Complexity and algorithms in matching problems under preferences

vorgelegt von
Ágnes Cseh, M.Sc.
Szolnok

Von der Fakultät II – Mathematik und Naturwissenschaften
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Naturwissenschaften
– Dr. rer. nat. –

genehmigte Dissertation

Promotionsausschuss

Vorsitzender:	Prof. Dr. Volker Mehrmann
Gutachter:	Prof. Dr. Martin Skutella
Gutachter:	Prof. Dr. Gerhard J. Woeginger

Tag der wissenschaftlichen Aussprache: 7. Dezember 2015

Berlin 2016

Our result provides a handy counterexample to some of the stereotypes which non-mathematicians believe mathematics to be concerned with. Most mathematicians at one time or another have probably found themselves in the position of trying to refute the notion that they are people with a head for figures or that they know a lot of formulas. At such times it may be convenient to have an illustration at hand to show that mathematics need not be concerned with figures. [...] The argument [of our theorem] is carried out not in mathematical symbols but in ordinary English; there are no obscure or technical terms. Knowledge of calculus is not presupposed. In fact, one hardly needs to know how to count. Yet any mathematician will immediately recognize the argument as mathematical, while people without mathematical training will probably find difficulty in following the argument, though not because of unfamiliarity with the subject matter.

What, then to raise the old question, is mathematics? The answer, it appears, is that any argument which is carried out with sufficient precision is mathematical.

D. Gale and L. S. Shapley.
College admissions and the stability of marriage. 1962

Acknowledgment

I am grateful to my supervisor Martin Skutella for all the advice, time and encouragement he gave me in the past years. Martin did not only guide me through his supervision, but also gave me full support whenever I showed interest in applying for grants, workshops or research stays, which happened quite frequently.

I thank Brian C. Dean for hosting me at Clemson University for 4 months in 2013. He facilitated a very vivid, active and friendly work relationship.

Right after my stint at Clemson, David F. Manlove hosted me at the University of Glasgow for 7 months in 2014. I felt immediately welcomed in Scotland. Through David's patience and enormous knowledge about matchings under preferences, I learned a lot.

In 2015, Telikepalli Kavitha hosted me at the Tata Institute of Fundamental Research in Mumbai for 3 months. Kavitha, besides being a strikingly friendly person, also inspired me to tackle quite challenging and beautiful problems.

The time spent at my home institution was wonderful, due to the COGA spirit which I will badly miss. Thanks to all Coganer, and among them, special thanks to Dorothea Kiefer who made administrative tasks so much easier during these years.

The silent guardians of this thesis are Martin Groß, Jannik Matuschke and Julie Meißner. Their comments on earlier versions lead to an improved presentation of the thesis. I am especially grateful to Ashwin Arulselvan, who corrected my peculiar phrasings in this thesis and often outside of it.

For their whole-hearted support (and for showing me what a stable marriage is in practice), I wish to thank my parents.

I also thank the Deutsche Telekom Stiftung for funding my PhD program and my various stints in abroad. Their generous support made it possible for me to gain experience in several groups. Some of these trips were partially funded by the Research Training Group 'Methods for Discrete Structures'. I also received financial support from the ICT COST Action IC1205, for which I am grateful.

Berlin, August 2015 Ágnes Cseh

Contents

Introduction

Motivation

The National Resident Matching Program (NRMP) is a non-profit organization created in 1952 to centrally match medical school graduates to residency positions in the United States. The main motivation behind founding NRMP was the chaotic competitive procedure students had to go through in order to receive a suitable residency contract. Since there had been no centralized framework, students could easily benefit from misreporting their true preferences and applying for jobs years before the starting date, which forced hospitals to grant contracts to the promising candidates well ahead of time unless they were ready to risk losing them [86]. This latter problem was so urgent that the Association of American Medical Colleges called for a centralized solution as early as in 1927 [31].

In the academic year of 1950-51, a dry run of the developed centralized algorithm was performed and student bodies were informed about the procedure. The medical students immediately protested claiming that the algorithm was more beneficial to the hospitals than to the students and it did not resolve the problem of students benefiting from reporting false preferences, thus encouraging them to trick the system. The NRMP then seemingly altered the matching procedure due to these complaints [103] – yet implemented essentially the same method, which was in use until 1997, even though by that time numerous renowned scientists pointed out the failures of the system with mathematical rigor and voiced their concerns against it [49, 86, 103, 104]. The modified algorithm used nowadays is described in [89] in detail. In 2015 alone, the NRMP handled applications from over 41000 medical students [107].

The major characteristic feature of the desired resident allocation is straightforward. If a student is not assigned to a specific hospital, then it must be either because the hospital could fill up all its positions with more suitable candidates or because the student was assigned to another hospital that she ranked better. This notion is exactly captured by stable matchings, defined with mathematical rigor by Gale and Shapley in 1962 [44].

Besides resident allocation, variants of the stable matching problem are widely used in other employer allocation markets [91], university admission decisions [9, 19], campus housing assignments [24, 84] and bandwidth allocation [43]. A recent honor proves the currentness and importance of results in the topic: in 2012, Lloyd S. Shapley and Alvin E. Roth were awarded the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel for their outstanding results on market design and matching theory.

In this thesis, we discuss various problems in stable matchings from the algorithmic point of view, either presenting an efficient algorithm for solving them or proving hardness.

Outline of the thesis

The problems investigated in all 7 chapters of this thesis are related to stable matchings and organized based on the complexity of the instance they are defined on. We start with discussing various problems in the simplest, one-to-one matching setting and then through more complex capacitated instances we move to problems defined on network flow instances. Finally, we also discuss an alternative optimality notion and its relation to stable matchings.

Chapter 1: Basic notions in stable matchings. We introduce the concept of stable matchings formally and present some of the most important theorems related to it, including the Gale-Shapley algorithm. Real-life applications are also discussed briefly.

Chapter 2: Stable marriage and roommates problems with restricted edges. We start with classical one-to-one matchings in bipartite and non-bipartite graphs and investigate the problem of stable matchings when forced and forbidden edges are present. A stable solution must contain all of the forced pairs, while it must contain none of the forbidden pairs. Two approximation concepts are described here: (1) the matching is blocked by as few pairs as possible, or (2) it violates as few constraints as possible on restricted pairs. This chapter is a comprehensive study of complexity and approximability results in these two problems.

Chapter 3: Other complexity results for stable matchings. In this chapter we investigate two problems in the stable matching setting. The first problem is defined on bipartite stable matching instances with free edges. Free edges can appear in stable matchings, but they are not able to block matchings by definition. We show that a maximum cardinality stable matching is NP-hard to find. The second problem tackled is a degree constrained version of the stable roommates problem with ties, which is known to be NP-complete [85] in the general case. Here we present complexity results depending on the highest degree in the graph.

Chapter 4: Paths to stable allocations. We introduce the stable allocation problem, a capacitated variant of the stable marriage problem. There, the agents are often called jobs and machines. We investigate the case of uncoordinated processes in stable allocation instances. In this setting, a feasible allocation is given and jobs and machines are allowed to selfishly modify it. We analyze both better and best response dynamics from an algorithmic point of view and discuss deterministic and random procedures as well.

Chapter 5: Unsplittable stable allocation problems. In this chapter we study a natural "unsplittable" variant of the stable allocation problem, where each assigned job must be fully assigned to a single machine. Our main result is to show that the problem is solvable in polynomial time. We also show that in the event there is no feasible solution, our approach computes a solution of minimal total congestion (overfilling of all machines collectively beyond their capacities). We also describe a technique for rounding the solution of a stable allocation problem to produce “relaxed” unsplit solutions that are only mildly infeasible, where each machine is overcongested by at most a single job.

Chapter 6: Stable flows. As most matching problems, stable matchings also can be extended to network flows. In this chapter we present the polynomial version of the Gale-Shapley algorithm for stable flows. Then, a direct combinatorial algorithm for stable flows with forced and forbidden edges is presented. Finally, we study stable multicommodity flows and show that it is NP-complete to decide whether an integral solution exists.

Chapter 7: Popular matchings. In the last chapter we discuss an optimality notion that can be seen as an alternative to stability. A matching M in a bipartite graph G with preferences is popular if there is no matching M' such that the vertices that prefer M' to M outnumber those that prefer M to M'. We investigate two problems. The first problem is defined on graphs with strict preferences on both sides. We identify a natural subclass of popular matchings called “dominant matchings” and show that every dominant matching in G can be realized as an image of a stable matching in a modified graph G'. In the second setting, ties might occur in the preferences, but only on one side of G. We show that the problem of deciding whether G admits a popular matching is NP-hard even in very restricted instances.

General concepts

We assume that the reader is familiar with basic concepts in graph theory such as bipartite graphs, paths, cycles, walks, directed networks, and so on. For any missing reference, we advise to turn to the book of West [102].

Running time of an algorithm. An algorithm is said to run *in polynomial time* if there is a polynomial p such that the number of elementary steps taken by the algorithm is bounded by $p(x)$ for every possible input, where x is the encoding size of the problem input. Constant factors are often omitted when discussing running times. The notation $\mathcal{O}(p(x))$ stands for all polynomial running times $r(x)$ so that $r(x) \leq cp(x)$ for some constant $c \in \mathbb{R}_{>0}$ and sufficiently large x. Since the focus of this thesis is complexity theory, we will call all polynomial algorithms efficient, even though in practice, only polynomials of very low degree are widely considered to be truly efficient.

NP-completeness. A decision problem is in the complexity class P, if it admits a polynomial time algorithm. We also call these problems tractable. A superclass of P is NP: a decision problem is in the complexity class NP, if each “yes” instance has a polynomial size certificate. A problem is NP-*hard* if all problems in class NP can be polynomially reduced to it. A decision problem is NP-*complete* if it is NP-hard and it is also in NP. It is common to assume that P $\neq$ NP, thus NP-hard and NP-complete problems are assumed to be computationally intractable.

Satisfiability problems. In a *satisfiability problem*, a Boolean formula is given and the question is whether there is a truth assignment of the variables so that the formula is satisfied. One of the most fundamental NP-complete problems is 3-SAT, where the Boolean formula is given in conjunctive normal form (CNF) and each clause contains at most 3 literals. This and also other variants of satisfiability problems will be used later in this thesis to prove NP-hardness of numerous problems discussed. For background on these and other notions in complexity theory, we refer the reader to the book of Wegener [101].

Approximation. When discussing optimization problems, one often talks about an approximate solution. A minimization problem $\min c(x)$, where $c(x) \geq 0$ for all feasible solutions x, admits an *α-approximation*, $\alpha \geq 1$, if there is a polynomial algorithm delivering a feasible solution x^* so that $c(x^*) \leq \alpha c(x_{\text{opt}})$, where x_{opt} is an optimal solution. In this thesis we will show approximation algorithms and inapproximability results as well with constant and instance-dependent α values. See the book of Williamson and Shmoys [105] for more information about approximation algorithms.

Distributive lattice. A partially ordered set (*poset*) is a pair $P = (S, \leq)$, where the binary relation $\leq$ is reflexive, transitive and antisymmetric on set S. If any two elements $s_1, s_2 \in S$ have a unique lowest upper bound $s_1 \vee s_2$ and a unique greatest lower bound $s_1 \wedge s_2$ with respect to $\leq$, then P is called a *lattice*. Moreover, if $s_1 \vee (s_2 \wedge s_3) = (s_1 \vee s_2) \wedge (s_1 \vee s_3)$ for arbitrary $s_1, s_2, s_3 \in S$, then the lattice is *distributive*. More details about these notions can be found in the book of Davey and Priestley [32].

Milestones in the literature

Until today, four books have been published on stable matchings. Each of them approaches the problem from a different point of view and provides the reader with the research focus of different times. The first book, written by Knuth [70], can be used as lecture notes, includes several exercises and established connections with various combinatorial problems. It also contains the famous 12 problems of Knuth that since then have served as targets for the stable matching community. In 1989, Gusfield and Irving published their elaborate study on the underlying structure of stable matchings [49]. An updated list of 12 major problems can also be found in this work. Two years later, a book from a game-theoretic viewpoint was published, authored by Roth and Sotomayor [91]. This book reflects the interests of the economics society related to matchings under preferences. Very recently, Manlove wrote a detailed study on the complexity and algorithms of matchings under preferences [73]. Besides stability, he also elaborates on alternative optimality concepts such as Pareto-optimality or popular matchings. This book served as an excellent source of currently known results and relevant literature while writing this thesis.

1 Basic notions in stable matchings

Matching problems lie at the heart of discrete mathematics. Besides their importance in theory, maximum cardinality and minimum cost matchings are clearly among the problems with the broadest scale of applications [6]. In this thesis we focus on matching markets under preferences – each market participant expresses their preferences as an ordered list of possible scenarios. Our task is to find a matching that is optimal with respect to these preferences. The most common notion of optimality is stability, formally defined below.

In this introductory chapter we give an overview of the most relevant results in stable matchings. Then, we also show various examples on how the theory is used in practice.

1.1 Theoretical background

1.1.1 The stable marriage problem

Stable matchings were first formally defined in the seminal paper of Gale and Shapley [44]. They described an instance of the college admission problem and introduced the terminology based on marriage that since then became wide-spread.

In the classical *stable marriage problem*, a bipartite graph $G = (V = (U \cup W), E)$ is given, where one side symbolizes a set of men U and the other side stands for a set of women W. The vertices in G are often called agents as well. Man u and woman w are connected by edge $uw \in E$ if they find one another mutually acceptable. Unless it is otherwise stated, we assume that G is not necessarily a complete bipartite graph.

Each agent in G provides a strictly ordered *preference list* of the acceptable agents of the opposite gender (see also Figure 1.1). The a set of these preference lists is denoted by O. We will now introduce the used notation focusing only on a $u \in U$ vertex, and remark that there is an analogous version for $w \in W$. Vertex u prefers w_1 to w_2 if w_1 has a better rank (a lower number) on u's preference list than w_2. In this case we say that w_1 *dominates* w_2 at u and denote it by $w_1 \geq_u w_2$, following the convention. The same can also be expressed with the help of the *rank function* $\text{rank}_{u \in U} : N(u) \to \mathbb{Z}_{>0}$, where $N(u)$ stands for the agents adjacent to u. We define $\delta(u) \subseteq E(G)$ to be the set of edges incident to u and $\deg(u) = |N(u)|$ to be the degree of u. Using this notation, $w_1 \geq_u w_2$ is equivalent to $\text{rank}_u(w_1) \leq \text{rank}_u(w_2)$. Occasionally we introduce non-integer ranks when adding new agents to lists in our proofs. For convenience we also allow ranking and comparing edges instead of vertices: we can write $uw_1 \geq_u uw_2$ or $\text{rank}_u(uw_1) \leq \text{rank}_u(uw_2)$. Being unmatched is always considered as a less beneficial position than being matched to anyone on the preference list.

A set of edges M is a *matching* if every vertex is adjacent to at most one edge in M. The vertex matched to $u \in U$ by M is denoted by $M(u)$.

Definition 1.1 (blocking edge, stable matching)**.** *An edge $uw \in E$* blocks *matching M if it fulfills the following three requirements:*

1. *$uw \notin M$;*
2. *$M(u) = \emptyset$ or $w \geq_u M(u)$;*
3. *$M(w) = \emptyset$ or $u \geq_w M(w)$.*

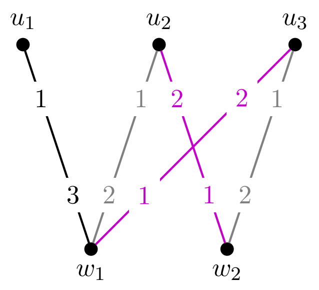
A stable matching *is a matching not blocked by any edge.*

Later on in this thesis, we will modify these three points several times to define the blocking element of more and more complex instances, such as blocking walks to network flows.

Problem 1. SM
Input: *$\mathcal{I} = (G, O)$; a bipartite graph $G = (V = (U \cup W), E)$ and a set of strictly ordered preference lists O.*
Question: *Is there a matching not blocked by any edge?*

In this thesis, we directly define the problem on instances with incomplete lists. In various complexity results, the completeness of the bipartite graph plays a crucial role. Note that an instance with incomplete lists can easily be transformed into an instance with complete lists. Assume that $u \in U$ has an incomplete list. We first introduce a dummy vertex w_u and add it at the bottom of the initial list of u. The women not yet listed by u can then be added to the bottom of u's list in arbitrary order. The most preferred man of w_u is u, then all the men follow in arbitrary order. Analogous operations are executed for all vertices in $V(G)$. With this construction, each vertex is either matched to someone on its initial list or to its dummy counterpart.

When illustrating stable matching instances, there are two widely accepted variants, as presented in Figure 1.1. One of them is a graph-oriented interpretation while the second one is focused on preference lists. In this thesis we will mostly rely on the graph-based illustration, but whenever the instance size is very prohibitive for it, we will use a list based structure.



u_1:	w_1		
u_2:	w_1	w_2	
u_3:	w_2	w_1	
w_1:	u_3	u_2	u_3
w_1:	u_2	u_3	

Figure 1.1: An SM instance with stable matchings $\{u_2w_2, u_3w_1\}$ and $\{u_2w_1, u_3w_2\}$. The graph-based illustration is on the left side, while the list-based one is to the right.

The Gale-Shapley algorithm

The linear-time Gale-Shapley algorithm provided the first proof for the existence of stable matchings. In the past decades numerous variants of it were developed to solve different stable matching problems. Some of them are presented in later chapters of this thesis.

The deferred acceptance algorithm of Gale and Shapley can be outlined in the following way. Each round of the algorithm consists of two steps: the proposal and the acceptance/refusal steps. In the very fist step, all men propose along their best edge. In the second step of the same round, women who received proposals keep the single edge ranked best by them and refuse the rest of the proposals. Refused edges are deleted from the graph. Now the second round starts with the currently unmatched men submitting proposals along their best edge not yet deleted from the graph. Then, women compare these offers and their currently accepted edge and only keep the best one. Such rounds are executed until there is no unmatched man whose edges have not been deleted yet. Since every edge can be proposed along and accepted or rejected at most once, the running time is bounded by $\mathcal{O}(|E|)$.

Theorem 1.2 (Gale, Shapley [44]). *For any instance $\mathcal{I}$ of* SM*, the Gale-Shapley algorithm delivers a stable matching in $\mathcal{O}(|E|)$ time.*

Lattice structure

The Gale-Shapley algorithm delivers one stable matching, but an instance can have exponentially many stable solutions. Stable matchings in an instance admit a rich structure.

Theorem 1.3 (Knuth [70], attributed to Conway). *Stable matchings in a fixed* SM *instance form a distributive lattice.*

The two extreme points of this lattice are called the *man-* and *woman-optimal stable matchings* [44]. The man-optimal stable matching assigns each man his best partner reachable in any stable matching. At the same time, it is also the woman-pessimal stable matching: it assigns to each woman the worst partner reachable in any stable matching. The analogous statement holds for woman-optimal stable matching. For SM, the Gale-Shapley algorithm can be reversed easily, with women proposing instead of men, to obtain a woman-optimal solution. The join of two stable matchings can be formed by choosing the better edge for every man, while the meet can be reached by choosing the worse edge.

