time line $t = [0;:::;T)$. Variables $y[t]$ are given in a vector, each entry correspond to a value of y at time $t = [0;:::;T)$. All number are rounded to two decimals.

Example 5.4.1. *The IP formulation includes constraints* (5.1) - (5.3), (5.9), (5.11), (5.12) - (5.14).

Input:

```
length = [[1,4,1,1,4],[2,3,2,2,2],[3,3,3,2]]
T = 11
K = 4
```

Output for IP

```
z[m][t]
[0][1][1][0][0][0][0][1][1][0][0]
[0][0][1][0][0][0][0][0][1][0][0]
[0][0][0][0][0][0][0][0][0][0][0]

x[1][j][t]
[1][0][0][0][0][0][0][0][0][0][0]
```

[0][0][0][1][0][0][0][0][0][0][0]

[0][0][0][0][0][0][0][0][0][1][0]

[0][0][0][0][0][0][0][0][0][0][1]

[0][0][0][0][0][0][0][0][0][0][0]

x[2][j][t]

[1][0][0][0][0][0][0][0][0][0][0]

[0][0][0][1][0][0][0][0][0][0][0]

[0][0][0][0][0][0][1][0][0][0][0]

[0][0][0][0][0][0][0][0][0][1][0]

[0][0][0][0][0][0][0][0][0][0][0]

x[3][j][t]

[1][0][0][0][0][0][0][0][0][0][0]

[0][0][0][1][0][0][0][0][0][0][0]

[0][0][0][0][0][0][1][0][0][0][0]

[0][0][0][0][0][0][0][0][0][1][0]

y[t]

[1][0][0][1][0][0][1][0][0][1][1]

Output for LP

z[m][t]

[0.00][0.67][0.33][0.00][0.00][0.00][0.00][0.67][0.33][0.00][0.00]

[0.00][0.00][0.67][0.00][0.00][0.00][0.00][0.00][0.33][0.00][0.00]

[0.00][0.00][0.00][0.33][0.33][0.00][0.00][0.00][0.00][0.00][0.00]

x[1][j][t]

```
[1.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00]
[0.00][0.33][0.33][0.33][0.00][0.00][0.00][0.00][0.00][0.00][0.00]
[0.00][0.00][0.00][0.00][0.00][0.33][0.33][0.00][0.00][0.33][0.00]
[0.00][0.00][0.00][0.00][0.00][0.00][0.33][0.00][0.33][0.00][0.33]
[0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.33][0.00][0.33][0.00]

x[2][j][t]
[1.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00]
[0.00][0.00][0.33][0.67][0.00][0.00][0.00][0.00][0.00][0.00][0.00]
[0.00][0.00][0.00][0.00][0.00][0.33][0.67][0.00][0.00][0.00][0.00]
[0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.33][0.33][0.33][0.00]
[0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.33][0.33]

x[3][j][t]
[1.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00]
[0.00][0.00][0.00][0.67][0.00][0.33][0.00][0.00][0.00][0.00][0.00]
[0.00][0.00][0.00][0.00][0.00][0.00][0.67][0.00][0.33][0.00][0.00]
[0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.00][0.67][0.00]

y[t]
[1.00][0.33][0.33][0.67][0.0][0.33][0.67][0.33][0.33][0.67][0.33]
```

Fig. 5.4 and Fig. 5.6 illustrate the optimal solutions for IP and LP respectively. Fig. 5.5 also give another optimal solution for IP.

The total idle time for IP is 6 time units, for LP the total idle time is $3\frac{2}{3}$ *time units. Notice that in LP takes can have fractional values. A fractional take of value p at time t means that p fraction of a new job can start at time t if the server has capacity. For example, in Fig. 5.6, at time* $t = 0$ *all servers started rst jobs, take*

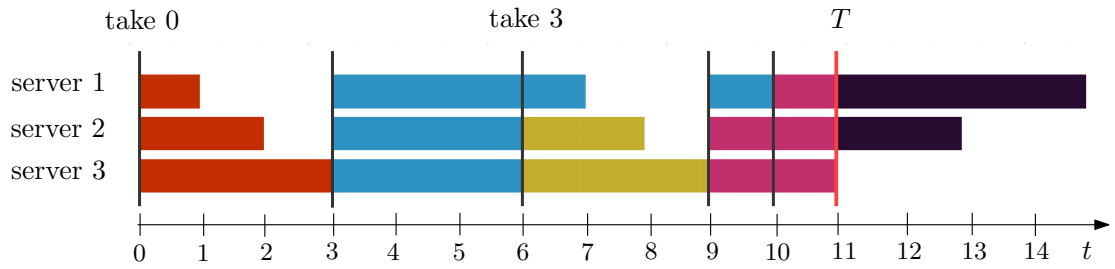

Figure 5.4: Optimal schedule for example 5.4.1. At time $t = 0$ all jobs are scheduled, the next take is placed at time $t = 3$, etc.

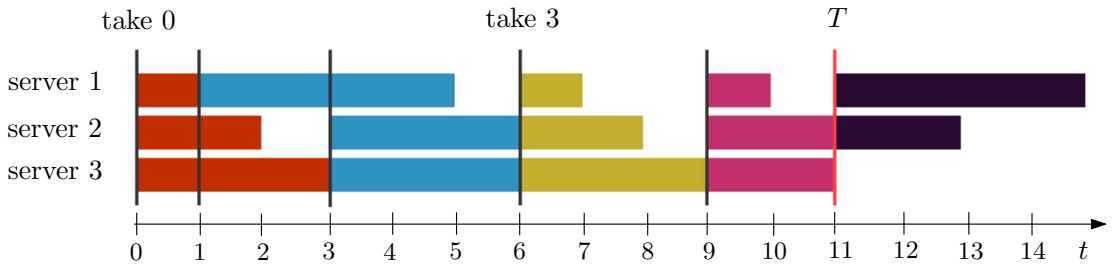

Figure 5.5: Optimal schedule for example 5.4.1. At time $t = 0$ all jobs are scheduled, the next take is placed at time $t = 1$, etc.

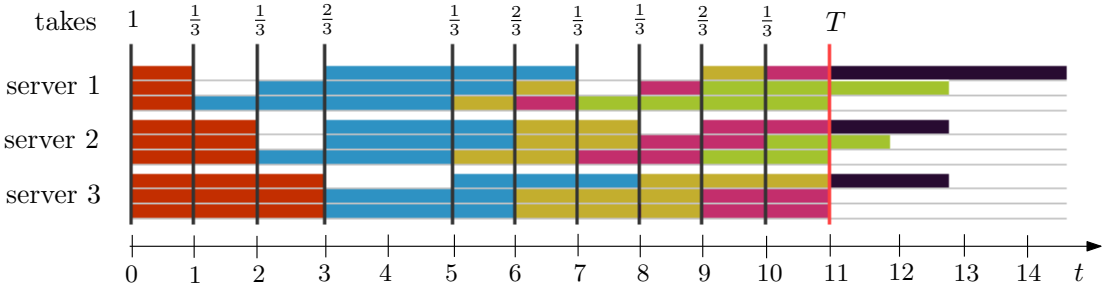

Figure 5.6: Optimal solution for LP for example 5.4.1. Takes are fractional.

at $t = 0$ is 1. At time $t = 1$ the take is $\frac{1}{3}$. Since the first server is idle at $t = 1$ then it starts $\frac{1}{3}$ of the second job (blue color), and continue processing $\frac{1}{3}$ of it during its total length of 4 time units. At time $t = 3$ the take is $\frac{2}{3}$, so all servers can not start more than $\frac{2}{3}$ fraction of some jobs. The first server has only time left to start $\frac{1}{3}$ of the next job, so it starts $\frac{1}{3}$ of the second job (blue color), while servers 2 and 3 have start processing $\frac{2}{3}$ of the second jobs, even though server 3 has $\frac{1}{3}$ of idle time left.