Rural Hospitals Theorem

Another interesting and useful property of stable solutions is the so-called Rural Hospitals Theorem, named after the less popular – typically rural – hospitals in the Hospitals / Residents problem (see below) that cannot fill up their open positions. The crucial part of this theorem states the following:

Theorem 1.4 (Gale, Sotomayor [45])**.** *If an agent is unmatched in one stable matching, then all stable solutions leave her unmatched.*

Rotations

In the following we will sketch some structural results that give rise to a technique used in generating all stable solutions of an SM instance. We are given a stable matching M in $\mathcal{I}$ which is not the woman-optimal stable solution. Our goal is to modify M so that a different stable matching M' is derived.

Let us mark all edges uw of a woman w so that $u >_w M(w)$. Intuitively, these are the potential marriages w is inclined to switch to. Now let all men who have marked edges choose the best marked edge. Note that all these marked edges are worse for men than their matching edge, otherwise M would not be stable. Gusfield and Irving [49] show that the chosen edges and edges of M on the same vertex set form a set of cycles. Each cycle is a *rotation*: augmenting M along it results in a different stable matching. As a matter of fact, more is true; rotations form a partially ordered set (poset), where the relation is the order they can be executed. The closed subsets of this rotation poset are in a one-to-one correspondence with the stable matchings in $\mathcal{I}$. Due to an efficient representation of the rotation poset, stable matchings can be listed in $\mathcal{O}(|V|)$ time per matching, following $\mathcal{O}(|E|)$ pre-processing time. Note that a polynomial algorithm for listing *all* stable matchings in an instance cannot occur, because the number of stable solutions itself can be exponentially large.

In our example instance depicted in Figure 1.1, the gray matching u_2w_1, u_3w_2 is not woman-optimal. Vertices w_1 and w_2 mark edges u_3w_1 and u_2w_2, respectively, which are also the best marked edges of men u_3 and u_2, respectively. These four edges thus form the rotation turning the gray matching into the purple matching.

Rotations opened the gate to solving a very important variant of SM: the weighted stable marriage problem. Here we are given edge weights on G and we seek a stable matching with minimum total weight. Each rotation can be assigned a weight depending on the edges it adds and edges it eliminates when it is augmented along. Finding a minimum weight stable matching then translates into finding a minimum weight closed subset in the rotation poset. Weighted stable matchings model various problems, such as fair stable marriage or stable marriage with restricted pairs, as demonstrated in Chapter 2.

1.1.2 Extensions of the stable marriage problem

Many-to-one matching

This generalization of SM is also know as the *Hospitals / Residents problem* and plays a crucial role in applications. As defined by Gale and Shapley [44], the problem instance involves residents as vertices on one side and hospitals as vertices on the other side of G. While residents need to be matched to at most one hospital, hospitals submit an upper quota on the number of admitted residents. The definition of stability can be translated easily to such b-matching instances. It is also straightforward to see that many-to-many stable matchings can be defined analogously. Later in this thesis, we will define stability on more complex instances along the line of one-to-one, many-to-one and many-to-many matchings.

Non-bipartite instances

One of the most widely studied extensions of SM is the *stable roommates problem* [44, 52], defined on general graphs instead of bipartite graphs. The notion of a blocking edge is as defined above, except that it can now involve any two agents in general.

Problem 2. SR
Input: *$\mathcal{I} = (G, O)$; a not necessarily bipartite graph $G = (V, E)$ and a set of strictly ordered preference lists O.*
Question: *Is there a matching not blocked by any edge?*

Several results for SM do not carry over to this setting. Most importantly, the existence of a stable solution is not guaranteed any more. However, there is a linear-time algorithm to find a stable matching or report that none exists [52]. Moreover, the corresponding variant of the Rural Hospitals Theorem holds in the roommates case as well: the set of matched agents is the same for all stable solutions [49].

Ties in preferences

Another natural extension is the *stable matching problem with ties*. So far we have assumed that each agent ranks all its neighbors strictly. Especially in large instances in practice it is an unrealistic assumption to make. Introducing *ties*, i.e., equally ranked neighbors in preference lists radically changes the complexity of many problems.

First of all, when ties are present, the definition of a blocking edge must be revised. Irving and Manlove [53, 72] define the following three types of stability.

Definition 1.5 (blocking edge in the weakly, strongly and super stable sense)**.** *An edge $uw \in E$* blocks *matching M* weakly *if it fulfills the following three requirements:*

1. *$uw \notin M$;*
2. *$M(u) = \emptyset$ or $w >_u M(u)$;*
3. *$M(w) = \emptyset$ or $u >_w M(w)$.*

An edge $uw \in E$ blocks *matching M* strongly *if it fulfills the first and at least one of the second and third requirements:*

1. *$uw \notin M$;*
2. *$(M(u) = \emptyset$ or $w >_u M(u))$ and $(M(w) = \emptyset$ or $u \geq_w M(w))$ or*
3. *$(M(u) = \emptyset$ or $w \geq_u M(u))$ and $(M(w) = \emptyset$ or $u >_w M(w))$.*

An edge $uw \in E$ blocks *matching M* in the super stable sense *if it fulfills the following three requirements:*

1. *$uw \notin M$;*

2. *$M(u) = \emptyset$ or $w \geq_u M(u)$;*

3. *$M(w) = \emptyset$ or $u \geq_w M(w)$.*

In words, an edge uw blocks M weakly if both u and w strictly prefer one another to their current partners in M, uw blocks M in the super stable sense if u and w do not prefer their current partners strictly to each other, and finally, uw blocks M strongly if u strictly prefers v to $M(u)$, and v does not prefer $M(v)$ to u, or the other way round. In all cases, the stricter notion of blocking implies the weaker notion.

It is trivial that the weakly stable marriage problem can be solved efficiently, since every stable matching for any linear extension of the preference lists remains stable under the weak notion of stability. On the other hand, stable solutions need not exist in the weakly stable roommates problem, in fact, it is NP-complete to decide whether any exists [85]. Many other problems turn out to be computationally hard under weak stability; one of the most vividly studied open questions in matchings under preferences is the best approximation of the NP-complete maximum size weakly stable marriage problem [57, 68, 74].

Deciding whether a super-stable matching exists and if so, producing one is possible in $\mathcal{O}(|E|)$ time for SM [56] and for SR [55] as well. The fastest known algorithm for finding a strongly stable matching or a proof for its nonexistence requires $\mathcal{O}(|V||E|)$ time in SM [61], and $\mathcal{O}(|E|^2)$ time in SR [94].

1.2 Applications

As already indicated in the introduction, the first algorithm to solve SM was discovered and implemented in practice years earlier than the notion of stability was mathematically defined. The medical students protesting in 1951 against a hospital-optimal solution and questioning the truthfulness of the Gale-Shapley algorithm with hospitals proposing actually established important theoretical results about the set of stable solutions and the proposed mechanism. Thus it is not surprising that stable matchings are used widely to model various real-life problems.

Stability is a well-known concept used for matching markets where the aim is to reach a certain type of social welfare, instead of profit-maximizing [86]. The measurement of optimality is not maximum cardinality or minimum cost, but the certainty that no two agents are willing to selfishly modify the market situation. Such markets model resident allocation, university admission decisions or bandwidth allocation. Both SM and SR are widely used in various applications worldwide. An online collection of these applications can be found under [11].

Employer allocation. For SM, the oldest and most common area of applications is employer allocation markets [91]. On one side, job applicants are represented, while the job openings form the other side. Each application corresponds to an edge in the bipartite graph. The employers rank all applicants to a specific job offer and similarly, each applicant sets up a preference list of jobs. Given a proposed matching M of applicants to jobs, if an employer-applicant pair exists such that the position is not filled or a worse applicant is assigned to it, and the applicant received no contract or a worse contract, then this pair blocks M. In this case, the employer and applicant find it mutually beneficial to enter into a contract outside of M undermining its integrity. If no such blocking pair exists, then M is stable. As already mentioned in the Introduction, stability as an underlying concept is also used to allocate graduating medical students to hospitals in many countries [88, 108].

College admission. The second largest application area is higher education admission procedures. There, high-school graduates rank the university programs they apply to, while the universities set up their lists based on the scores of the applicants. A stable matching guarantees that every student is allocated to their most preferred university that could not fill up its quota with students who scored higher. A centralized matching algorithm outputs the college placement of over 140000 high school graduates each year in Hungary alone [13].

P2P networks. SR has applications in the area of peer-to-peer (P2P) networks [43]. In such problems, nodes in a communication network need to be paired up to perform some action, e.g., file transfer from one computer to another or to cooperate in a multiplayer game. Depending on the task, each node ranks all other nodes based on properties, e.g., their upload and download speed, computer performance or physical distance. It is in the interest of each peer to find a stable matching.

Living donor kidney exchange. Many patients with chronic kidney disease are on the waiting list for a transplantation, despite having a volunteering donor, because the donor and the recipient are medically incompatible. If there is a large pool of such incompatible pairs, cross-donations or even donations in cycles can be organized. SR provides a framework suitable for organizing cross-donations centrally, based on preferences calculated from the compatibility of two patient-donor pairs. Living donor kidney exchange programs are organized in several countries, such as the UK [109] or the USA [110]. To the best of our knowledge, these programs operate under cardinal (and not ordinal) preferences, and the main objective is to maximize the number of transplantations. Even though no kidney exchange program seems to currently work on the basis of finding stable matchings, SR provides an alternative framework that is also well-studied in the literature, even with focus on kidney exchange [75, 90].

2 Stable marriage and roommates problems with restricted edges

In this chapter we focus on the stable marriage and stable roommates problems. We investigate the complexity of finding a stable solution satisfying additional constraints on restricted pairs of vertices. Restricted pairs can be either *forced* or *forbidden*. A stable solution must contain all of the forced pairs, while it must contain none of the forbidden pairs.

Dias et al. [36] gave a polynomial-time algorithm to decide whether such a solution exists in the presence of restricted edges. If the answer is "no", one might look for a solution close to optimal. Since optimality in this context means that the matching is stable and satisfies all constraints on restricted pairs, there are two ways of relaxing the constraints by permitting a solution to: (1) be blocked by as few as possible pairs, or (2) violate as few as possible constraints on restricted pairs.

Our main theorems prove that for the stable marriage problem, case (1) leads to NP-hardness and inapproximability results, whilst case (2) can be solved in polynomial time. For stable roommates problem instances, case (2) yields an NP-hard but (under some cardinality assumptions) 2-approximable problem. In the case of NP-hard problems, we also discuss polynomially solvable special cases, arising from restrictions on the lengths of the preference lists, or upper bounds on the numbers of restricted pairs.

The results presented in this chapter are joint work with David F. Manlove and have been published in [30].

2.1 Introduction

Motivation. As mentioned in Chapter 1.2, SM and SR are widely used notions in various market modeling tasks. Forced and forbidden edges in SM and SR open the way to formulate various special requirements on the sought solution. Such edges now form part of the extended problem instance: if an edge is *forced*, it must belong to a constructed stable matching, whilst if an edge is *forbidden*, it must not. In certain market situations, a contract is for some reason particularly important, or to the contrary, not wished by the majority of the community or by some central authority in control. In such cases, forcing or forbidding the edge and then seeking a stable solution ensures that the wishes on these specific contracts are fulfilled while stability is guaranteed. Henceforth, the term *restricted edge* will be used to refer either to a forbidden edge or a forced edge. The remaining edges of the graph are referred as *unrestricted edges*.

Note that simply deleting forbidden edges or fixing forced edges and searching for a stable matching on the remaining instance does not solve the problem of finding a stable matching with restricted edges. Deleted edges (corresponding to forbidden edges, or those adjacent to forced edges) can block that matching. Therefore, to meet both requirements on restricted edges and stability, more sophisticated methods are needed.

Literature review. The attention of the community was drawn very early on to the characterization of stable matchings that must contain a prescribed set of edges. In the seminal book of Knuth [70], forced edges first appeared under the term *arranged marriages*. Knuth presented an algorithm that finds a stable matching with a given set of forced edges or reports that none exists. This method runs in $\mathcal{O}(|V|^2)$ time, where V is the set of vertices in the graph. Gusfield and Irving [49] provided an algorithm based on rotations (see Section 1.1) that terminates in $\mathcal{O}(|Q|^2)$ time, following $\mathcal{O}(|V|^4)$ pre-processing time, where Q is the set of forced edges. This latter method is favored over Knuth's if multiple problems are proposed in the same instance, all of them with forced sets of small cardinality.

Forbidden edges appeared only in 2003 in the literature, and were first studied by Dias et al. [36]. In their paper, complete bipartite graphs were considered, but the methods can easily be extended to incomplete preference lists. Their main result was the following.

Theorem 2.1 (Dias et al. [36])**.** *The problem of finding a stable matching in an* SM *instance with forced and forbidden edges or reporting that none exists is solvable in* $\mathcal{O}(|E|)$ *time.*

While Knuth's method relies on basic combinatorial properties of stable matchings, the other two algorithms make use of rotation (see Section 1.1). The problem of finding a stable matching with forced and forbidden edges can easily be formulated as a weighted stable matching problem (that is, we seek a stable matching with minimum weight, where the weight of a matching M is the sum of the weights of the edges in M). Let us assign all forced edges weight -1, all forbidden edges weight 1, and all remaining edges weight 0. A stable matching satisfying all constraints on restricted edges exists if and only if there is a stable matching of weight $-|Q|$ in the weighted instance, where Q is the set of forced edges. With the help of rotations, minimum weight stable matchings can be found in polynomial time [38, 39, 54].

Since finding a minimum weight stable matching in SR instances is known to be an NP-hard task [38], it follows that solving the problem with forced and forbidden edges requires different methods from the aforementioned weighted transformation. Fleiner et al. [42] showed that any SR instance with forbidden edges can be converted into an SM problem involving ties that can be solved in $\mathcal{O}(|E|)$ time [55] and the transformation has the same time complexity as well. Forced edges can easily be eliminated by forbidding all edges adjacent to them, therefore we can state the following result.

Theorem 2.2 (Fleiner et al. [42])**.** *The problem of finding a stable matching in an* SR *instance with forced and forbidden edges or reporting that none exists is solvable in* $\mathcal{O}(|E|)$ *time.*

As we have seen so far, answering the question as to whether a stable solution containing all forced and avoiding all forbidden edges exists can be solved efficiently in the case of both SM and SR. We thus concentrate on cases where the answer to this question is "no". What kind of approximate solutions exist then and how can we find them?

	SM	SR
case BP: min # blocking edges	NP-hard to approximate within $\vert V\vert^{1-\varepsilon}$	NP-hard to approximate within $\vert V\vert^{1-\varepsilon}$
case CV: min # violated restricted edge constraints	solvable in polynomial time	NP-hard; 2-approximable if $\vert Q\vert$ is large or 0

Table 2.1: Summary of results.

Our contribution and structure. Since optimality is defined by two criteria, it is straightforward to define approximation from those two points of view. In *case BP*, all constraints on restricted edges must be satisfied, and we seek a matching with the minimum number of blocking edges. In *case CV*, we seek a stable matching that violates the fewest constraints on restricted edges. The optimization problems that arise from each of these cases are defined formally in Section 2.2.

In Section 2.3, we consider case BP: that is, all constraints on restricted edges must be fulfilled, while the number of blocking edges is minimized. We show that in the SM case, this problem is computationally hard and not approximable within $|V|^{1-\varepsilon}$ for any $\varepsilon > 0$, unless P = NP. We also discuss special cases for which this problem becomes tractable. This occurs if the maximum degree of the graph is at most 2 or if the number of blocking edges in the optimal solution is a constant. We point out a striking difference in the complexity of the two cases with only forbidden and only forced edges: the problem is polynomially solvable if the number of forbidden edges is a constant, but by contrast it is NP-hard even if the instance contains a single forced edge. We also prove that when the restricted edges are either all forced or all forbidden, the optimization problem remains NP-hard even on very sparse instances, where the maximum degree of a vertex is 3.

Case CV, where the number of violated constraints on restricted edges is minimized while stability is preserved, is studied in Section 2.4. It is a rather straightforward observation that in SM, the setting can be modeled and efficiently solved with the help of edge weights. Here we show that on non-bipartite graphs, the problem becomes NP-hard, but 2-approximable if the number of forced edges is sufficiently large or zero. As in case BP, we also discuss the complexity of degree-constrained restrictions and establish that the NP-hardness results remain intact even for graphs with degree at most 3, while the case with degree at most 2 is polynomially solvable.

A structured overview of our results is contained in Table 2.1.

2.2 Preliminaries

In this section, we introduce the notation used in the remainder of the chapter and also define the key problems that we investigate later.

As already mentioned in Chapter 1.1, an SR instance need not admit a stable solution. The number of blocking edges is a characteristic property of every matching. The set of edges blocking M is denoted by $bp(M)$. A natural goal is to find a matching mini-

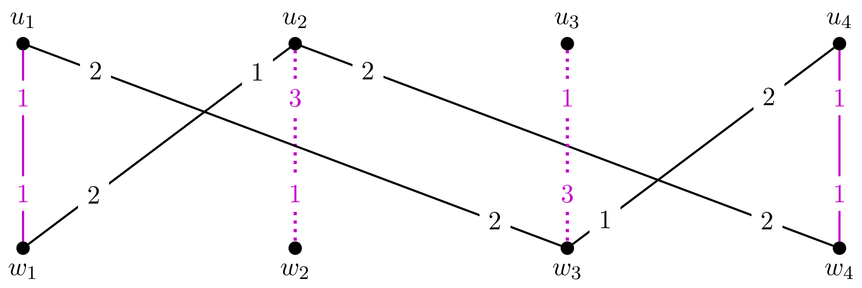


Figure 2.1: An example stable marriage instance with forbidden edges.

mizing $|bp(M)|$. For convenience, the minimum number of edges blocking any matching of an instance $\mathcal{I}$ is denoted by $bp(\mathcal{I})$. Following the consensus in the literature, matchings blocked by $bp(\mathcal{I})$ edges are called *almost stable matchings*. This approach has a broad literature: almost stable matchings have been investigated in SM [17, 50, 65] and SR [1, 16] instances.

All problems investigated in this chapter deal with at least one set of restricted edges. The set of forbidden edges is denoted by P, while Q stands for the set of forced edges. We assume throughout the chapter that $P \cap Q = \emptyset$. A matching M satisfies all constraints on restricted edges if $M \cap P = \emptyset$ and $M \cap Q = Q$.

In Figure 2.1, an example SM instance on four men and four women can be seen. The preference ordering is shown on the edges. The set of forbidden edges $P = \{u_2w_2, u_3w_3\}$ is marked by dotted colored edges. The unique stable matching $M = \{u_1w_1, u_2w_2, u_3w_3, u_4w_4\}$ contains both forbidden edges. Later on, we will return to this example instance to demonstrate approximation concepts on it.