Example 5.4.2.

Input:

```
length = [[1,4,1,1,4],[2,3,2,2,2],[3,3,3,2]]
T = 11
K = 4
```

Optimal schedule has idle time of 6 units.

The LP formulation includes constraints (5.1) - (5.3), (5.9), (5.11), (5.12) - (5.14). *Optimal has idle time of* $3\frac{2}{3}$ *units.*

The LP formulation includes constraints (5.1) - (5.3), (5.5), (5.7), (5.12) - (5.14). *Optimal has idle time of* $3{:}52$ *units.*

We conclude that using constraints (5.9) *and* (5.11) *strengthen LP formulation.*

Example 5.4.3.

Input:

```
length = [[1,3,1,1,5],[3,3,3,2]]
T = 11
K = 4
```

Optimal schedule has idle time of 3 units.

The LP formulation includes constraints (5.1) - (5.3), (5.5) *and* (5.7), (5.12) - (5.14). *Optimal solution has idle time of* $2\frac{1}{3}$ *units.*

The LP formulation includes constraints (5.1) - (5.3), (5.4) *and* (5.6), (5.12) - (5.14). *Optimal solution has idle time of* $2{:}01$ *units.*

We conclude that using constraints (5.5) *and* (5.7) *strengthen LP formulation.*

5.5 Integrality Gap

In this section we construct an example for which the integrality gap is an unbounded function of number of jobs on a server. We employ the notations of the IP formulation

with constraints (5.1) - (5.3), (5.9), (5.11) - (5.14).

Consider the following PLADD scheduling problem with two servers. First server has $3n$ jobs, each of length 2 units. Second server has $2n$ jobs, each of length 3 units. Global time T is $6n$ units, and there are $3n$ takes including the take at time 0. Notice that the total length of jobs on any server equals to T.

Lemma 5.5.1. *Optimal LP solution for the proposed problem has idle time of at most* 1 *unit for* $n \quad 2$.

Proof. Consider the following LP solution:

$$y_0 = 1; y_1 = 0; \text{ and } y_i = 1=2 \text{ for } i = 2; : : : ; T \quad 1;$$

$$x_{0;0;0} = 1; x_{0;i;2i} = 1=2; x_{0;i;2i+1} = 1=2 \text{ for } i = 1; :::; 3n \quad 1;$$

$$x_{1;0;0} = 1; x_{1;i;3i} = 1=2; x_{1;i;3i+1} = 1=2 \text{ for } i = 1; :::; 2n \quad 1;$$

$$z_{0;2} = 1=2 \text{ and } z_{0;3} = 1=2:$$

All other variables equal to 0. Recall that $x_{i;j;t}$ indicates a fraction of job j that starts on server i at time t. All constraints of LP are satisfied, hence the solution is feasible.

Fig. 5.7 provides the schedule for $n = 2$.

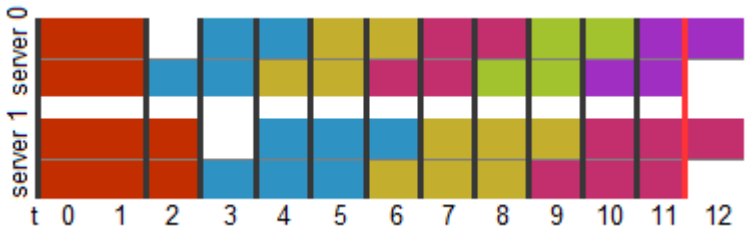

Figure 5.7: For each server blocks of the same color represent the same job. Each server is divided into two parts for visualization of fractional takes. Vertical black lines represent takes, and vertical red line represents the global time T

Total idle time is 1 unit. □

Lemma 5.5.2. *Optimal IP solution for the proposed problem has idle time of n units.*

Proof. Any schedule can be represented as a union of the blocks in Fig. 5.8, since blocks of types $x; y; z$ and t represent all possible combinations of placing takes and not repeating the existing combination. Blocks $x; y; z; t$ can be repeated many times in any order in a schedule, and blocks $x_1; x_2; y_1; y_2; y_3; z_1; z_2; z_3; z_4; t_1; t_2; t_3; t_4$ represent all possible combinations for the last block, if it is not $x; y; z; t$. Note that block y_2 can be represented by x_1, since we only concern about the schedule before the global time T (red vertical line); similarly $y_3; z_3; z_4; t_3; t_4$ are represented by $x_2; y_1; x_1; y_1; x_1$ respectively.