The first approximation concept (case BP described in Section 2.1) is to seek a matching M that satisfies all constraints on restricted edges, but among these matchings, it admits the minimum number of blocking edges. This leads to the following problem definition.

Problem 3. MIN BP SR RESTRICTED
Input: $\mathcal{I} = (G, O, P, Q)$; *an* SR *instance, a set of forbidden edges P and a set of forced edges Q.*
Output: A matching M such that $M \cap P = \emptyset$, $Q \subseteq M$ and $|bp(M)| \leq |bp(M')|$ for every matching M' in G satisfying $M' \cap P = \emptyset$, $Q \subseteq M'$.

Special attention is given to two special cases of MIN BP SR RESTRICTED: in MIN BP SR FORBIDDEN, $Q = \emptyset$, while in MIN BP SR FORCED, $P = \emptyset$. Note that an instance of MIN BP SR FORCED or MIN BP SR RESTRICTED can always be transformed into an instance of MIN BP SR FORBIDDEN by forbidding all edges that are adjacent to a forced edge. This transformation does not affect the number of blocking edges.

According to the other intuitive approximation concept (case CV described in Section 2.1), stability constraints need to be fulfilled, while some of the constraints on restricted edges are relaxed. The goal is to find a stable matching that violates as few constraints on restricted edges as possible.

Problem 4. SR MIN RESTRICTED VIOLATIONS
Input: *$\mathcal{I} = (G, O, P, Q)$; an* SR *instance, a set of forbidden edges P and a set of forced edges Q.*
Output: *A stable matching M such that $|M \cap P| + |Q \setminus M| \leq |M' \cap P| + |Q \setminus M'|$ for every stable matching M' in G.*

Just as in the previous approximation concept (referred to as case BP in Section 2.1), we separate the two subcases with only forbidden and only forced edges. If $Q = \emptyset$, SR MIN RESTRICTED VIOLATIONS is referred as SR MIN FORBIDDEN, while if $P = \emptyset$, the problem becomes SR MAX FORCED. In case BP, the subcase with only forced edges can be transformed into the other subcase, simply by forbidding edges adjacent to forced edges. This straightforward transformation is not valid for case CV. Suppose a forced edge was replaced by an unrestricted edge, but all of its adjacent edges were forbidden. A solution that does not contain the original forbidden edge might contain two of the forbidden edges, violating more constraints than the original solution. Yet most of our proofs are presented in detail for the problem with only forbidden edges, and they require only slight modifications for the case with forced edges.

A powerful tool used in several proofs in this chapter is to convert some of these problems into a weighted SM or SR problem, where the goal is to find a stable matching with the lowest edge weight, taken over all stable matchings. Irving et al. [54] were the first to show that the weighted SM can be solved in $\mathcal{O}(|V|^4 \log |V|)$ time if the weight function is monotone in the preference ordering, non-negative and integral. Feder [38, 39] shows a method to drop the monotonicity requirement. He also presents the best known bound for the running time of an algorithm for finding a minimum weight stable matching in SM: $\mathcal{O}(|V|^2 \cdot \log(\frac{K}{|V|^2} + 2) \cdot \min\{|V|, \sqrt{K}\})$, where K is the weight of an optimal solution. Redesigning the weight function to avoid the monotonicity requirement using Feder's method can radically increase K. For weighted SR, finding an optimal matching is NP-hard, but 2-approximable, under the assumption of monotone, non-negative and integral weights [38]. These constraints restrict the practical use of Feder's results to a large extent. Fortunately, linear programming techniques allow the majority of the conditions to be dropped while retaining polynomial-time solvability. A simple and elegant formulation of the SM polytope is known [93] and using this, a minimum weight stable matching can be computed in polynomial time via linear programming. For weighted SR, a 2-approximation can be found for every non-negative weight function [99, 100].

Throughout this chapter, the discussed SM and SR problems are defined as optimization problems. Not defining them as decision problems is for the sake of approximation results. Every time we work with the decision version of the problem – which occurs often in hardness proofs – we explicitly say so.

2.3 Almost stable matchings with restricted edges

In this section, constraints on restricted edges must be fulfilled strictly, while the number of blocking edges is minimized. Our results are presented in three subsections, and most of the results are given for MIN BP SM RESTRICTED. Firstly, in Section 2.3.1, basic complexity results are discussed. In particular, we prove that the studied problem MIN BP SM RESTRICTED is in general NP-hard and also hard to approximate. Thus, restricted cases are analyzed in Section 2.3.2. First we assume that the number of forbidden, forced or blocking edges can be considered as a constant. Due to this assumption, two of the three problems that naturally follow from imposing these restrictions become tractable, but surprisingly, not all of them. Then, degree-constrained cases are discussed. We show that the NP-hardness result for MIN BP SM RESTRICTED holds even for instances where each preference list is of length at most 3, while on graphs with maximum degree 2, the problems become tractable. Finally, in Section 2.3.3 we mention the problem MIN BP SR RESTRICTED and briefly elaborate on how results established for the bipartite case carry over to the SR case.

2.3.1 General complexity and approximability results

When minimizing the number of blocking edges, one might think that removing the forbidden edges temporarily and then searching for a stable solution in the remaining instance leads to an optimal solution. Such a matching can only be blocked by forbidden edges, but as the upcoming example demonstrates, optimal solutions are sometimes blocked by unrestricted edges exclusively. In some instances, every almost stable solution admits only non-forbidden blocking edges. Moreover, a man- or woman-optimal almost stable matching with forbidden edges may not always exist.

Let us recall the SM instance in Figure 2.1. In the graph with edge set $E(G) \setminus P$, a unique stable matching exists: $M = \{u_1w_1, u_4w_4\}$. However, in the original instance M is blocked by both forbidden edges. On the other hand, matching $M_1 = \{u_1w_1, u_2w_4, u_4w_3\}$ is blocked by exactly one edge: $bp(M_1) = \{u_4w_4\}$. Similarly, matching $M_2 = \{u_1w_3, u_2w_1, u_4w_4\}$ is blocked only by u_1w_1. Therefore, M_1 and M_2 are both almost stable matchings and $bp(\mathcal{I}) = 1$. One can easily check that M_1 and M_2 are the only matchings with the minimum number of blocking edges. They both are blocked only by unrestricted edges. Moreover, M_1 is better for u_1, w_1 and w_3, whereas M_2 is preferred by u_2, u_4 and w_4.

In Theorems 2.3 and 2.4 we present two results demonstrating the NP-hardness and inapproximability of restricted variants of MIN BP SM RESTRICTED.

Theorem 2.3. MIN BP SM FORBIDDEN *and* MIN BP SM FORCED *are* NP*-hard.*

The NP-hard problem we reduce to MIN BP SM RESTRICTED is perfect almost stable matching with incomplete preference lists:

Problem 5. MIN BP PSMI
Input: $\mathcal{I} = (G, O)$*; an* SM *instance on an incomplete bipartite graph.*
Output: A perfect matching M *such that* $|bp(M)| \leq |bp(M')|$ *for every perfect matching* M'*.*

MIN BP PSMI is NP-hard and unless P = NP, it is not approximable within a factor of $|V|^{1-\varepsilon}$, for any $\varepsilon > 0$ [17].

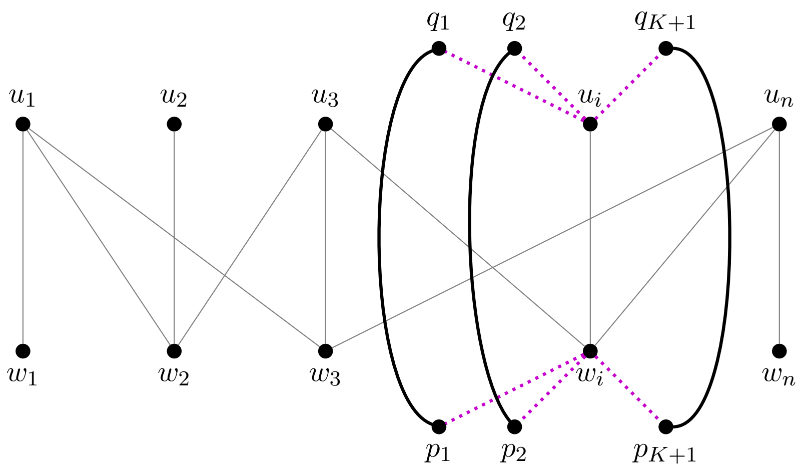


Figure 2.2: The transformation from MIN BP PSMI to MIN BP SM FORBIDDEN. The edges in P are colored and dotted.

Proof. We firstly show NP-hardness of MIN BP SM FORBIDDEN and then indicate how to adapt the proof to show a similar result for MIN BP SM FORCED. We reduce from MIN BP PSMI as mentioned above. Given an instance $\mathcal{I} = (G, O, K)$ of the decision version of this problem, we define the following instance $\mathcal{I}' = (G', O', P, K)$ of the decision version of MIN BP SM FORBIDDEN. The vertices of graph G: $u_i, 1 \leq i \leq n$ and $w_i, 1 \leq i \leq n$ form a subset of $V(G')$. In addition, $K + 1$ new vertices representing women are introduced. They are denoted by $q_1, q_2, ...q_{K+1}$. Similarly, $K+1$ new men are added to $V(G)$, denoted by $p_1, p_2, ...p_{K+1}$. Thus, each side of G' consists of $n + K + 1$ vertices. Edges form a complete bipartite graph on them. The edges added to a sample edge $u_i w_i$ can be seen in Figure 2.2.

The preference lists of vertices already in $V(G)$ are structured in three blocks. Each man u_i of the original instance $\mathcal{I}$ keeps his preference list in O at the top of his new list in O'. After these vertices in W, the entire set of newly-introduced $q_1, q_2, ...q_{K+1}$ women follows, in arbitrary order. Finally, the rest of the women are listed. The ordering within this last block is also arbitrary. An analogous ordering is used when defining the preference list of each w_j. The original list in O is followed by the vertices $p_1, p_2, ...p_{K+1}$ added to G, then the rest of the men follow.

u_i:	w listed in O	$q_1, q_2, ...q_{K+1}$	rest
w_j:	u listed in O	$p_1, p_2, ...p_{K+1}$	rest
p_k :	$w_1, w_2, ..., w_n$	q_k	rest
q_k :	$u_1, u_2, ..., u_n$	p_k	rest

The added vertices have the following preference orderings. Man p_k's list consists of the set of w_j vertices from $V(G)$, followed by p_k, and then the rest of the women in G'

in arbitrary order. Similarly, q_k ranks all $u_i \in V(G)$ first, followed by q_k, and then the rest of the men in arbitrary order.

Having described G' and O' completely, all that remains is to specify the set of forbidden edges P. Each man u_i has $K+1$ forbidden edges adjacent to him, namely, all edges to the newly-introduced $q_1, q_2, ...q_{K+1}$ vertices. Similarly, edges between every w_j and all $p_1, p_2, ...p_{K+1}$ vertices are also forbidden. In total, $\mathcal{I}'$ has $2n(K+1)$ forbidden edges.

Claim 1. *If M is a perfect matching in $\mathcal{I}$ and $|bp(M)| \leq K$, then there is a matching M' in $\mathcal{I}'$ with $M' \cap P = \emptyset$ and $|bp(M')| \leq K$.*

Proof. The construction of M' begins with copying M to G'. Since M is a perfect matching, all vertices in $V(G)$ are matched to vertices in $V(G)$ and thus, no forbidden edge can be in M'. The remaining vertices $q_1, q_2, ...q_{K+1}$ and $p_1, p_2, ...p_{K+1}$ are paired to each other: each $q_j p_j$ is added to M'.

M' is a perfect matching in G' and $M' \cap P = \emptyset$. Next, we show that no edge in $E(G') \setminus M'$ blocks M' that did not block M already. First of all, none of the forbidden edges blocks M', because the preference lists of the vertices already in $V(G)$ were constructed in such a way that the vertices on preference lists in O are better than the added vertices and all u_i, w_j vertices were matched in the perfect matching M. The first n choices of any newly-added vertex are thus not blocking edges. At the same time, all these new vertices are matched to their first-choice partners among the newly-added vertices. Therefore no edge incident to them can block M'. All that remains is to observe that $u_i w_j$ edges blocking M' in $\mathcal{I}'$ already blocked M in $\mathcal{I}$, because M is the restriction of M' to G. Therefore, the edges blocking M and M' are identical. ■

Claim 2. *If M' is a matching in $\mathcal{I}'$ with $M' \cap P = \emptyset$ and $|bp(M')| \leq K$, then its restriction to G is a perfect matching M in $\mathcal{I}$ with $|bp(M)| \leq K$.*

Proof. First, we discuss some essential structural properties of M'. The forbidden edges are not in M', and at most K of them block it. Suppose that there is a man u_i not married to any woman w_j in graph G. Since w_j ranks exactly $K+1$ forbidden edges after its listed partners in G, and forbidden edges are the first n choices of their other end vertex, all $K+1$ of them block M', regardless of the remaining edges in M'. Having derived a contradiction in our assumption that at most K edges block M' in total, we can state that each man u_i is matched to a vertex w_j in M'. Thus, the restriction of M' to G is a perfect matching with at most K blocking edges. ■

NP-hardness can be obtained for MIN BP SM FORCED by simply forcing all edges of the form $p_k q_k$ in the above reduction. □

Theorem 2.4. *Each of* MIN BP SM FORBIDDEN *and* MIN BP SM FORCED *is not approximable within a factor of $|V|^{1-\varepsilon}$, for any $\varepsilon > 0$, unless* P = NP.

The NP-complete problem we make use of in this proof is EXACT MAXIMAL MATCHING.

Problem 6. EXACT MAXIMAL MATCHING
Input: $\mathcal{I} = (G, K)$; a bipartite graph G and an integer K.
Question: Is there a maximal matching M in G such that $|M| = K$?

EXACT MAXIMAL MATCHING is NP-complete even for graphs where all vertices representing men have degree two, while all vertices of the other side have degree three [81]. Here we also use this restricted case of the problem. We show that if there were a polynomial approximation algorithm within a factor of $|V|^{1-\varepsilon}$ for some $\epsilon > 0$ to MIN BP SM FORBIDDEN, then it would also find an exact maximal matching in $\mathcal{I}$.

Proof. In our proof, every degree-restricted instance $\mathcal{I}$ of EXACT MAXIMAL MATCHING is transformed into a corresponding instance $\mathcal{I}''$ of MIN BP SM FORBIDDEN. Let n_1 and n_2 denote the size of each side in $\mathcal{I}$, such that $m = 2n_1 = 3n_2$. Furthermore, another transformation is used, involving $\mathcal{I}'$, an instance of perfect matching with incomplete preference lists (Problem 5). In [17], an instance $\mathcal{I}'$ of MIN BP PSMI is created corresponding to each $\mathcal{I}$ of EXACT MAXIMAL MATCHING with some special properties. The crucial one of them is that if G in $\mathcal{I}$ has a maximal matching of cardinality K, then $\mathcal{I}'$ has a perfect matching admitting exactly $n_1 + n_2$ blocking edges, where $n_1 + n_2$ is the number of vertices in $\mathcal{I}$. Otherwise, if G has no maximal matching of cardinality K, then any perfect matching in $\mathcal{I}'$ is blocked by at least $n_1 + n_2 + C$ edges, where C is a huge number. To be more precise, let $B = \lceil \frac{3}{\varepsilon} \rceil$ and $C = (n_1 + n_2)^{B+1} + 1$. The number of vertices in each side of $\mathcal{I}'$ is $3n_1 + 2mC + 4n_2 - K$.

Now we describe how $\mathcal{I}'$ is transformed into $\mathcal{I}''$. Note that this method is very similar to the one we used in the proof of Theorem 2.3. First, C new men: p_i, $1 \le i \le C$ and C new women: q_i, $1 \le i \le C$ are introduced. Therefore, each side consists of $3n_1 + 2mC + 4n_2 - K + C$ vertices. The preference lists can be sketched in the following way:

u in $\mathcal{I}'$:	w listed in O in $\mathcal{I}'$	$q_1, q_2, ..., q_C$	all remaining women in $\mathcal{I}'$
w in $\mathcal{I}'$:	u listed in O in $\mathcal{I}'$	$p_1, p_2, ..., p_C$	all remaining men in $\mathcal{I}'$
p_i:	all women in $\mathcal{I}'$	q_i	$q_1, q_2, ..., q_{i-1}, q_{i+1}, ..., q_C$
q_i:	all men in $\mathcal{I}'$	p_i	$p_1, p_2, ..., p_{i-1}, p_{i+1}, ..., p_C$

The set of forbidden edges comprises all edges of the form pw or uq. For MIN BP SM FORCED, the set of forced edges consists of all edges of the form $p_i q_i$ (with identical indexes). Due to this construction, if M is a matching in $\mathcal{I}''$ in which there is a man u_i not matched to a woman he is adjacent to in $\mathcal{I}'$, then M is blocked by at least C edges.

We will show that if N is the number of vertices in $\mathcal{I}''$, then $N^{1-\varepsilon} < C$. Therefore, any $N^{1-\varepsilon}$-approximation of MIN BP SM FORBIDDEN guarantees a matching in $\mathcal{I}'$ admitting less than C blocking edges. This latter induces a perfect matching in $\mathcal{I}'$, which corresponds to a solution of EXACT MAXIMAL MATCHING.

With Inequalities (2.1)-(2.7) we give an upper bound for N, while with Inequalities (2.8)-(2.12) we establish a lower bound. Then, combining these two in Inequalities (2.13)-(2.16), we derive that $N^{1-\varepsilon} < C$. Explanations for the steps are given as necessary after each of the three sets of inequalities.

$$\begin{aligned}
N &= 2(3n_1 + 2mC + 4n_2 - K + C) && (2.1)\\
&= 6n_1 + 8n_1C + 8n_2 - 2K + 2C && (2.2)\\
&\leq 6n_1 + 8n_1((n_1+n_2)^{B+1} + 1) + 8n_2 - 2K + 2(n_1+n_2)^{B+1} + 2 && (2.3)\\
&= 14n_1 + (n_1+n_2)^{B+1}(8n_1 + 2) + 8n_2 + 2 && (2.4)\\
&\leq 14n_1 + 14n_2 + (n_1+n_2)^{B+1}(14n_1 + 14n_2) && (2.5)\\
&\leq 14[(n_1+n_2)^{B+1}(n_1+n_2) + (n_1+n_2)^{B+1}(n_1+n_2)] && (2.6)\\
&= 28(n_1+n_2)^{B+2} && (2.7)
\end{aligned}$$

(2.1): N is the number of vertices in $\mathcal{I}''$
(2.2): $m = 2n_1$
(2.3): $C = (n_1+n_2)^{B+1} + 1$ by definition
(2.4): omit $-2K$
(2.5): $n_2 \geq 1$, increase all coefficients to the highest coefficient 14
(2.6): multiply $14n_1 + 14n_2$ by $(n_1+n_2)^{B+1}$

$$\begin{aligned}
N &= 6n_1 + 8n_1C + 9n_2 - 2K + 2C && (2.8)\\
&> C && (2.9)\\
&> (n_1+n_2)^{B+1} && (2.10)\\
&> n_1^B && (2.11)\\
&\geq 28^B && (2.12)
\end{aligned}$$

(2.8): N is the number of vertices in $\mathcal{I}''$
(2.9): keep only C from the sum
(2.10): $C = (n_1+n_2)^{B+1} + 1$ by definition
(2.11): keep only n_1^B from the sum coefficient 14
(2.12): without loss of generality, we can assume that $n_1 > 28$

$$\begin{aligned}
C &> (n_1+n_2)^B && (2.13)\\
&\geq 28^{-\frac{B}{B+2}} N^{\frac{B}{B+2}} && (2.14)\\
&\geq N^{1-\frac{3}{B+2}} && (2.15)\\
&\geq N^{1-\varepsilon} && (2.16)
\end{aligned}$$

(2.13): $C = (n_1+n_2)^{B+1} + 1$ by definition
(2.14): (2.1)-(2.7)
(2.15): (2.8)-(2.12)
(2.16): $B = \lceil \frac{3}{\varepsilon} \rceil$ by definition □

2.3.2 Bounded parameters

Our results presented so far show that MIN BP SM RESTRICTED is computationally hard even if $P = \emptyset$ or $Q = \emptyset$. Yet if certain parameters of the instance or the solution can be considered as a constant, the problem can be solved in polynomial time. Theorem 2.5 firstly shows that this is true for MIN BP SM FORBIDDEN.