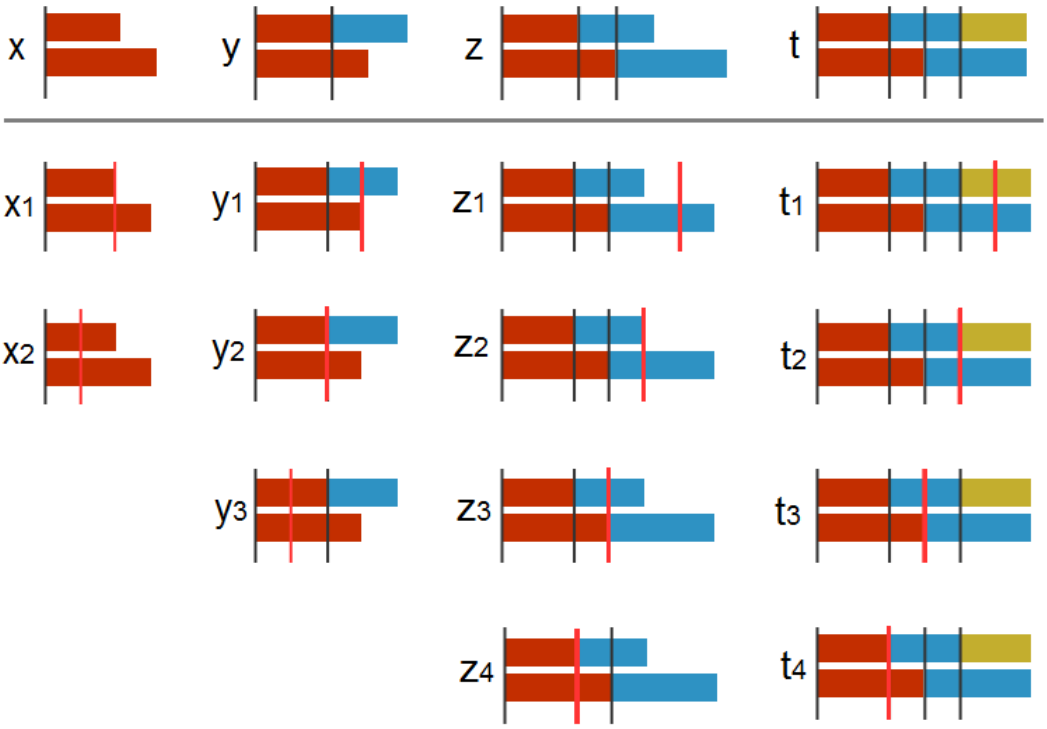

Figure 5.8: Any schedule can be represented as a union of the blocks above. Red vertical line represent global time T

The following is the IP formulation for the problem with $n = 2k$. Each variable represents number of blocks of certain type from Fig. 5.8. For example, variable x equals to the number of blocks of type x in a schedule.

The objective is to minimize idle time:

$$\min x + y + 2z + 0t + 0x_1 + 0x_2 + 0y_1 + 1z_1 + 0z_2 + 0t_1$$

Total number of takes equals $6k$ (for example, in block x there is only one take used):

$$x + 2y + 3z + 4t + x_1 + x_2 + 2y_1 + 3z_1 + 3z_2 + 4t_1 + 3t_2 = 6k$$

Total length of all blocks equals $T = 12k$:

$$3x + 4y + 6z + 6t + x_1 + 2x_2 + 3y_1 + 5z_1 + 4z_2 + 5t_1 + 4t_2 = 12k$$

We can use at most one block from $x_1; x_2; y_1; z_1; z_2; t_1; t_2$:

$$x_1 + x_2 + y_1 + z_1 + z_2 + t_1 + t_2 \quad 1$$

All variables represent number of blocks of a certain type in a schedule:

$$x; y; x; t; x_1; x_2; y_1; z_1; z_2; t_1; t_2 \; 2 \; Z$$

$$x; y; x; t; x_1; x_2; y_1; z_1; z_2; t_1; t_2 \quad 0$$

Consider LP relaxation of IP. The following is a basic feasible solution: $x = 2k \quad 1=3; t = k \quad 1=6; x_2 = 1$, all other variables are zero, since there are exactly 12 (we add one slack variable for the inequality to transform the problem into canonical form) linearly independent active constraints. Reduced costs vector corresponding to $(x; y; z; t; x_1; x_2; y_1; z_1; z_2; t_1; t_2; s)$ is $(0; 1=3; 1; 0; 2=3; 0; 1=3, 1; 2=3; 1; 2=3; 1=3)$. We conclude that this vertex is optimal solution to the LP. Optimal value of the objective function is $2k \quad 1=3$. Notice that $Opt_{LP} \quad Opt_{IP}$ and the cost vector is integer, hence

optimal value for the IP is bounded below by $2k$.

The following schedule achieves the lower bound $2k$ units of idle time. Consider an instance of the problem with $n = 2k$ and create $2k$ block of type t and $2k$ blocks of type x. Blocks of type x have one unit of idle time each. Total idle time is $2k$ units. Fig. 5.9 shows optimal schedules for $n = 2; 4$.

Similar argument can be applied for odd values of n. The optimal schedule for any n has n units of idle time. □

Figure 5.9: The top and bottom schedules are optimal for the problems with $n = 2$ and $n = 4$

5.6 Visualization Software

The Schedule Simulator (see Fig. 5.10) allows the visualization and calculation of optimal configurations for small instances. A user can interact with the schedule by dragging and dropping synchronization points. It provides tools for creating custom scenarios where a user can add/delete jobs from the end or servers from the bottom using buttons + and − correspondingly. In order to change a job length a user needs to select the job and enter its length in | - + Job.

Buttons Use T and Don't use T switch between problems with and without global time T correspondingly.

Capture and Hide buttons store a snapshot of a previously considered example.

Undo and Redo corresponds to cancel/recover previous versions.

There is an option between Brute and Greedy algorithms (heuristics). The first one considers all possible cases and chooses the one with the smallest total idle time.

Save and Load provides a tool to save and load scenarios.

Export CSV saves the idle times of all finite cases explored by the optimization algorithms.