Theorem 2.5. MIN BP SM FORBIDDEN *is solvable in* $\mathcal{O}(|V|^2|E|^L)$ *time, where* $L = |P|$, *which is polynomial if* L *is a constant.*

Proof. Here we work with the decision version of MIN BP SM FORBIDDEN, asking whether there is a matching M such that $M \cap P = \emptyset$ and $|bp(M)| \leq K$ for a fixed $K \in \mathbb{Z}_{>0}$. Our first observation is that the problem is trivially solvable if the target value K satisfies $K \geq L$. In this case, deleting the L forbidden edges from $E(G)$ and finding a stable matching in the remaining graph delivers a matching that is blocked in the original instance by only a subset of the removed edges. Thus, a matching M with $M \cap P = \emptyset$ and $|bp(M)| \leq K$ always exists.

Otherwise, we assume that $K < L$. Suppose firstly that there is a matching M with $M \cap P = \emptyset$ and $|bp(M)| = k \leq K < L$. If those k blocking edges are deleted from $E(G)$, then there is a stable matching M' in the remainder of G that contains none of the forbidden edges. Note that we did not specify which edges block M: they can be both forbidden and unrestricted. Due to Theorem 2.1, deciding whether a stable matching with no forbidden edges exists is polynomially solvable. The last task is to check each possible set of blocking edges. Every edge set of size at most K is such a potential blocking set. During the execution of our algorithm, we try out all of these sets one by one. After such an edge set is deleted from G, a stable matching that avoids all of the remaining forbidden edges is searched for. If such a matching exists, then it admits at most K blocking edges. It is sufficient to try out $\sum_{i=0}^{K} \binom{m}{i}$ sets of edges. In other words, $\sum_{i=0}^{K} \binom{m}{i} \leq \sum_{i=0}^{L} \binom{m}{i}$ subsets are generated to decide whether there is a matching that does not contain any of the L forbidden edges and admits at most K blocking edges. The number of rounds is thus at most $\mathcal{O}(|E|^L)$, while each round takes $\mathcal{O}(|V|^2)$ time to complete. □

In sharp contrast to the previous result on polynomial solvability when the number of forbidden edges is small, we state the following theorem for MIN BP SM FORCED.

Theorem 2.6. MIN BP SM FORCED *is* NP*-hard even if* $|Q| = 1$.

Proof. The NP-complete problem we reduce to the decision version of MIN BP SM FORCED is EXACT MAXIMAL MATCHING. As previously mentioned, this problem is NP-complete even for graphs where all vertices representing men have degree two, while all vertices of the other side have degree three [81]. Hence suppose we are given an instance $\mathcal{I}$ of this restriction, comprising a graph $G = (U_0 \cup W_0, E)$ and an integer K.

In this proof, we construct a MIN BP SM FORCED instance $\mathcal{I}'$ with a single forced edge in such a way that there is a maximal matching of cardinality K in $\mathcal{I}$ if and only if there

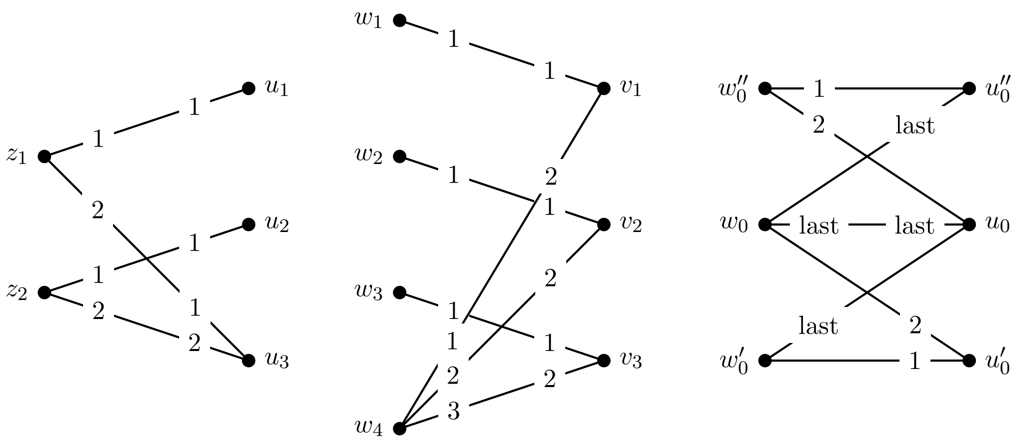


Figure 2.3: A u-gadget, a w-gadget and the special gadget.

is a matching containing the forced edge and admitting exactly $|U_0| + |W_0|$ blocking edges in $\mathcal{I}'$. Our construction is based on ideas presented in [17].

All vertices label their edges in an arbitrary but fixed order. We will refer to these labels when constructing $\mathcal{I}'$. We now describe $\mathcal{I}'$. The vertex set of graph G' in $\mathcal{I}'$ can be partitioned into seven sets: U, V, W, Z, S, X and Y. Specific subgraphs of G' are referred as u-gadgets, w-gadgets, a special gadget containing the forced edge, see Figure 2.3. Aside from these, G' also contains some extra vertices, the so-called *garbage collectors*, partitioned into two sets: X and Y. Later we will see that these garbage collectors are paired to the vertices not covered by the matching in G. To that end, $|X| = |W_0| - K$ and $|Y| = |U_0| - K$. The whole construction is illustrated in Figure 2.4.

Each u-gadget replaces a vertex $u \in U_0$ in G. It is defined on five vertices: $u_1, u_2, u_3 \in U$ and $z_1, z_2 \in Z$. Its edges and the preferences on them are shown in Figure 2.3. Two *interconnecting edges* connect the special gadget to u_3. They are ranked as the last two edges by u_3. It is described later which vertices of the special gadget are incident to these edges of u_3. The u-gadget also has edges to all w-gadgets representing vertices in W_0 to which u was adjacent. After describing the w-gadget, we elaborate on the position of these edges referred as *relevant edges*. Aside from these, every u_1 has edges to all garbage collectors in Y. These edges are all worse than the relevant edges of u_1 and they are ranked arbitrarily at the bottom of u_1's list. The vertices in Y also rank all u_1 vertices arbitrarily.

The w-gadgets are structured similarly. Each gadget consists of seven vertices: $w_1, w_2, w_3, w_4 \in W$ and $v_1, v_2, v_3 \in V$. Aside from the edges within the gadget, it has two interconnecting edges between w_4 and vertices in the special gadget (described in detail later), and three relevant edges between w_1, w_2, w_3 and vertices in U of u-gadgets. These are the edges drawn in accordance with the edge labels. Suppose edge uw was ranked i-th by u and j-th by w, where $i \in \{1, 2\}$ and $j \in \{1, 2, 3\}$. Then, u_i in the u-gadget is connected to w_j in the w-gadget. Therefore, each edge in $\mathcal{I}$ is transformed into a single edge in $\mathcal{I}'$ and each u_i, $i \in \{1, 2\}$ and w_j, $j \in \{1, 2, 3\}$, has exactly one relevant edge. All of these edges are second choices of both of their end vertices. In addition to these, if a u- and a w-gadget, for which $uv \in E(G)$ are not already connected by u_1w_1, we add u_1w_1, which is referred as an *adjacency edge*. This edge is ranked by both u_1 and w_1 after their relevant edges, but ahead of their edges to garbage collectors. Similar to u-gadgets, w-gadgets are also connected to garbage collectors. Each w_1 vertex has $|W_0| - K$ edges to the vertices in X, ranked arbitrarily at the bottom of w_1's preference list. Also the vertices in X rank the w_1 vertices arbitrarily.

The special gadget is defined on only six vertices forming set S: $u_0, u_0', u_0'', w_0, w_0'$ and w_0''. The unique forced edge in the entire instance is u_0w_0. Apart from u_0' and w_0'', they are connected to u- and w-gadgets. In each u-gadget, u_3 is adjacent to w_0 and w_0', while in each w-gadget, w_4 is adjacent to u_0 and u_0''. These four vertices prefer their interconnecting edges to their edges inside of the special gadget. Moreover, u_0 and w_0 are connected to all garbage collectors of the opposite side. These edges are ranked better than u_0w_0 by these two vertices and ranked last by the vertices in X and Y.

Claim 3. *To each maximal matching M in $\mathcal{I}$ of cardinality K there is a matching M' in $\mathcal{I}'$ with $u_0w_0 \in M'$ and $|bp(M')| = |U_0| + |W_0|$.*

Proof. First, the set of relevant edges in G' corresponding to M is chosen. They cover exactly K of the $|U| = 3|U_0|$ vertices of U, and analogously, exactly K of the $|W| = 4|W_0|$ vertices in W.

In u-gadgets, where either of u_1 and u_2 has a relevant edge in M', the other vertex in U is matched to its copy in Z. The remaining two vertices of the gadget are then paired to each other. In the other case, if u was unmatched in M, then $u_2z_2, u_3z_1 \in M$, and $M(u_1) \in Y$. Given the set of u_1 vertices to pair with the garbage collectors, we find any stable matching in this subgraph and add it to M'. Note that this step matches the $|U_0| - K$ u_1 vertices to the $|U_0| - K$ garbage collectors in Y.

The strategy is similar for the w-gadgets. Suppose that w_j is already matched to a vertex in U, because that relevant edge corresponds to a matching edge in M. We then connect w_4 to v_j and pair the remaining two vertices in W with their partners in V. Otherwise, if w was unmatched in M, then w_1 is matched to a garbage collector, and $\{w_2v_2, w_3v_3, w_4v_1\} \subseteq M'$. On the subgraph induced by the garbage collectors and w_1 vertices corresponding to unmatched w vertices we construct a stable matching and add it to M'. This step matches the $|W_0| - K$ w_1 vertices to the $|W_0| - K$ garbage collectors in X.

In the special gadget, $u_0w_0, u_0'w_0'$ and $u_0''w_0''$ are chosen.

Now we investigate the number of blocking edges incident to at least one vertex in any u-gadget. The edges running to garbage collectors cannot block, because M' restricted to that subgraph is a stable matching and u_1 vertices not matched to garbage collectors have better relevant edges in M'. Since all u_3 vertices are matched to their first or second choices, their edges to the special gadget do not block either. Consider now a relevant edge u_iw_j. Since M was a maximal matching, at least one of the two gadgets are set so that it corresponds to a matched vertex in M. On that side, u_iw_j is dominated by the matching edge. Regarding the adjacency edges, they only block M' if both of their end vertices are matched to garbage collectors. But they both are then unmatched

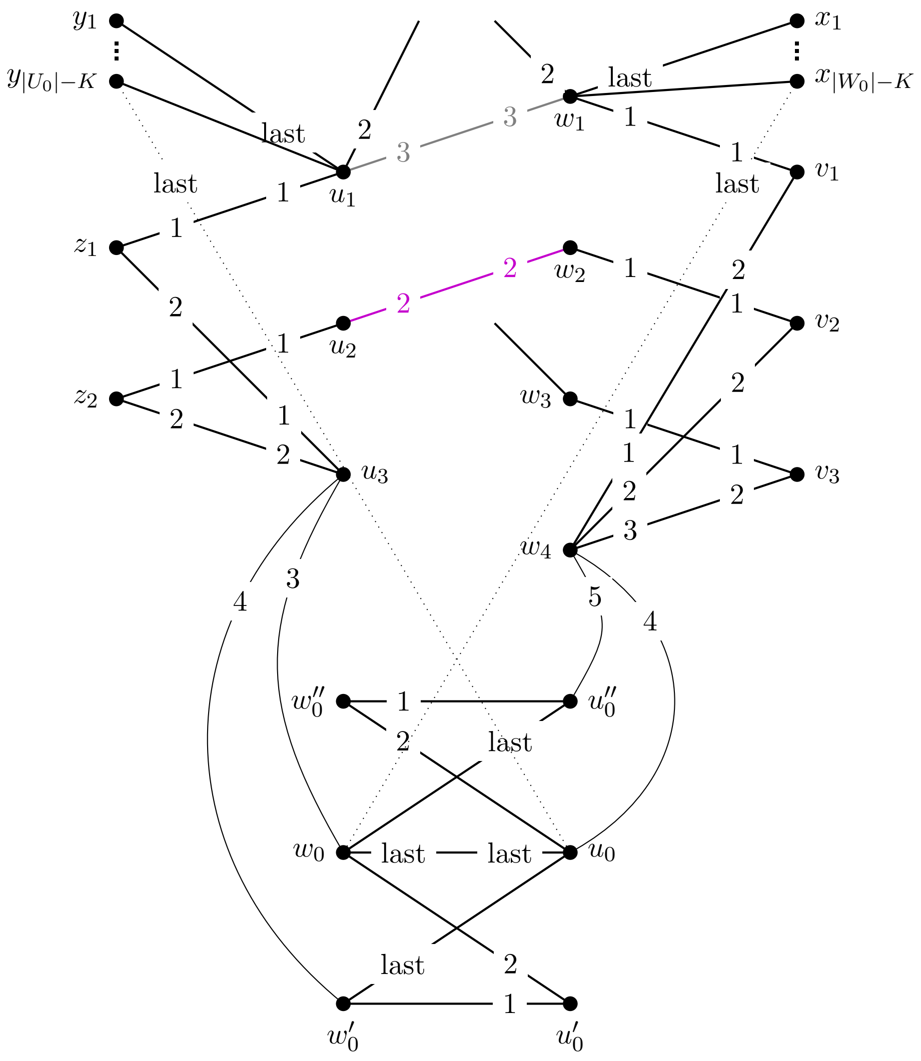


Figure 2.4: As the purple relevant edge shows, u and w were connected in $\mathcal{I}$ by an edge labeled second by both of them. The gray edge is an adjacency edge.

and adjacent in G, which contradicts to the fact that M is maximal. The only edges remaining are in the u-gadgets. In each u-gadget, exactly one edge blocks M': if the vertex was matched to its i-th labeled edge in M, then $u_i z_i$ blocks M', otherwise $u_1 z_1$ blocks M'. Therefore, up to this point, we have exactly $|U_0|$ blocking edges.

Analogous arguments prove that among the edges incident to all w-gadgets, $|W_0|$ are blocking. In the previous paragraph we discussed that no relevant or adjacency edge blocks M'. The subgraph induced by the garbage collectors and w_1 vertices does not contain any blocking edge, because a stable matching was chosen and the unmatched w_1 vertices are all matched to a better vertex. Edges connecting w_4 vertices and the special gadget are last choice edges of the matched w_4 vertices. In the w-gadget, exactly one edge blocks M': if w was matched and therefore $u_i w_j \in M'$, then $w_j v_j$, otherwise $w_1 v_1$.

It is easy to see that in the special gadget, none of the four non-matching edges blocks M'. ■

Claim 4. *To each matching M' in $\mathcal{I}'$ with $u_0 w_0 \in M'$ and $|bp(M')| = |U_0| + |W_0|$ there is a maximal matching M in $\mathcal{I}$ of cardinality K.*

Proof. First we show that if $u_0 w_0 \in M'$, then each u- and w-gadget is adjacent to at least one blocking edge. Since w_0 prefers all its edges to $u_0 w_0$, if a u-gadget is not incident to any blocking edge, then u_3 is matched to its first or second choice edge. But either of z_1 or z_2 is then not matched to its first-choice edge, which is the best edge of its other end vertex as well. Since such edges block the matching immediately, we have found at least one blocking edge incident to the vertices of the u-gadget. The same argument applies to u_0 and w_4. If $M(w_4) \in V$, then $M(w_4)$ has a blocking edge, otherwise $u_0 w_4$ blocks M'. Therefore, if $|bp(M')| \leq |U_0| + |W_0|$, then each u- and w-gadget is adjacent to exactly one blocking edge.

In the coming two paragraphs, we investigate which edges can play the role of blocking edges. Trivial candidates are the edges $u_i z_i$, where $i \in \{1, 2\}$ and $w_j v_j$, where $j \in \{1, 2, 3\}$, because they are the first-choice edges of both of their end vertices. Suppose $u_1 z_1 \notin M'$, therefore it blocks M'. Then, $u_2 z_2 \in M'$, otherwise it also blocks M'. Now we know that $u_3 z_1 \in M'$, otherwise $u_3 z_1$ forms a blocking edge. This construction guarantees that the edges incident to vertices in the u-gadget are all dominated by matching edges, except for the edges of u_1. The second option is that $u_2 z_2 \notin M'$. Similarly, $u_1 z_1 \in M'$ and $u_3 z_2 \in M'$, moreover, the only edges incident to the gadget that might block M' are the edges of u_2. The remaining case is that $u_1 z_1, u_2 z_2 \in M'$. In this case, u_3 is either unmatched or matched to w_0' and in both cases $u_3 w_0$ blocks M'. In all three cases, no edge incident to the gadget at u_1 or u_2 can block M', because that would mean more than one blocking edge per gadget.

An analogous reasoning can be derived for w-gadgets. If $w_j v_j \notin M'$ for a $j \in \{1, 2, 3\}$, then $w_{j-1} v_{j-1}, w_{j+1} v_{j+1} \in M'$, where addition and subtraction are taken modulo 3. In this case, $w_j v_j$ blocks M'. In the very last case, if $w_j v_j \in M'$ for all $j \in \{1, 2, 3\}$, then $w_4 u_0$ forms a blocking edge. For all four matchings, no further blocking edge can occur.

At this point, we have proved that each u-gadget and each w-gadget has exactly one blocking edge and blocking edges never appear anywhere else in the graph. In particular, the special gadget contains no blocking edge. Moreover, if $u_0' w_0' \notin M'$, then $u_0' w_0$ blocks M'. Therefore, $u_0' w_0' \in M'$. This observation has an effect on the u-gadgets. If any vertex u_3 would be unmatched in M', then it would immediately have two blocking edges, one to w_0 and another one to w_0'. Therefore, u_3 is matched to z_1 or z_2 and either u_1 or u_2 is matched to a vertex outside of the u-gadget. From this follows the fact that the number of relevant edges in M' plus the number of matching edges incident to garbage collectors in Y is $|U_0|$. Similarly, if $u_0'' w_0'' \notin M'$, then $u_0 w_0''$ blocks M'. Thus, all w_4 vertices are matched to a vertex in V and the number of relevant edges in M' plus the number of matching edges incident to garbage collectors in X is $|W_0|$.

Suppose now that there is an unmatched vertex $y \in Y$. This vertex has an edge to u_0 which blocks M'. Therefore, all $|U_0| - K$ vertices in Y are matched to u_1 vertices of various u-gadgets. Similarly, all $|W_0| - K$ vertices in X are matched to w_1 vertices. Therefore, there are in total K u-gadgets contributing a single relevant edge to M'. These edges therefore form a matching of size K in G. All that remains to show is that this matching is maximal. If we suppose otherwise, i.e., there are two gadgets corresponding to vertices u and w in G such that all their vertices in U and W in G' are matched to either garbage collectors or to their z- or v-copies. This is only possible if u_1 and w_1 are both matched to garbage collectors, but the adjacency edge $u_1w_1$then blocks M'. ■ □

A counterpart to Theorem 2.5 holds in the case of MIN BP SM RESTRICTED if the number of blocking pairs in an optimal solution is a constant.

Theorem 2.7. MIN BP SM RESTRICTED *is solvable in* $\mathcal{O}(|E|^{L+1})$ *time, where* L *is the minimum number of edges blocking an optimal solution, which is polynomial if* L *is a constant.*

For the sake of simplicity, we use the decision version of MIN BP SM RESTRICTED, asking whether there is a matching M such that $M \cap P = \emptyset$, $Q \subseteq M$ and $|bp(M)| \leq K$ for $K \in \mathbb{Z}_{>0}$ fixed in the input.

Proof. Let us consider all edge sets of cardinality L. Removing one of these sets from G yields a graph G'. If there is a matching in G blocked by a subset of the L removed edges and not violating any of the constraints on restricted edges, then the same matching is stable and still does not violate any constraint on the restricted edges in G'. Therefore, it is sufficient to check whether there is a matching not violating any of the constraints on the restricted edges in G'. There are $\mathcal{O}(|E|^L)$ sets to remove and checking the existence of a stable matching with forced and forbidden edges can be done in $\mathcal{O}(|E|)$ time. □

Next we study the case of degree-constrained graphs, because for most hard SM and SR problems, it is the most common special case to investigate [16, 50, 81]. Here, we show in Theorem 2.8 that MIN BP SM RESTRICTED remains computationally hard even for instances with preference lists of length at most 3. On the other hand, according to Theorem 2.9, the problem can be solved by identifying forbidden subgraphs when the length of preference lists is bounded by 2.

Theorem 2.8. MIN BP SM FORBIDDEN *and* MIN BP SM FORCED *are* NP*-hard even if each vertex has a preference list consisting of at most 3 elements.*

The problem we reduce to our problem is (2,2)-E3-SAT. This satisfiability problem is NP-complete [10].

Problem 7. (2,2)-E3-SAT
Input: $\mathcal{I} = B$*; a Boolean formula in CNF, in which each clause comprises exactly 3 literals and each variable appears exactly twice in unnegated and exactly twice in negated form.*
Question: *Is there a truth assignment satisfying* B*?*

For convenience, let us denote the number of variables by n and the number of clauses by m. Our goal is to construct an instance $\mathcal{I}$ of MIN BP SM FORBIDDEN so that $bp(\mathcal{I}) = m + n$ if and only if B is a satisfiable formula.

Our construction for such an instance combines ideas from two papers. To each Boolean formula, we introduce a variable gadget and a clause gadget. The first one is a slightly more sophisticated variant of the variable gadget used in Theorem 7 of [17], to show NP-hardness of finding a maximum cardinality almost stable matching. Our clause gadget is a simplified version of another clause gadget from Theorem 1 in [16]. There, the almost stable roommates problem is shown to be NP-hard. Both proofs investigate the case with bounded preference lists.

Proof. Using the already described transformation, any MIN BP SM FORCED instance with short preference lists can be converted into a MIN BP SM FORBIDDEN instance with short preference lists. Therefore, it is sufficient to investigate MIN BP SM FORBIDDEN.

The problem we reduce to our problem is (2,2)-E3-SAT. Our goal is to construct an instance $\mathcal{I}$ of MIN BP SM FORBIDDEN so that $bp(\mathcal{I}) = m + n$ if and only if B is a satisfiable formula.

When constructing graph G to a given Boolean formula B, we keep track of the order of the three literals in each clause and the order of the two unnegated and two negated appearances of each variable. Each appearance is represented by an interconnecting edge.

The variable gadget. To each variable in B, a graph on 44 vertices is defined. The right hand-side of Figure 2.5 illustrates the essential part of a variable gadget, a cycle of length 24. This cycle contains no forbidden edge, and all vertices along it have degree 3 due to an additional forbidden edge. For sake of simplicity, only four of these edges are depicted in the figure, namely the ones incident to vertices x_1, x_2, x_3 and x_4, as they are responsible for the communication between clause and variable gadgets. Each of these four vertices has its third, forbidden edge connected to a clause gadget. These edges are called *interconnecting edges* and ranked second on the preference list of both of their end vertices.

Consider the variable v. Due to the properties of MAX (2,2)-E3-SAT, v occurs twice in unnegated form, say, in clauses C_1 and C_2. Its first appearance, as the i-th literal of C_1 is represented by the interconnecting edge between x^1 and a^i of the clause gadget corresponding to C_1. Similarly, x^2 is connected to an a-vertex in the clause gadget of C_2. The same variable, v, also appears twice in negated form. The variable gadgets representing those clauses are connected to x^3 and x^4, respectively. The other end vertices of these two interconnecting edges mark where these two literals appear in their clauses.

As mentioned before, all vertices along the cycle have exactly one forbidden edge attached to them. Regarding the remaining 20 vertices of the cycle, there is a dummy vertex with a forbidden edge attached to each of them (these edges are not depicted in Figure 2.5). This edge is their last choice. These edges guarantee that if any of these 20 vertices remains unmatched in the cycle, then it contributes a blocking edge.

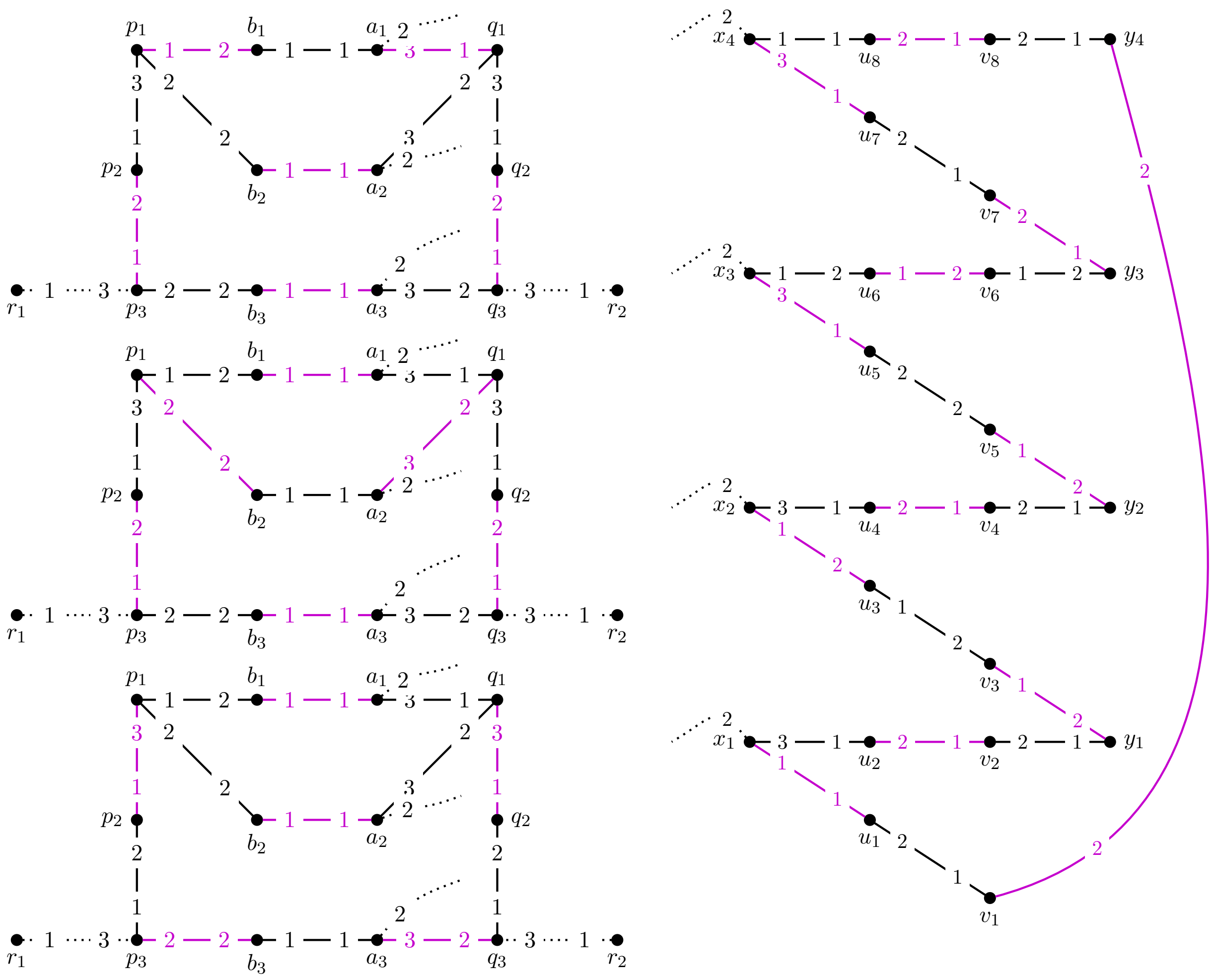


Figure 2.5: A clause and a variable gadget with their special matchings, marked by colors. The dotted edges are forbidden.

Two special matchings are defined on a variable gadget: M_T, denoted by black edges and M_F, comprising the colored edges. While M_T is blocked by x_1u_1 exclusively, M_F is blocked by x_4u_8 exclusively.

Claim 5. *Let M be a matching on a variable gadget. If M is not M_T or M_F, then it is blocked by at least two edges, one of them belonging to the variable gadget.*

Proof. Since x_1u_1 and x_4u_8 are best-choice edges of both of their end vertices, they block any matching not containing them. If both of them are in M, then there is at least one unmatched vertex on the cycle between u_8 and u_1 and another unmatched vertex between x_4 and x_1. The first path comprises vertices with a forbidden edge as their last choice, therefore, it already contributes a blocking edge. The only way to avoid additional blocking edges on the second path is to leave a special vertex unmatched, namely x_2 or x_3. Since they both are first choices of some other vertex along the cycle, edges of the cycle block M. Therefore, x_1u_1 and x_4u_8 cannot be in M simultaneously.

The remaining case is when exactly one of them is in M. If every second edge in the cycle belongs to M, then it is either M_T or M_F. Otherwise, for simple parity reasons, there are two unmatched vertices on the 24-cycle between u_1 and u_8. They already induce two blocking edges, since their last-choice edges are the forbidden edges hanging on them. ■

The clause gadget. To each clause in B, a graph on 14 vertices is defined. Three of them; a_1, a_2 and a_3, are connected to variable gadgets via interconnecting edges, all ranked second. There are three special matchings on a clause gadget, blocked by only a single edge. They can be seen on the left hand-side of Figure 2.5. The colored edges denote M_1, M_2 and M_3, from the top to the bottom.

Claim 6. *Let M be a matching in a clause gadget. If M is not M_1, M_2 or M_3, then it is blocked by at least two edges, one of them belonging to the clause gadget.*

Proof. First, suppose that there is a matching $M \neq M_i$ for all $i \in \{1,2,3\}$ blocked by a single edge. Since all three edges connecting a and b-vertices are first choices of both of their end vertices, they block any matching not including them. Another restriction arises from the fact that the forbidden edges r_1p_3 and r_2q_3 ensure that if p_3 or q_3 are unmatched, they also contribute a blocking edge. Similarly, if p_1 or q_1 is unmatched, they contribute a blocking edge.

Suppose $b_ia_i \in M$ for all $i \in \{1,2,3\}$. Then, p_2 is matched either to p_1 or to p_3, leaving the other one unmatched. The same argument applies for the q-vertices on the other side of the gadget. Therefore, at least two edges block M.

In the remaining case, exactly one of the b_ia_i edges is outside of M. Since we are searching for a matching blocked by only a single edge, no further blocking edge can occur. Therefore, p_1, q_1, p_3 and q_3 are all matched in M. From this point on, it is easy to see that all matchings fulfilling these requirements are M_1, M_2 and M_3. ■

Claims 5 and 6 guarantee that if a matching M's restriction to any of the variable or clause gadgets deviates from their special matchings, then $|bp(M)| > n + m$.

Claim 7. *In the* MIN BP SM FORBIDDEN *instance $\mathcal{I}$, $bp(\mathcal{I}) \leq m + n$ if and only if B is a satisfiable formula.*

Proof. First, we construct a matching M with $|bp(M)| = m + n$ to a given truth assignment. On the variable gadgets, the edges of M_T are chosen if the corresponding variable is `true`, and the edges of M_F otherwise. There is at least one literal in each clause that is true in the truth assignment. If this literal is the i-th in the clause, matching M_i is chosen, where $i \in \{1,2,3\}$. If more than one literal is true, we choose one of them arbitrarily. Due to Claims 5 and 6, each gadget contributes a single blocking edge. As a last step, we show that no interconnection edge blocks M. Suppose that a_ix_j blocks M. Since it is the second choice of a_i, it follows that $b_ia_i \notin M$. We now know that the i-th literal of the clause was `true` in the truth assignment. Therefore, x_i is matched to its first choice.

To prove the opposite direction, we again rely on Claims 5 and 6. On one hand, these two statements prove that $bp(\mathcal{I}) \geq m + n$. On the other hand, $|bp(M)| = m + n$ occurs if and only if M's restriction to variable gadgets are M_T or M_F and its restriction to clause gadgets are M_1, M_2 or M_3. Then, assigning true to all variables with M_T in their gadgets and false to the rest results in a truth assignment. Since no interconnection edge blocks M, at least one literal per clause is true. ■ □

Theorem 2.9. MIN BP SM RESTRICTED *is solvable in* $\mathcal{O}(|V|)$ *time if each preference list consists of at most 2 elements.*

Proof. In this constructive proof we describe an algorithm that produces an optimal matching. First, the input is simplified. Then, the graph is segmented so that each subgraph falls under a category with a specified choice rule for the edges of an optimal matching. As in previous cases, it is sufficient to tackle MIN BP SM FORBIDDEN, because instances of MIN BP SM FORCED can be transformed to this problem.

Due to the degree constraints, every component of the underlying graph is a path or a cycle. If any of these components is free of forbidden edges, then we simply fix a stable matching on it. This step is executed whenever such a component appears during the course of the algorithm. For those components with forbidden edges, we split all vertices having a first-choice forbidden edge and a second edge - unrestricted or forbidden - into two vertices. This change does not affect $|bp(M)|$, because in this case, both edges block the matching if and only if their other end vertex is matched to a worse partner or is unmatched. After this splitting is executed, all components contain forbidden edges that start paths or that are inside a path, being the last choices of both of their end vertices.

Each component consists of *segments* of unrestricted edges, separated by forbidden edges. When talking about a segment, we always mean a series of adjacent unrestricted edges. Since unrestricted cycles have already been eliminated by fixing a stable matching on them, every segment is a path. Due to the Rural Hospitals Theorem, each path admits a unique stable matching. Fixing a matching on a segment induces blocking edges only among the unrestricted edges of the segment and the forbidden edges adjacent to the segment. We claim that in an optimal solution, each segment and the (at most two) forbidden edges surrounding it contribute at most two blocking edges. This is simply due to the fact that any stable solution on the unrestricted edges is blocked only by forbidden edges. Therefore, deviating from this solution might only pay off if the matching restricted to this segment is blocked by a single edge and covers both of its end vertices.

The unique stable matching M on a segment $\langle v_1, v_2, ..., v_k \rangle$ falls into exactly one of the following categories:

1. M covers both v_1 and v_k;
2. M covers either v_1 or v_k;
3. M covers neither v_1 or v_k.

In each step of our algorithm, a segment is chosen and a matching is fixed on it. The segment and some of the forbidden edges adjacent to it is then removed from the graph. This is done in the following way in these three cases.

In case 1, an optimal solution arises from choosing M. If a forbidden edge e is incident to either v_1 or v_k, it cannot block M. Nor can it block any superset of M in the original instance, so e can be deleted. In case 2, again the optimal solution arises from choosing M. Without loss of generality suppose that v_1 is covered. As in case 1, if a forbidden edge is incident to v_1, it cannot block a superset of M in the original instance. Now suppose that a forbidden edge e is incident to v_k. Edge e may block M, and may also block a superset of M in the original instance, so it is retained.

The third case is divided into two subcases, depending on whether there is a matching M' that is blocked by only one edge and covers both v_1 and v_k. Finding such a matching or proving that none exists can be done iteratively, assuming that a chosen edge is the single blocking edge and then constructing M' so that no more edge blocks it. If such an M' does not exist, then M is fixed, and the segment (but not the forbidden edges) is removed. At the end, the matching restricted to this segment will be blocked by no other edge than the two forbidden edges. Suppose that changing the optimal matching to M' increases the number of blocking edges adjacent to this segment. This is only possible if at least one of the forbidden edges becomes blocking. Assume without loss of generality that it is v_1. Then, the optimal matching covered v_1. Since no stable matching covers v_1, the optimal matching is blocked by at least one unrestricted edge of the segment. This must still be less than the number of edges blocking M', therefore the forbidden edge at v_k blocks M', but it does not block the optimal matching. This is only possible if the optimal matching covers v_k, which contradicts our assumption.