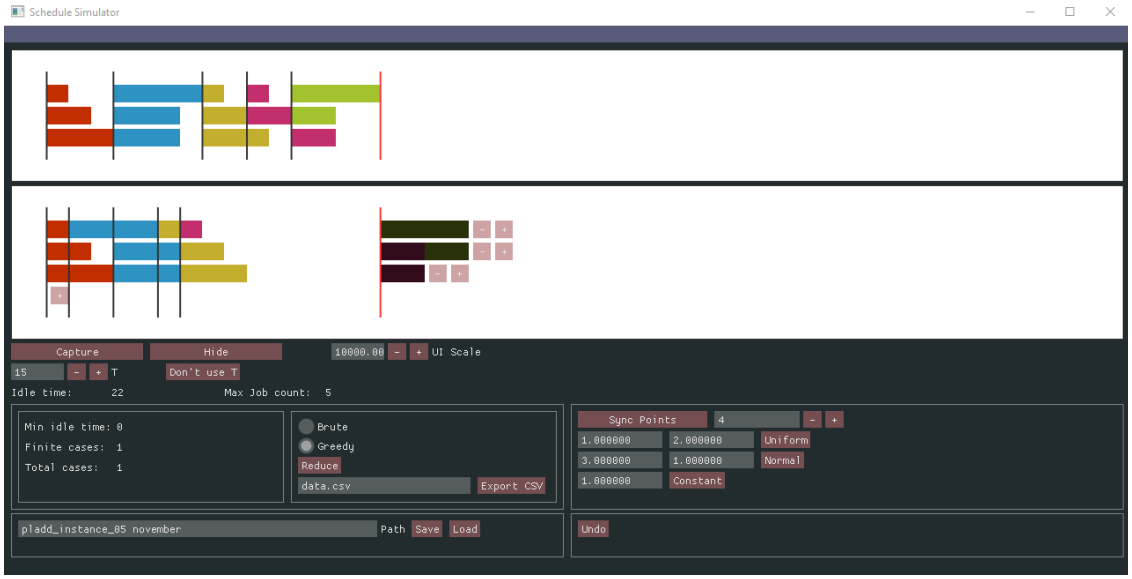

Figure 5.10: A screenshot of Schedule Simulator application

The source code and the built application can be found on GitHub at `https://github.com/VladlenaPowers/PLADD`.

Chapter 6

Conclusions

In this thesis we studied several problems arising in popular matchings and scheduling, proving positive and negative complexity results, investigating the effectiveness of current tools for attacking those problems, as well as providing new ones.

In Chapter 2, we settle the complexity of many open problems around popular matchings in bipartite graphs [12, 13, 41, 28, 33]. In particular, we developed a framework that allows to show NP-hardness and inapproximability results for problems on popular and stable matchings, as well as providing new, and we believe simpler, proofs for recently established problems in the literature. This framework is based on a reduction from a variant of the classical 3-SAT.

In Chapter 4, we develop this framework further. First, we introduce a certain concept of distance between the sets of stable (resp. popular) and Pareto-optimal matchings. We show that hardness of approximation results for (a variant of) 3-SAT can be employed to show that this distance is, in general, very hard to approximate. We complement those results showing special cases in which it can be computed exactly, by exploiting the lattice structure of stable matchings.

We believe that future work in this area could expand our framework further, and apply it to more problems on popular matchings, and more generally to matchings under preferences.

In Chapter 3, we show that some problems around popular matchings are solvable in polynomial time if we restrict the input graph to have bounded treewidth. The latter is a classical assumption, often turning intractable problems into tractable ones.

However, for popular matchings, classical techniques do not seem to apply, and we have to develop a suitable generalization of the concept of popularity, that we called *local popularity*. The algorithm we propose is, for the moment, of theoretical interest only. However, we believe that this result could be the starting point for applying the concept of popular matchings to real-world problems whose underlying graphs has small treewidth. Indeed, stable matchings are found in many real-world applications, and replacing them with popular matching (or subclasses of those) could increase the cardinality of the output matching, while preserving a certain notion of global stability.

Last, in Chapter 5, we analyze a combinatorial formulation for PLADD game introduced in [30]. This game represents an abstract model for moving target defense strategies that allows to mitigate sophisticated cyber attacks by incorporating randomness or changing static parts of the system into dynamic.

The problem results in finding a schedule that minimizes the total idle time for a given number of servers and ordered lists of jobs on each of them. We propose a natural integer programming formulation where variables represent jobs and takes, and we count minimize total idle time. We study multiple instances that allowed us to identify and strengthen certain parts of the formulation. The linear relaxation of this formulation introduces fractional values for the variables, for which we provided intuition and meaning. We prove that the integrality gap of all those formulations is unbounded. The software *The Schedule Simulator* was developed to find an optimal schedule for small instances by enumerating all possible scenarios and visualizing any schedule for analysis. It is available on GitHub at `https://github.com/VladlenaPowers/PLADD`. It also allows to build custom instances in an interactive manner and experiment with different heuristics.

Future work on this problem could potentially include a development of a heuristic that would result in an approximation algorithm with a constant approximation ratio,

analysis of the integrality gap for a maximization version of the problem, or different integer programming formulations. This new scheduling problem could also lead to new insights in other areas of research, such as adaptive machine learning.