On the remaining components, M leaves v_1 and v_k unmatched in every segment. Therefore, all remaining forbidden edges block M. On the other hand, to each segment there is a matching M', covering both v_1 and v_k and admitting only one unrestricted blocking edge. A chain of such segments - always connected by a forbidden edge - can be eliminated in the following way. On the first segment, M' is fixed, then M, and so on, in an alternating manner. Consider an arbitrary segment of a chain. A matching on this segment is blocked by at least one unrestricted edge or it leaves both v_1 and v_k unmatched. Take one of the optimal solutions in which the latter case occurs the least times. Then, neither of the forbidden edges incident to v_1 and v_k can block in the optimal solution. Otherwise, fixing M' on the segment would lead to a solution that is at least as good as the optimal one. This is only possible if this segment is between two segments with at least one end vertex covered by the optimal matching. But their other end vertex then also can be covered, because M' already minimizes the number of blocking edges on a segment provided that at least one end vertex is covered. With this we showed that if M occurs in a component, then it must be between two components with M' on them. On the other hand, if M' occurs at least once, the two forbidden edges at the two end vertices do not block any more. Therefore, each of the two segments surrounding this segment is blocked by at most one edge in an optimal solution. Choosing M on these segments leads to at most one blocking edge. Therefore, the strategy of fixing M and M' in alternating manner eliminates as many blocking forbidden edges as possible. □

Even with the previous two theorems, we have not quite drawn the line between tractable and hard cases in terms of vertex degrees. The complexity of MIN BP SM RESTRICTED remains open for the case when preference lists are of length at most 2 on one side of the bipartite graph and they are of unbounded length on the other side. However we believe that this problem is solvable in polynomial time.

Conjecture 1. MIN BP SM RESTRICTED *is solvable in polynomial time if each woman's preference list consists of at most 2 elements.*

2.3.3 Stable roommates problem

Having discussed several cases of SM, we turn our attention to non-bipartite instances. Since SM is a restriction of SR, all established results on the NP-hardness and inapproximability of MIN BP SM RESTRICTED carry over to the non-bipartite SR case. As a matter of fact, more is true, since MIN BP SR RESTRICTED is NP-hard and difficult to approximate even if $P = \emptyset$ and $Q = \emptyset$ [1]. We summarize these observations as follows.

Remark 1. *By Theorems 2.3 and 2.4,* MIN BP SR FORBIDDEN *and* MIN BP SR FORCED *are* NP*-hard and not approximable within* $|V|^{1-\varepsilon}$*, for any* $\varepsilon > 0$*, unless* P = NP*. Moreover Theorems 2.6 and 2.8 imply that* MIN BP SR FORBIDDEN *and* MIN BP SR FORCED *are* NP*-hard even if all preference lists are of length at most 3 or, in the latter case,* $|Q| = 1$*. Finally* MIN BP SR RESTRICTED *is* NP*-hard and not approximable within* $|V|^{\frac{1}{2}-\varepsilon}$*, for any* $\varepsilon > 0$*, unless* P = NP*, even if* $P = \emptyset$ *and* $Q = \emptyset$ *[1].*

Remark 1 already shows that Theorem 2.5 does not carry over to the SR case, since MIN BP SR FORBIDDEN is computationally hard even if $P = \emptyset$. As for the other polynomially solvable cases, the proof of Theorem 2.7 carries over without applying any modifications. Theorem 2.9 also carries over to the SR case, but it needs a slight modification. If $\deg(v) \leq 2$ for every $v \in V(G)$, then G consists of paths and cycles. Each odd preference cycle without a forbidden edge contributes at least one blocking edge to any matching [97] and any maximal matching on such a cycle is blocked by exactly one edge. On the remainder of the graph, the algorithm described in the proof of Theorem 2.9 delivers an optimal matching for MIN BP SR RESTRICTED.

Remark 2. MIN BP SR RESTRICTED *is solvable in polynomial time if the minimal number of edges blocking an optimal solution is a constant or if each preference list consists of at most 2 elements.*

2.4 Stable matchings with the minimum number of violated constraints on restricted edges

In this section, we study the second intuitive approximation concept. The sought matching is stable and violates as few constraints on restricted edges as possible. We return to our example that already appeared in Figure 2.1. As already mentioned earlier, the instance admits a single stable matching, namely $M = \{u_1w_1, u_2w_2, u_3w_3, u_4w_4\}$. Since M

contains both forbidden edges, the minimum number of violated constraints on restricted edges is 2.

This section is structured as follows: in Section 2.4.1, complexity and approximability results are presented for SM MIN RESTRICTED VIOLATIONS, SR MIN FORBIDDEN, SR MAX FORCED and SR MIN RESTRICTED VIOLATIONS. In Section 2.4.2 we consider the complexity of SR MIN RESTRICTED VIOLATIONS when the degree of the underlying graph is bounded.

2.4.1 General complexity and approximability results

As mentioned in Section 2.1, a weighted stable matching instance models SM MIN RESTRICTED VIOLATIONS.

Theorem 2.10. SM MIN RESTRICTED VIOLATIONS *is solvable in polynomial time.*

Proof. We convert SM MIN RESTRICTED VIOLATIONS into a weighted SM problem so that a matching M is of weight $|P \cap M| - |Q \cap M| = |Q| + |P \cap M| + |Q \setminus M|$ using the following weight function:

$$\omega(e) = \begin{cases} -1 & \text{if } e \text{ is forced,} \\ 0 & \text{if } e \text{ is unrestricted,} \\ 1 & \text{if } e \text{ is forbidden.} \end{cases}$$

□

In the SR context, finding a minimum weight stable matching is NP-hard [38], so the above technique for SM does not carry over to SR. Indeed even restricted variants of SR MIN RESTRICTED VIOLATIONS are NP-hard, as the following result shows.

Theorem 2.11. SR MIN FORBIDDEN *and* SR MAX FORCED *are* NP*-hard.*

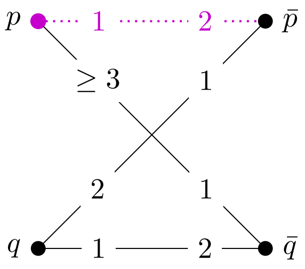
Here, we give two alternative proofs for the same theorem. On one hand, the complexity results in Theorem 2.11 can be derived by reducing the minimum vertex cover problem to our current problems. We use this reduction later to establish an inapproximability bound. On the other hand, the other reduction to prove Theorem 2.11 is somewhat shorter and uses a satisfiability problem.

Problem 8. MIN VX COVER
Input: $\mathcal{I} = G$*; a graph* G *on* n *vertices and* m *edges.*
Output: *A vertex cover* $C \subseteq V(G)$ *with* $|C| \leq |C'|$ *for every vertex cover* C'*.*

MIN VX COVER is NP-hard and cannot be approximated within a factor of $2-\varepsilon$ for any positive ε, assuming that the Unique Games Conjecture is true [64]. The set of vertices covered by any maximal matching in G gives a 2-approximation.

Proof. Given an instance (G, K) of the decision version of MIN VX COVER, the following instance (G', O, K) of the decision version of SR MIN FORBIDDEN is constructed. The entire graph G is copied, and then, a gadget is attached to each vertex $v_i \in V(G)$. It is

p_i:	$\bar{p}_i$	adjacent p vertices	$\bar{q}_i$	rest
$\bar{p}_i$:	q_i	p_i	rest	
q_i:	$\bar{q}_i$	$\bar{p}_i$	rest	
$\bar{q}_i$:	p_i	q_i	rest	



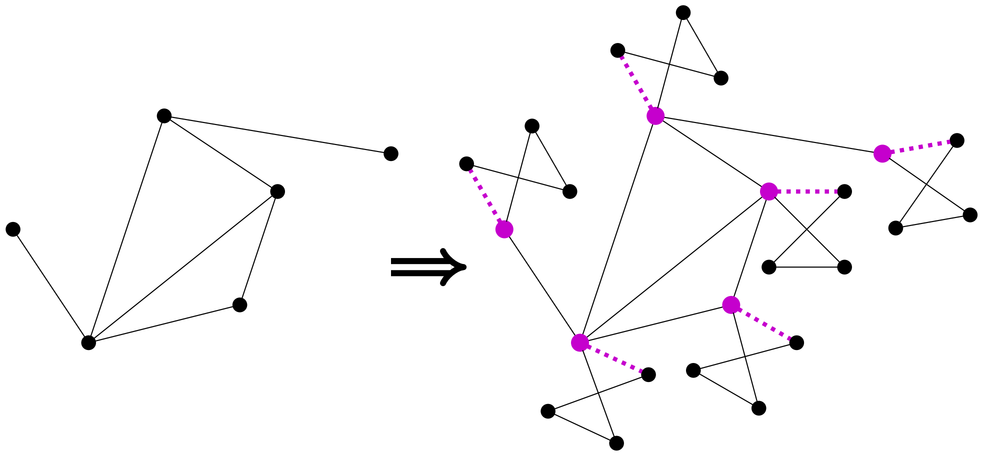


Figure 2.6: Adding $K_{2,2}$ to each vertex of the MIN VX COVER instance.

a complete bipartite graph on four vertices: one of them is $v_i = p_i$, whilst the remaining three are denoted by $\bar{p}_i, q_i$ and $\bar{q}_i$. Vertex p_i preserves v_i's preference list and places $\bar{p}_i$ at the top, and $\bar{q}_i$ at the bottom of this list. The new vertices' orderings can be seen in Figure 2.6. In order to derive an instance with complete lists, all remaining vertices can be placed in arbitrary order to the bottom of the lists. Later we will see that these edges never appear in stable matchings, neither do they block them. The set of forbidden edges is formed by all $p_i\bar{p}_i$ edges corresponding to the dotted colored edges in our illustrations in Figure 2.6.

Claim 8. *If M is a stable matching in G', then for each $1 \leq i \leq n$ either $p_i\bar{p}_i \in M$ and $q_i\bar{q}_i \in M$, or $p_i\bar{q}_i \in M$ and $\bar{p}_i q_i \in M$.*

Proof. This claim follows from the structure of the introduced gadget. First, we observe that in M, each $\bar{p}_i$ is either matched to p_i or to q_i. Otherwise, $p_i\bar{p}_i$ blocks M, since $\bar{p}_i$ is p_i's first choice. Similarly, $q_i\bar{q}_i \in M$ or $q_i\bar{p}_i \in M$. Otherwise $\bar{p}_i q_i$ would block M. Finally, $p_i\bar{q}_i \in M$ or $q_i\bar{q}_i \in M$, otherwise $q_i\bar{q}_i$ blocks M. These three requirements imply that for each $1 \leq i \leq n$ either $p_i\bar{p}_i \in M$ and $q_i\bar{q}_i \in M$, or $p_i\bar{q}_i \in M$ and $\bar{p}_i q_i \in M$. ■

Claim 9. *If there is a vertex cover $C \subseteq V(G)$ with $|C| \leq K$, then there is a stable* *matching M in G for which $|M \cap P| \leq K$.*

Proof. The matching M is constructed based on the following case distinction:

$$\begin{aligned} \{p_i\bar{p}_i, q_i\bar{q}_i\} \subseteq M \quad &\text{if} \quad v_i \in C \\ \{p_i\bar{q}_i, \bar{p}_iq_i\} \subseteq M \quad &\text{if} \quad v_i \notin C \end{aligned}$$

Clearly $|M \cap P| \leq K$. Moreover, no edge in the gadgets can block M, because the preferences inside the gadget are cyclic. Due to the vertex cover property, edges between two p-vertices have at least one end vertex in C, thus, at least one of their end vertices is matched to its first-choice partner $\bar{p}$ in G'. For each vertex in G', the edges in the sets "rest" are worse than the edge in M. ■

Claim 10. *If there is a stable matching M in G for which $|M \cap P| \leq K$, then there is a vertex cover $C \subseteq V(G)$ with $|C| \leq K$.*

Proof. Claim 8 allows us to investigate only two cases per gadget. Exactly the same function is used to derive C from M, as above, in the opposite direction:

$$\begin{aligned} v_i \in C \quad &\text{if} \quad \{p_i\bar{p}_i, q_i\bar{q}_i\} \subseteq M \\ v_i \notin C \quad &\text{if} \quad \{p_i\bar{q}_i, \bar{p}_iq_i\} \subseteq M \end{aligned}$$

Trivially, $|C| \leq K$. Suppose C is not a vertex cover. Then, there is an edge $v_iv_j = p_ip_j$ for which $\{p_i\bar{q}_i, p_j\bar{q}_j\} \subset \{p_i\bar{q}_i, \bar{p}_iq_i, p_j\bar{q}_j, \bar{p}_jq_j\} \subseteq M$. Then p_ip_j blocks M. ■

To prove the complexity result for SR MAX FORCED, let $Q = \{p_i\bar{q}_i : 1 \leq i \leq n\}$. □

The second NP-hard problem we reduce to SR MIN FORBIDDEN is weighted-2-satisfiability. It can be regarded as a generalization of the minimum vertex cover problem in undirected graphs.

Problem 9. W2SAT
Input: $\mathcal{I} = B$; *a Boolean formula B with n variables in m clauses, each of them consisting of exactly 2 non-negated literals.*
Output: *A valid truth assignment that maximizes the number of variables set to* true.

Proof. To each variable x_i in B we introduce four vertices in our SR instance: $p_i, \bar{p}_i, q_i$ and $\bar{q}_i$. The complete preference lists are formed based on the clauses that appear in B. Their structure can be sketched in the following way:

p_i:	$\bar{p}_i$	intermediate p vertices	$\bar{q}_i$	rest
$\bar{p}_i$:	q_i	p_i	rest	
q_i:	$\bar{q}_i$	$\bar{p}_i$	rest	
$\bar{q}_i$:	p_i	q_i	rest	

The block of intermediate vertices on p_i's preference list consists of all p_j vertices in arbitrary order for which $(x_i \vee x_j)$ is a clause in B. All four preference lists are completed by the remainder of $V(G)$. To each variable x_i, a single forbidden edge $p_i\bar{p}_i$ is introduced.

Claim 11. *If f is a satisfying truth assignment of B with at most K* true *variables, then there is a stable matching M in G for which $|M \cap P| \leq K$.*

Proof. Let us define M in the following way. For each $1 \leq i \leq n$

$$\begin{array}{lll} \{p_i\bar{p}_i, q_i\bar{q}_i\} \subseteq M & \text{if} & f(x_i) = \mathsf{true} \\ \{p_i\bar{q}_i, \bar{p}_i q_i\} \subseteq M & \text{if} & f(x_i) = \mathsf{false} \end{array}$$

Trivially, $|M \cap P| \leq K$. It also follows from the construction that every vertex in $V(G)$ is matched to a vertex better than any other placed in the last block on its preference list, denoted by rest above. Therefore, this block can also be omitted. Suppose that M is unstable, i.e., there is a blocking edge. This blocking edge must belong to either of the following groups:

- $p_i\bar{p}_i, p_i\bar{q}_i, \bar{p}_i q_i$ or $q_i\bar{q}_i$: They are last-choice edges of either of their end vertices, therefore, they cannot block M.
- $p_i p_j$: The values of x_i and x_j are determined by the fact that $p_i p_j$ is better than p_i's and p_j's edges in M. Since $p_i\bar{q}_i \in M$ and $p_j\bar{q}_j \in M$, $f(x_i) = f(x_j) =$ false. Since p_j appeared on p_i's list as a intermediate vertex, $(x_i \vee x_j)$ appeared in B. Thus, f is not a satisfying truth assignment. ■

Claim 12. *If there is a stable matching M in G for which $|M \cap P| \leq K$, then there is a satisfying truth assignment f that has at most K* true *variables.*

Proof. First, we observe that in M, each $\bar{p}_i$ is either matched to p_i or to q_i. Otherwise, $p_i\bar{p}_i$ blocks M, since $\bar{p}_i$ is p_i's first choice. Similarly, $q_i\bar{q}_i \in M$ or $q_i\bar{p}_i \in M$. Otherwise $\bar{p}_i q_i$ would block M. Finally, $p_i\bar{q}_i \in M$ or $q_i\bar{q}_i \in M$, otherwise $q_i\bar{q}_i$ blocks M. These three requirements imply that for each $1 \leq i \leq n$ either $p_i\bar{p}_i \in M$ and $q_i\bar{q}_i \in M$, or $p_i\bar{q}_i \in M$ and $\bar{p}_i q_i \in M$. At this point, we are ready to construct the truth assignment, since either $\{p_i\bar{p}_i, q_i\bar{q}_i\} \subseteq M$ or $\{p_i\bar{q}_i, \bar{p}_i q_i\} \subseteq M$.

$$\begin{array}{lll} f(x_i) = \mathsf{true} & \text{if} & \{p_i\bar{p}_i, q_i\bar{q}_i\} \subseteq M \\ f(x_i) = \mathsf{false} & \text{if} & \{p_i\bar{q}_i, \bar{p}_i q_i\} \subseteq M \end{array}$$

Function f has at most K true variables. The last step in our proof is to show that f is a valid truth assignment. Suppose that an invalid $(x_i \vee x_j)$ is in B. Since both literals are false, $\{p_i\bar{q}_i, p_j\bar{q}_j\} \subset \{p_i\bar{q}_i, \bar{p}_i q_i, p_j\bar{q}_j, \bar{p}_j q_j\} \subseteq M$. At the same time, p_i is listed on p_j's preference list as a intermediate vertex and vice versa. Thus, $p_i p_j$ blocks M. ■

Our proof for the NP-hardness of the problem SR MIN FORBIDDEN can be converted easily to SR MAX FORCED. Instead of labeling all $p_i\bar{p}_i$ edges as forbidden, here we take all $p_i\bar{q}_i$ edges as forced edges. The set of stable matchings in G is not impaired by this modification. The bijection between pairs of edges in M and truth values of variables in B implies that $|M \cap Q|$ equals the number of false variables. □

In our first proof above, we reduced MIN VX COVER to SR MIN FORBIDDEN and SR MAX FORCED. Since MIN VX COVER is NP-hard and cannot be approximated within a factor of $2 - \varepsilon$ for any positive ε, assuming that the Unique Games Conjecture is true [64], the reduction also answers basic questions about the approximability of our two problems. Since any vertex cover on K vertices can be interpreted as a stable matching containing K forbidden edges in SR MIN FORBIDDEN and vice versa, the $(2 - \varepsilon)$-inapproximability result carries over. The same holds for the number of violated forced edge constraints in SR MAX FORCED. On the positive side, we can close the gap with the best possible approximation ratio if $Q = \emptyset$ or $|Q|$ is sufficiently large. To derive this result, we use the 2-approximability of weighted SR for non-negative weight functions [38, 100]. Due to the non-negativity constraint, the case of $0 < |Q| < |M|$ remains open.

Theorem 2.12. *If $|Q| \geq |M|$ for a stable matching M, then* SR MIN RESTRICTED VIOLATIONS *is 2-approximable in polynomial time.*

Proof. We know that the NP-hard minimum weight SR problem is 2-approximable in polynomial time [38, 100] for all non-negative weight functions. The assumption $Q = \emptyset$ or $|Q| \geq |M|$ is needed to guarantee non-negativity. A simple computation shows that the weight of the a stable matching M is $|Q \setminus M| + |P \cap M|$, if the following weight function is defined on the edges:

$$\omega(e) = \begin{cases} \frac{|Q|}{|M|} - 1 & \text{if } e \text{ is forced,} \\ \frac{|Q|}{|M|} & \text{if } e \text{ is unrestricted,} \\ \frac{|Q|}{|M|} + 1 & \text{if } e \text{ is forbidden.} \end{cases}$$

□

When studying SR MAX FORCED, we measured optimality by keeping track of the number of violated constraints. One might find it more intuitive instead to maximize $|Q \cap M|$, the number of forced edges in the stable matching. Our NP-hardness proof for SR MAX FORCED remains intact, but the approximability results need to be revisited. In fact, this modification of the measure changes the approximability of the problem as well:

Theorem 2.13. *For* SR MAX FORCED*, the maximum of $|Q \cap M|$ cannot be approximated within $|V|^{\frac{1}{2}-\varepsilon}$ for any $\varepsilon > 0$, unless* P = NP.

Proof. We adapt the first proof of Theorem 2.11 so that the reduction is from MAX IND SET, the problem of finding a maximum independent set in a given graph $G = (V, E)$. MAX IND SET is not approximable within $N^{1-\varepsilon}$ for any $\varepsilon > 0$, unless P = NP [106], where $N = |V|$. In the modified reduction the forced edges comprise $Q = \{p_i\bar{q}_i : 1 \leq i \leq n\}$. An independent set of vertices S in G corresponds to the matching

$$M = \{p_i\bar{q}_i, \bar{p}_i q_i : v_i \in S\} \cup \{p_i\bar{p}_i, q_i\bar{q}_i : v_i \notin S\}$$

in the constructed instance $\mathcal{I}$ of SR MAX FORCED. Suppose that A is a $|V|^{\frac{1}{2}-\varepsilon}$-approximation algorithm that approximates $|Q \cap M_{opt}|$ in $\mathcal{I}$, for some $\varepsilon > 0$, where M_{opt} is an optimal solution in $\mathcal{I}$. Note that $|S_{opt}| = |M_{opt}|$, where S_{opt} is a maximum independent set in G. Moreover $|V|^{\frac{1}{2}-\varepsilon} = (4N)^{\frac{1}{2}-\varepsilon} \leq N^{1-\varepsilon}$ since $|V| = 4N$ and without loss of generality we can assume that $N \geq 4$. We thus reach a contradiction to the inapproximability of MAX IND SET. $\square$

2.4.2 Bounded parameters

We now turn to the complexity of SR MIN RESTRICTED VIOLATIONS and its variants when the degree of the underlying graph is bounded or some parameter of the instance can be considered as a constant. With Theorems 2.14 and 2.15 we draw the line between NP-hard and polynomially solvable cases in terms of degree constraints.

Theorem 2.14. SR MIN FORBIDDEN *and* SR MAX FORCED *are* NP*-hard even if every preference list is of length at most 3.*

Proof. Let $\mathcal{I} = (G, K)$ be an instance of the decision version of MIN VX COVER in a cubic graph, where $G = (V, E)$, $E = \{e_1, \dots, e_m\}$ and $V = \{v_1, \dots, v_n\}$. For each i, $1 \leq i \leq n$, suppose that v_i is incident to edges e_{j_1}, e_{j_2} and e_{j_3} in G, where without loss of generality $j_1 < j_2 < j_3$. Define $e_{i,s} = e_{j_s}$, where $s \in \{1, 2, 3\}$. Similarly for each j, $1 \leq j \leq m$, suppose that $e_j = v_{i_1}v_{i_2}$, where without loss of generality $i_1 < i_2$. Define $v_{j,r} = v_{i_r}$, $r \in \{1, 2\}$. Note that these are not vertices or edges of the SR MIN FORBIDDEN instance we are constructing, it is only notation necessary to introduce in order to define the graph.

We construct an instance $\mathcal{I}'$ of SR MIN FORBIDDEN as follows. The set $V' \cup W \cup E' \cup F$ constitutes the set of vertices in $\mathcal{I}'$. The preference lists of these vertices (also indicating the edges of the graph) are as shown in Figure 2.7. In the preference list of a vertex v_i^r, the symbol $e(v_i^r)$ denotes vertex $e_j^s \in E'$ such that $e_j = e_{i,r}$ and $v_i = v_{j,s}$. Since i and r are given, e_j can be computed. Now we know i and j in the second equation, therefore we can compute s. Similarly in the preference list of vertex e_j^s, the symbol $v(e_j^s)$ denotes vertex $v_i^r \in V'$ such that $e_j = e_{i,r}$ and $v_i = v_{j,s}$.

$$\begin{aligned}
V' &= \{v_i^r : 1 \leq i \leq n \wedge r \in \{1, 2, 3\}\} \\
W &= \{w_i^r : 1 \leq i \leq n \wedge r \in \{1, 2, 3\}\} \\
E' &= \{e_j^s : 1 \leq j \leq m \wedge s \in \{1, 2\}\} \\
F &= \{f_j^s : 1 \leq j \leq m \wedge s \in \{1, 2\}\}
\end{aligned}$$

Let $P = \{v_i^1 w_i^1 : 1 \leq i \leq n\}$ be the set of forbidden edges in $\mathcal{I}'$. The edges connecting V' and E' capture the incidence relations of the original graph G, while vertices in W and F can be seen as garbage collectors.

Finally we define some further notation in $\mathcal{I}$. For each i, $1 \leq i \leq n$, let $V_i^c = \{v_i^r w_i^r : r \in \{1, 2, 3\}\}$ and let $V_i^u = \{v_i^r w_i^{r+1} : r \in \{1, 2, 3\}\}$, where addition is taken modulo 3. Note that each V_i^c contains exactly one forbidden edge, while V_i^u has no forbidden edge. Similarly for each j, $1 \leq j \leq m$, let $E_j^1 = \{e_j^1 e_j^2, f_j^1 f_j^2\}$ and let $E_j^2 = \{e_j^1 f_j^2, e_j^2 f_j^1\}$.

Claim 13. *$\mathcal{I}'$ admits a stable matching in which every vertex is matched.*

$v_i^1:$	$w_i^1 \quad e(v_i^1) \quad w_i^2$	$(1 \leq i \leq n)$		$w_i^1:$	$v_i^3 \quad v_i^1$	$(1 \leq i \leq n)$
$v_i^2:$	$w_i^2 \quad e(v_i^2) \quad w_i^3$	$(1 \leq i \leq n)$		$w_i^2:$	$v_i^1 \quad v_i^2$	$(1 \leq i \leq n)$
$v_i^3:$	$w_i^3 \quad e(v_i^3) \quad w_i^1$	$(1 \leq i \leq n)$		$w_i^3:$	$v_i^2 \quad v_i^3$	$(1 \leq i \leq n)$
$e_j^1:$	$e_j^2 \quad v(e_j^1) \quad f_j^2$	$(1 \leq j \leq m)$		$f_j^1:$	$f_j^2 \quad e_j^2$	$(1 \leq j \leq m)$
$e_j^2:$	$f_j^1 \quad v(e_j^2) \quad e_j^1$	$(1 \leq j \leq m)$		$f_j^2:$	$e_j^1 \quad f_j^1$	$(1 \leq j \leq m)$

Figure 2.7: Preference lists in the constructed instance of SR MIN FORBIDDEN.

Proof. Let $M = \bigcup_{i=1}^{n} V_i^c \cup \bigcup_{j=1}^{m} E_j^1$. Starting with the argument that each v_i^r, each e_j^1 and each f_j^1 receives its best-choice edge in M, it is straightforward to verify that M is stable: the remaining vertices list only edges running to these vertices better than their matching edges. The Rural Hospitals Theorem implies then that every stable matching in $\mathcal{I}'$ matches every vertex in $\mathcal{I}'$. ■

In Claims 14 and 15 we show that G has a vertex cover C where $|C| \leq K$ if and only if $\mathcal{I}'$ has a stable matching M where $|M \cap P| \leq K$.

Claim 14. *If G has a vertex cover C such that $|C| \leq K$ in $\mathcal{I}$, then there is a stable matching M in $\mathcal{I}'$ with $|M \cap P| \leq K$.*

Proof. We construct a matching M in $\mathcal{I}$ as follows. For each i, $1 \leq i \leq n$, if $v_i \in C$, add V_i^c to M, otherwise add V_i^u to M. For each j, $1 \leq j \leq m$, if $v_{j,1} \in C$, add E_j^2 to C, otherwise add E_j^1 to C. Then $|M \cap P| = |C| \leq K$.

Now we verify that M is stable in $\mathcal{I}$. The edge sets V_i^c, V_i^u, E_j^1 and E_j^2 are defined in such a way that edges running between V' and W or between E' and F are always dominated by them at one end. Edges connecting V' and E' cannot block for the following reasons. These edges dominate M at v_i^r only if $v_i \notin C$ (and therefore, V_i^u was added to M). Since C was a vertex cover, every e_j incident to v_i in G has its other end vertex in C. Assume now that $v_i = v_{j,1}$. The proof for the case $v_i = v_{j,2}$ is analogous.

We know that every e_j with $v_{j,1} \notin C$ is covered by an E_j^1 subgraph in M. These edges dominate all non-matching edges at e_j^1 vertices, thus e_j^2 is the only potential end vertex of a blocking edge in E'. But e_j^2 is only adjacent to vertices derived from $v_{j,2}$, and those are matched to their best choice partners in M, because $v_{j,2} \in C$. ■

Claim 15. *If there is a stable matching M with $|M \cap P| \leq K$ in $\mathcal{I}'$, then G has a vertex cover C in $\mathcal{I}$ such that $|C| \leq K$.*

Proof. We construct a set of vertices C in G as follows. Claim 13 states that M matches every vertex in $\mathcal{I}$, then for each i, $1 \leq i \leq n$, either $V_i^c \subseteq M$ or $V_i^u \subseteq M$. In the former case add v_i to C. As $|M \cap P| \leq K$, it follows that $|C| \leq K$. Also, for each j, $1 \leq j \leq m$, as M matches every vertex in $\mathcal{I}$, either $E_j^1 \subseteq M$ or $E_j^2 \subseteq M$.

Assume that C is not a vertex cover in G, i.e., there is an edge $e_j = v_{j,1}v_{j,2}$ such that for both $v_{j,1}$ and $v_{j,2}$, and for the corresponding $v_{i_1} = v_{j,1}$ and $v_{i_2} = v_{j,2}$, the edge sets

$V_{i_1}^u \in M$ and $V_{i_2}^u \in M$. In this case, M is dominated by some edge connecting V' and E' at v_i^r vertices. In $\mathcal{I}'$, the vertices corresponding to v_{i_1} and v_{i_2} are adjacent to e_j^1 and e_j^2, respectively. All these edges dominate M at their end vertex in V'. Since it is either E_j^1 or E_j^1 that covers e_j^1 and e_j^2, one of these vertices are matched to its worst partner, thus we found an edge that blocks M. ■

For SR MAX FORCED, an analogous proof can be derived if we define the set of forced edges as $Q = \{v_i^1 w_i^2 : 1 \leq i \leq n\}$. □

Theorem 2.15. SR MIN RESTRICTED VIOLATIONS *is solvable in* $\mathcal{O}(|V|)$ *time if every preference list is of length at most 2.*

Proof. Since the set of matched vertices is the same in all stable matchings, finding a stable matching in $\mathcal{O}(|V|)$ time in these very strongly restricted instances marks all vertices that need to be matched. In each component, there are at most two possible matchings satisfying these constraints. We choose the one that is stable and violates fewer constraints. □

Short preference lists are not the only case when SR MIN RESTRICTED VIOLATIONS becomes tractable, as our last theorem shows.

Theorem 2.16. SR MIN RESTRICTED VIOLATIONS *is solvable in polynomial time if the number of restricted edges or the minimal number of violated constraints is constant.*

Proof. Suppose that the number of restricted edges is L. No stable matching can violate more than L constraints on restricted edges, therefore, it is sufficient to investigate the case when the target number of violated constraints in the sought solution satisfies $K \leq L$. Choose a set of restricted edges of cardinality K, where $K \leq L$. For all edges in this set, reverse the restriction: let forced edges become forbidden and forbidden edges become forced. With the modified set of restricted edges, stable matchings violate exactly K constraints on restricted edges. Checking all edge sets of cardinality at most K takes $\mathcal{O}(|E|^K) = \mathcal{O}(|E|^L)$ iterations. □

2.5 Conclusion and open problems

In this chapter, we investigated the stable marriage and the stable roommates problems on graphs with forced and forbidden edges. Since a solution satisfying all constraints need not exist, two relaxed problems were defined. In MIN BP SM RESTRICTED, constraints on restricted edges are strict, while a matching with the minimum number of blocking edges is searched for. On the other hand, in SR MIN RESTRICTED VIOLATIONS, we seek stable solutions that violate as few constraints on restricted edges as possible. For both problems, we determined the complexity and studied several special cases.

One of the most striking open questions is the approximability of SR MIN RESTRICTED VIOLATIONS if $0 < |Q| < |M|$. Our other open question is formulated as Conjecture 1: Is there a polynomial algorithm for MIN BP SM RESTRICTED if each woman's preference list consists of at most 2 elements? A more general direction of further research involves the SM MIN RESTRICTED VIOLATIONS problem. We have shown that it can be solved in polynomial time, due to algorithms for minimum weight stable marriage. The following question arises naturally: is there a faster method for SM MIN RESTRICTED VIOLATIONS that avoids reliance on Feder's algorithm or linear programming methods?

3 Other complexity results for stable matchings

In this chapter we investigate the complexity of two additional problems in the one-to-one stable matching setting. The first problem, MAX SM FREE is defined on bipartite stable matching instances with free edges. Free edges can be seen as a third sort of restricted edges, besides forced and forbidden edges: they can appear in stable matchings, but they are not able to block matchings by definition. While a stable matching always exists in the presence of free edges, we show that a maximum cardinality stable matching is NP-hard to find. This result has been proved independently by Askalidis et al. [5].

The second problem tackled is a degree constrained version of the weakly stable roommates problem with ties (or shorter, WEAK SRT), which is known to be NP-complete [85]. Here we prove that WEAK 3-SRTI, the weakly stable roommates problem with preference lists of length at most 3 is already NP-complete, even if the preference lists are either strictly ordered or comprised of a single tie of length 2. Later we also study MIN BP 3-SRTI, the almost stable approximation version of the previous problem and prove an inapproximability bound. At the end of this chapter, a polynomial time algorithm for the case of degrees at most 2 is presented.

The results presented in Section 3.2 are joint work with David F. Manlove.

This short chapter is divided into two sections, each of them focused on a computationally hard variant of SM or SR. Since the two settings only share their one-to-one matching nature, we present them separately, motivating and placing them into the literature on their own.

3.1 Maximum stable marriage with free edges

Motivation. An edge $uw \in E(G)$ of an SM instance can play three essentially different roles with respect to a (not necessarily stable) matching M. Based on these roles, $E(G)$ can be partitioned into the following three disjoint subsets.

Set 1: $uw \in M$

Set 2: $uw \notin M$ and uw is dominated by some edge in M

Set 3: $uw \notin M$ and uw blocks M $(uw \in bp(M))$

An equivalent definition of stability is that M is stable if and only if Set 3 is empty. Introducing restrictions on these sets leads to concepts that are interesting not only from the theoretical point of view, but also from the perspective of applications.

Forced and forbidden edges are derived from the first two properties above, i.e., forced edges must fall into Set 1, while forbidden edges must fall into Set 2. A third type of special edges can be introduced with the help of the third property. Here we declare a matching stable if it is only blocked by edges in a predefined edge set F, called *free* edges. In other terms, Set 3 must be a subset of F.

Literature review. Clearly, the presence of free edges allows a larger set of solutions. Cechlárová and Fleiner [21] show that the polynomially solvable SR becomes NP-complete when free edges appear in the graph. The application motivating this research is the living donor kidney exchange program. Free edges can model special market contacts, where pair forming is controlled by central authorities. Such deals can be made in order to reach stability, but they also can be left unused. In the latter case, the pair of agents does not block the matching, since they might not have full information on the preferences in the system. Moreover, all agents know that their partnership is strictly controlled by some authority and they are not able to to cooperate on their own.

Recently, Askalidis et al. published several results on free edges [5]. They call the problem the *socially stable* matching problem, interpreting free edges as lack of social ties between agents of a many-to-one SM instance. After proving that the problem of finding a maximum socially stable matching in SM is NP-hard even if the preference lists are of length at most 3, they also give a 2/3-approximation algorithm and show that this approximation is the best possible, assuming the Unique Games Conjecture.

Our contribution. In this section we provide an alternative proof for the NP-completeness of the maximum cardinality stable marriage problem with free edges. Askalidis et al. [5] reduce the NP-complete perfect weakly stable marriage problem [74] to our problem, while we use the very closely related maximum cardinality weakly stable marriage problem. Our proof is more technical than theirs.

Problem 10. MAX SM FREE
Input: $\mathcal{I} = (G, O, F)$*; an* SM *instance and a set of free edges* F*.*
Output: A matching M *such that* $bp(M) \subseteq F$ *and* $|M| \geq |M'|$ *for every matching* M' *in* G *with* $bp(M') \subseteq F$*.*

The NP-hard problem we reduce to MAX SM FREE is the maximum cardinality weakly stable marriage problem with incomplete lists.

Problem 11. MAX WSMI
Input: $\mathcal{I} = (G, O)$*; an* SM *instance in an incomplete bipartite graph with ties.*
Output: A weakly stable matching M *such that* $|M| \geq |M'|$ *for every weakly stable matching* M'*.*

Theorem 3.1 (Manlove at al. [74])**.** MAX WSMI *is* NP*-hard even if ties only occur on the men's side, only at the bottom of the list and they are of length 2.*

We are now ready to present our hardness proof.

Theorem 3.2. MAX SM FREE *is* NP*-hard.*

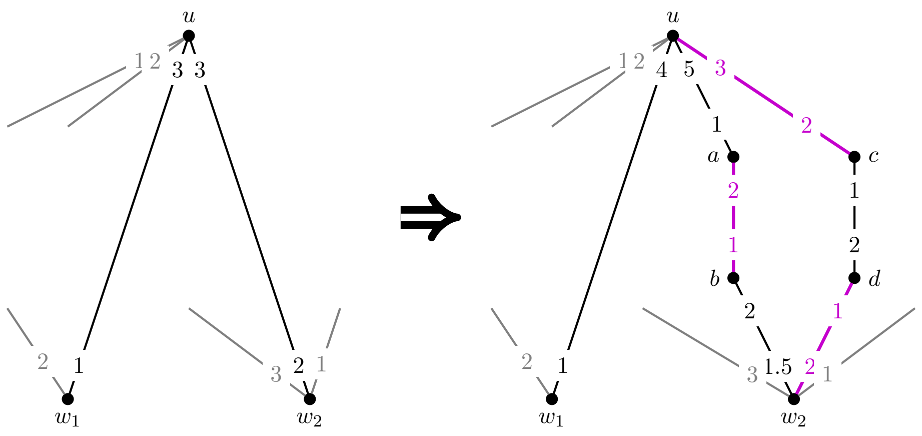


Figure 3.1: Constructing a MAX SM FREE instance from an example restricted MAX WSMI instance. The free edges are colored purple.