Bibliography

[1] A. Abdulkadiroğlu and T. Sönmez. Random serial dictatorship and the core from random endowments in house allocation problems. *Econometrica*, 66(3): 689–701, 1998.

[2] A. Abdulkadiroğlu and T. Sönmez. School choice: A mechanism design approach. *American economic review*, 93(3):729–747, 2003.

[3] A. Abdulkadiroğlu, P. A. Pathak, and A. E. Roth. Strategy-proofness versus efficiency in matching with indifferences: Redesigning the nyc high school match. *American Economic Review*, 99(5):1954–78, 2009.

[4] D. J. Abraham, K. Cechlárová, D. F. Manlove, and K. Mehlhorn. Pareto optimality in house allocation problems. In *International Symposium on Algorithms and Computation*, pages 3–15. Springer, 2004.

[5] D. J. Abraham, A. Blum, and T. Sandholm. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In *Proceedings of the 8th ACM conference on Electronic commerce*, pages 295–304, 2007.

[6] G. Aggarwal, S. Muthukrishnan, D. Pál, and M. Pál. General auction mechanism for search advertising. In *Proceedings of the 18th international conference on World wide web*, pages 241–250, 2009.

[7] P. Biró, R. W. Irving, and D. F. Manlove. Popular matchings in the marriage and roommates problems. In *International Conference on Algorithms and Complexity*, pages 97–108. Springer, 2010.

[8] L. Bodin and A. Panken. High tech for a higher authority: The placement of graduating rabbis from hebrew union college—jewish institute of religion. *Interfaces*, 33(3):1–11, 2003.

[9] H. L. Bodlaender. A tourist guide through treewidth. *Acta cybernetica*, 11(1-2): 1, 1994.

[10] H. L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on computing*, 25(6):1305–1317, 1996.

[11] H. K. Büning and T. Lettmann. *Propositional logic: deduction and algorithms*, volume 48. Cambridge University Press, 1999.

[12] Á. Cseh. Popular matchings. *Trends in Computational Social Choice*, 105, 2017.

[13] Á. Cseh and T. Kavitha. Popular edges and dominant matchings. *Mathematical Programming*, 172(1-2):209–229, 2018.

[14] Á. Cseh, Y. Faenza, T. Kavitha, and V. Powers. Understanding popular matchings via stable matchings. *arXiv preprint arXiv:1811.06897*, 2018.

[15] Y. Faenza and X. Zhang. Legal assignments and fast eadam with consent via classical theory of stable matchings. *arXiv preprint arXiv:1809.08506*, 2018.

[16] Y. Faenza, V. Powers, and X. Zhang. Two-sided popular matchings in bipartite graphs with forbidden/forced elements and weights. *arXiv preprint arXiv:1803.01478*, 2018.

[17] Y. Faenza, T. Kavitha, V. Powers, and X. Zhang. Popular matchings and limits to tractability. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2790–2809. SIAM, 2019.

[18] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[19] P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioral Science*, 20(3):166–173, 1975.

[20] S. Gupta, P. Misra, S. Saurabh, and M. Zehavi. Popular matching in roommates setting is np-hard. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2810–2822. SIAM, 2019.

[21] V. Guruswami and S. Khot. Hardness of max 3sat with no mixed clauses. In *20th Annual IEEE Conference on Computational Complexity (CCC'05)*, pages 154–162. IEEE, 2005.

[22] D. Gusfield and R. W. Irving. *The stable marriage problem: structure and algorithms*. MIT press, 1989.

[23] J. Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

[24] M. Hirakawa, Y. Yamauchi, S. Kijima, and M. Yamashita. On the structure of popular matchings in the stable marriage problem-who can join a popular matching. In *the 3rd International Workshop on Matching under Preferences (MATCH-UP)*, 2015.

[25] G. J. Hitsch, A. Hortaçsu, and D. Ariely. Matching and sorting in online dating. *American Economic Review*, 100(1):130–63, 2010.

[26] G. J. Hitsch, A. Hortaçsu, and D. Ariely. Matching and sorting in online dating. *American Economic Review*, 100(1):130–63, 2010.

[27] C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. *Information and Computation*, 222:180–194, 2013.

[28] C.-C. Huang and T. Kavitha. Popularity, mixed matchings, and self-duality. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2294–2310. SIAM, 2017.

[29] R. W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM (JACM)*, 34(3):532–543, 1987.

[30] S. Jones, A. Outkin, J. Gearhart, J. Hobbs, J. Siirola, C. Phillips, S. Verzi, D. Tauritz, S. Mulder, and A. Naugle. Evaluating moving target defense with pladd. *Sandia National Laboratories*, 2015.

[31] K. Joshi and S. Kumar. Matchmaking using fuzzy analytical hierarchy process, compatibility measure and stable matching for online matrimony in india. *Journal of Multi-Criteria Decision Analysis*, 19(1-2):57–66, 2012.

[32] T. Kavitha. A size-popularity tradeoff in the stable marriage problem. *SIAM Journal on Computing*, 43(1):52–71, 2014.

[33] T. Kavitha. Popular half-integral matchings. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[34] T. Kavitha. Max-size popular matchings and extensions. *arXiv preprint arXiv:1802.07440*, 2018.

[35] T. Kavitha. The popular roommates problem. *arXiv preprint arXiv:1804.00141*, 2018.

[36] J. Kennes, D. Monte, and N. Tumennasan. The day care assignment: A dynamic matching problem. *American Economic Journal: Microeconomics*, 6(4):362–406, 2014.

[37] O. Kesten. School choice with consent. *The Quarterly Journal of Economics*, 125(3):1297–1348, 2010.

[38] A. Laszka, B. Johnson, and J. Grossklags. Mitigating covert compromises. In *International Conference on Web and Internet Economics*, pages 319–332. Springer, 2013.

[39] A. Laszka, G. Horvath, M. Felegyhazi, and L. Buttyán. Flipthem: Modeling targeted attacks with flipit for multiple resources. In *International Conference on Decision and Game Theory for Security*, pages 175–194. Springer, 2014.

[40] B. M. Maggs and R. K. Sitaraman. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66, 2015.

[41] D. Manlove. *Algorithmics of matching under preferences*, volume 2. World Scientific, 2013.

[42] H. Okhravi, M. Rabe, T. Mayberry, W. Leonard, T. Hobson, D. Bigelow, and W. Streilein. Survey of cyber moving target techniques. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON LINCOLN LAB, 2013.

[43] M. Ostrovsky. Stability in supply chain networks. *American Economic Review*, 98(3):897–923, 2008.

[44] V. Pham and C. Cid. Are we compromised? modelling security assessment games. In *International Conference on Decision and Game Theory for Security*, pages 234–247. Springer, 2012.

[45] M. J. Rosenfeld, R. J. Thomas, and S. Hausen. Disintermediating your friends: How online dating in the united states displaces other ways of meeting. *Proceedings of the National Academy of Sciences*, 116(36):17753–17758, 2019.

[46] A. E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of political Economy*, 92(6):991–1016, 1984.

[47] A. E. Roth, T. Sönmez, and M. U. Ünver. Kidney exchange. *The Quarterly journal of economics*, 119(2):457–488, 2004.

[48] A. E. Roth, T. Sönmez, and M. U. Ünver. Pairwise kidney exchange. *Journal of Economic theory*, 125(2):151–188, 2005.

[49] M. A. Satterthwaite and H. Sonnenschein. Strategy-proof allocation mechanisms at differentiable points. *The Review of Economic Studies*, 48(4):587–597, 1981.

[50] L. Shapley and H. Scarf. On cores and indivisibility. *Journal of mathematical economics*, 1(1):23–37, 1974.

[51] M. Van Dijk, A. Juels, A. Oprea, and R. L. Rivest. Flipit: The game of "stealthy takeover". *Journal of Cryptology*, 26(4):655–713, 2013.