Proof. For the sake of simplicity, we work with the decision versions of both problems. Each MAX WSMI instance $\mathcal{I}$ can be converted into an instance $\mathcal{I}'$ of MAX SM FREE. The construction is very simple, the two instances differ only at edges that form the ties in $\mathcal{I}$. As Theorem 3.1 states it, we can assume that all ties are of length 2 and they occur at the bottom of men's list. Let us denote the two edges in a tie by uw_1 and uw_2. In $\mathcal{I}'$, uw_1 remains unchanged, its rank on u's preference list is $\deg(u)$. On the other hand, we substitute uw_2 with two parallel paths: $\langle u, a, b, w_2 \rangle$ and $\langle u, c, d, w_2 \rangle$. Edges uc, dw_2 and ab are free. The preferences are the following.

$$\begin{aligned}
\text{rank}_u(ua) &= \deg(u) + 1 \\
\text{rank}_u(uw_1) &= \deg(u) \\
\text{rank}_u(uc) &= \deg(u) - 1 \\
\text{rank}_a(ua) &= 1 \\
\text{rank}_a(ab) &= 2 \\
\text{rank}_b(ba) &= 1 \\
\text{rank}_b(bw_2) &= 2
\end{aligned}
\qquad
\begin{aligned}
\text{rank}_c(dc) &= 1 \\
\text{rank}_c(uc) &= 2 \\
\text{rank}_d(dw_2) &= 1 \\
\text{rank}_d(dc) &= 2 \\
\text{rank}_{w_2}(bw_2) &= \text{rank}_{w_2}(uw_2) \text{ in } \mathcal{I} - 0.5 \\
\text{rank}_{w_2}(dw_2) &= \text{rank}_{w_2}(uw_2) \text{ in } \mathcal{I}
\end{aligned}$$

Such an instance conversion is illustrated in Figure 3.1.

Claim 16. *If there is a weakly stable matching M with $|M| \geq K$ for every $K \in \mathbb{N}_{\geq 0}$ in $\mathcal{I}$, then there is a stable matching M' with free edges in $\mathcal{I}'$ such that $|M'| \geq K + 2$.*

Proof. The construction of M' is as follows. Apart from uw_2 for every $u \in U$, we copy all edges of M to $\mathcal{I}'$. For the rest of the graph, three scenarios are possible, as also shown in Figure 3.2.

Scenario 1: If $uw_2 \in M$, then $uc, dw_2, ab \in M'$.

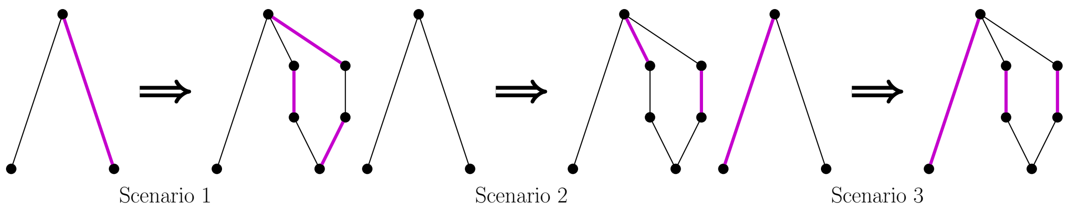


Figure 3.2: The three possible scenarios for M. The purple edges in the transformed graph belong to M'.

Scenario 2: If u is unmatched (and w_1 and w_2 are both matched to better partners than u), then $ua, cd \in M'$.

Scenario 3: If u is matched to a vertex different from w_2, then $cd, ab \in M'$.

In all three cases, $|M'| = |M| + 2$. Moreover, edges in $\mathcal{I}$ preserve their dominance relationship in $\mathcal{I}'$: each edge dominated by the M-edge in $\mathcal{I}$ remains dominated by its representative in M', and each edge dominating the M-edge also dominates its copy in $\mathcal{I}'$. This is due to the construction of preference lists of u and w_2: their edges substituting $uw \in E(G)$ are placed where $uw \in E(G)$ stood. Also note that the transformations do not change the matched or unmatched status of u, w_1 and w_2, except for Scenario 2, where the originally unmatched u gets matched in $\mathcal{I}'$. But in this case, u's edge is its worst ranked edge, which is equivalent to being unmatched. We shall now show that M' is a stable matching in $\mathcal{I}'$.

Suppose M' is blocked by an edge. Since M was weakly stable in $\mathcal{I}$, the blocking edge must be incident to at least one of the vertices whose position has been changed: u, w_1, w_2, a, b, c and d. Three of their edges, ab, uc and dw_2, are free, therefore they cannot block. If the blocking edge is incident to u, it has to be ua or uw_1, because only these unrestricted edges on u's preference list were affected by the transformation. Similarly, if it is incident to w_2, it must be bw_2. The last edge that could block M' is cd. We consider for each of these four edges individually Scenarios 1,2 and 3 and lead the assumption that the edge is blocking to a contradiction. The two reasons that contradict an edge to be blocking are

(i) it is in M' or

(ii) one of its end vertices is matched in M' to a better vertex.

To avoid an elaborate case distinction we list for all 12 cases the applicable reason for the contradiction in Table 3.1. For contraction reason (ii) we include in the table the name of the end vertex matched to a better vertex. With this we showed that no edge blocks M'. ■

Claim 17. *If there is a stable matching M' with free edges in $\mathcal{I}'$ such that $|M'| \geq K+2$, then there is a weakly stable matching M with $|M| \geq K$ in $\mathcal{I}$ for every $K \in \mathbb{N}_{\geq 0}$.*

	Scenario 1	Scenario 2	Scenario 3
ua	(ii) u	(i)	(ii) u
cd	(ii) d	(i)	(i)
bw_2	(ii) b	(ii) w_2	(ii) b
uw_1	(ii) u	(ii) w_1	(i) or (ii) u

Table 3.1: The four possibly blocking edges in the three scenarios and the reason why they do not block M'.

Proof. If M' is given, it is straightforward to construct a stable matching M in $\mathcal{I}$. Let M and M' be identical on all edges apart from uw_2 for every $u \in U$. Note that $\mathcal{I}'$ was constructed in such a way that the subgraph spanned by the vertices u, a, b, c, d and w_2 contains either two or three edges of any stable matching M. It is easy to see that apart from the three Scenarios above, exactly one more matching, $\{ua, cd, bw_2\}$ can also be part of a stable matching M'. Considering all cases, if u is matched to a or c in M' and w_2 is matched to b or d, then let $uw_2 \in M$, otherwise let $uw_2 \notin M$. These operations decrease the cardinality of the matching by exactly 2.

All that remains is to show that no edge blocks M. The preference lists were only changed inside of the ties: any dominance between edges in the tie and out of it is still valid. Thus, the only edges that can block M are uw_1 and uw_2. They are at the bottom of the preference list of u and hence they only block M if u is unmatched and w_1 or w_2 has a worse partner than u or they are unmatched. Suppose uw_1 blocks M. Since the same edge did not block M' in $\mathcal{I}'$, $uc \in M'$. This implies that $dw_2 \in M''$ as well, otherwise cd blocks M'. Then $uw_2 \in M$ by construction, which contradicts our assumption that $uw_2 \notin M$. Let us consider the other case, when uw_2 blocks M for some $uw_2 \in E(G)$. As mentioned above, u is then unmatched in M. Because of edge ua, u cannot be unmatched in M' and since u is then unmatched in M, either $ua \in M'$ or $uc \in M'$. Thus, if $uw_2 \notin M$, then $bw_2 \notin M'$ and $dw_2 \notin M'$, otherwise we would have included uw_2 in M. Then cd blocks M'. ■ □

We remark here that there is an alternative, analogously structured proof of Theorem 3.2 that avoids most of the technical details. If two parallel edges are introduced instead of the paths $\langle u, a, b, w_2\rangle$ and $\langle u, c, d, w_2\rangle$, then most of the case distinctions can be spared. On the other hand, the hardness of MAX SM FREE is proved then for instances with parallel edges, which is against the convention.

3.2 Stable roommates with ties and short preference lists

Motivation. As mentioned in Section 1.1.2, ties in the preference lists allow us to define weak, strong and super stability, out of which weak stability is the most actively researched concept. An edge $uv \notin M$ blocks M in the weak sense if v is strictly preferred to $M(u)$ by u and conversely, u is strictly preferred to $M(v)$ by v. To the best of our knowledge, there has not been any work published on weakly stable matchings in SR with bounded length preference lists. Here we make an attempt to fill this gap in the literature.

Literature review. Presenting a highly technical reduction from 3-SAT, Ronn [85] proves the NP-completeness of WEAK SRT, the weakly stable roommates problem with ties. Gusfield and Irving [49] give a different reduction from R-3-SAT, a variant of 3-SAT, where no literal occurs more than twice. Finally, Irving and Manlove [55] provide a short proof using the NP-complete perfect weakly stable bipartite matching problem [74]. This latter problem is shown to be hard via a reduction from 1-IN-3-SAT –where exactly one literal must be true in each clause– by Iwama et al. [57]. Even though all these proofs including our proof use a reduction from a variant of 3-SAT, the gadgets employed to obtain them are essentially different.

Our contribution. First we define the stable roommates problem with ties and bounded-length preference lists and its almost stable variant formally. Then, we present an NP-completeness and an inapproximability proof for the case with degree bound 3, and finally finish with a polynomial time algorithm for the case of degrees at most 2.

Problem 12. d-SRTI
Input: $\mathcal{I} = (G, O)$*; an* SR *instance with ties and preference lists of length at most* d*.*
Question: *Is there a weakly stable matching* M *in* $\mathcal{I}$*?*

Problem 13. MIN BP d-SRTI
Input: $\mathcal{I} = (G, O)$*; an* SR *instance with ties and preference lists of length at most* d*.*
Output: *A matching* M *in* $\mathcal{I}$ *so that* $|bp(M)| \leq |bp(M')|$ *for every matching* M' *in* $\mathcal{I}$*.*

Both for the hardness proof of 3-SRTI and for the inapproximability result on MIN BP 3-SRTI, (2,2)-E3-SAT (Problem 7) is used. We begin with the hardness of 3-SRTI and then build on the constructed graph in the proof of Theorem 3.5, which is the inapproximability result on MIN BP 3-SRTI.

Theorem 3.3. 3-SRTI *is* NP*-complete.*

Proof. In this proof, we use ideas of Biró et al. [16], in particular the gadgets designed to prove the NP-completeness of finding an almost stable roommates assignment in bounded-degree graphs, called MIN BP 3-SRI.

First, we sketch the 3-SRTI instance $\mathcal{I} = (G, O)$ created to a given Boolean formula B. Each graph G consists of a variable gadget, a clause gadget and a set of interconnecting edges between them. These subgraphs and the preference orderings on edges are shown in Figure 3.3.

When constructing graph G for a given Boolean formula B, we keep track of the order of the three literals in each clause and the order of the two unnegated and two negated appearances of each variable. Each appearance is represented by an interconnecting edge.

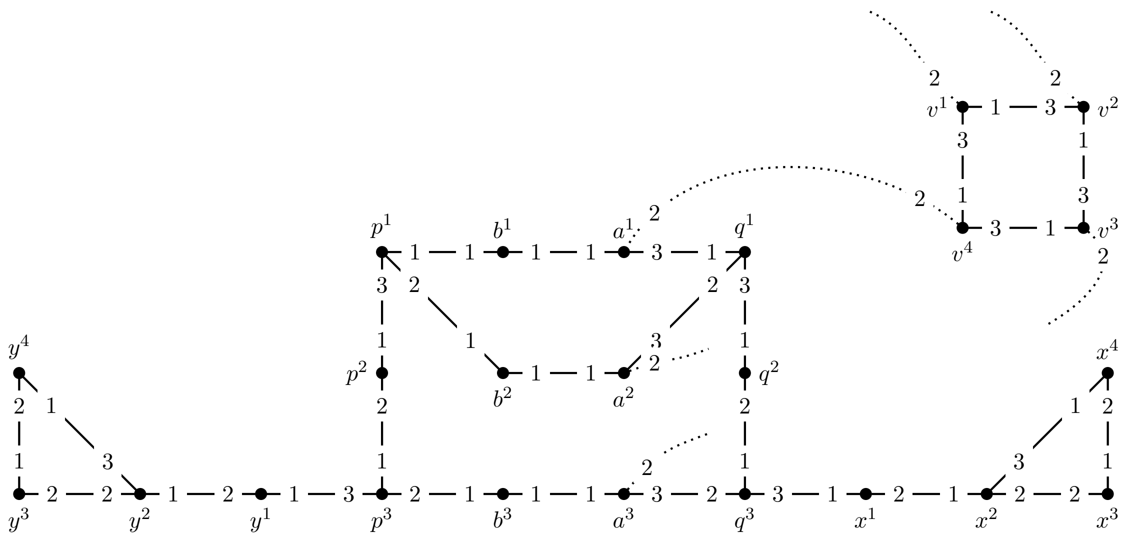


Figure 3.3: Clause and variable gadgets for 3-SRTI. The dotted edges are the interconnecting edges. We know it from edge a^1v^4 that the first literal of the corresponding clause (a^1) is the second appearance of the corresponding variable in negated form (v^4).

A variable gadget comprises the 4-cycle $\langle v^1, v^2, v^3, v^4\rangle$ with cyclic preferences. Each of these four vertices is incident to an interconnecting edge. These edges end at specific vertices of clause gadgets. Consider the variable x. Due to the properties of (2,2)-E3-SAT, x occurs twice in unnegated form, say, in clauses C_1 and C_2. Its first appearance, as the i-th literal of C_1, is represented by the interconnecting edge between v^1 and a^i of the clause gadget corresponding to C_1. Similarly, v^3 is connected to a vertex a^i, $i \in \{1, 2, 3\}$ in the clause gadget of C_2. The same variable, x, also appears twice in negated form. The variable gadgets representing those clauses are connected to v^2 and v^4, respectively. The other end vertices a^i, $i \in \{1, 2, 3\}$ of these two interconnecting edges mark where these two literals appear in their clauses.

Claim 18. *For any truth assignment satisfying B, a weakly stable matching M can be constructed in G.*

Proof. In Figure 3.4, we define two matchings, M_T and M_F, on the variable gadgets and three matchings, M_1, M_2 and M_3, on the clause gadgets.

If a variable is assigned to be true, M_T is added to M, otherwise M_F is added. Similarly, if the first literal of a clause is true, M_1 is added to M, otherwise, if the second literal is true, M_2 is added, and finally, if only the third literal is true, M_3 is added. The intuition behind this choice is that if a literal is true, then the vertex representing it in the variable gadget is matched to its best choice. On the other hand, if some literals in a clause are true, then the vertex representing the appearance of one of them in that clause is matched to its last-choice vertex.

$$
\begin{aligned}
M_T &= \{v^1v^2, v^3v^4\} & M_1 &= \{a^1q^1, b^1p^1, a^2b^2, a^3b^3, q^2q^3, p^2p^3, y^1y^2, y^3y^4, x^1x^2, x^3x^4\} \\
M_F &= \{v^1v^4, v^2v^3\} & M_2 &= \{a^2q^1, b^2p^1, a^1b^1, a^3b^3, q^2q^3, p^2p^3, y^1y^2, y^3y^4, x^1x^2, x^3x^4\} \\
& & M_3 &= \{a^3q^3, b^3p^3, a^1b^1, a^2b^2, q^1q^2, p^1p^2, y^1y^2, y^3y^4, x^1x^2, x^3x^4\}
\end{aligned}
$$

Figure 3.4: The matchings corresponding to true and false variables and to the first, second or third literal being true in a fixed clause.

We claim that no edge blocks M. Checking the edges in the clause and variable gadgets is easy. The five special matchings were designed in such a way that no edge in the gadgets blocks them. More explanation is needed regarding the interconnecting edges. Suppose one of them, a^iv^j, blocks M. Since M is a perfect matching, a^i needs to be matched to a q-vertex. Similarly, v^j has to be matched to its worst partner. While the first edge indicates that the literal represented by v^j is true in the clause, the latter edge means that the literal is false. ■

Claim 19. *For any weakly stable matching M in G, there is a truth assignment satisfying B.*

Proof. In the next three paragraphs we show that the restriction of M to any variable or clause gadget is one of the above listed special matchings, and no interconnecting edge is in M.

First of all, if vertex v is the only first choice of another vertex, then v certainly needs to be matched in M. This property is fulfilled for all vertices of all clause gadgets except for each of x^3, y^3, a^2 and a^3. Let us first study an arbitrary clause gadget. If any y^4 is matched to y^2, then y^2y^3 blocks M. Thus, y^3y^4, and similarly, x^3x^4 are part of M for all clause gadgets. Since y^2 and x^2 both need to be matched, the same can be said about y^1y^2 and x^1x^2.

Our proof for clause gadgets from this point on consists of listing all matchings covering all ten vertices that need to be matched out of the remaining twelve. We differentiate two possible cases, depending on the matching edge of p^3. In the first case, $p^3b^3 \in M$. Therefore, $p^2p^1 \in M$ too, because p^2 has to be matched. For similar reasons, $\{b^1a^1, b^2a^2, q^1q^2, q^3a^3\} \subseteq M$. This gives us matching M_3. In the second case, if p^3 is matched to p^2, then $\{b^3a^3, q^3q^2\} \subseteq M$. There are two possible matchings on the remaining six vertices: $\{p^1b^1, a^1q^1, b^2a^2\}$ and $\{p^1b^2, q^1a^2, b^1a^1\}$. These two matchings together with the lower part of the gadget form M_1 and M_2.

Since all a-vertices are matched to vertices in their clause gadgets, no interconnecting edge can be a part of M. For the variable gadgets, it is straightforward to see that M_T and M_F are the only matchings covering all vertices of the 4-cycles.

The truth assignment to B is then defined in the following way. Each variable whose gadget has the edges of M_T in M is assigned to be true, while all other variables with M_F on their gadgets are false.

All that remains is to show that this is indeed a truth assignment. Suppose that there is an unsatisfied clause C in B. Since all three of its literals are false, the three interconnecting edges dominate the matching edges at the variable gadgets. Since M was stable, these three edges are dominated by M at their other end vertex. It is only possible if a^1b^1, a^2b^2 and a^3b^3 are all in M, which never occurs in any stable matching as shown above. ■ □

Our construction guarantees that the complexity result holds even if the preference lists are either strictly ordered or comprised of a single tie of length two. Moreover, this proof also justifies that minimizing the number of blocking edges also must be computationally hard.

Corollary 3.4. MIN BP 3-SRTI *is also* NP*-hard.*

We establish an even stronger certificate for the hardness of MIN BP 3-SRTI below.

Theorem 3.5. *Unless* $\mathsf{P} = \mathsf{NP}$*,* MIN BP 3-SRTI *is not approximable within* $|V|^{1-\varepsilon}$ *for any* $\varepsilon > 0$.

Proof. Our proof is an analogous version of a similar inapproximabilty result for the almost stable Hospitals / Residents problem with couples [15]. The core idea of the proof is to gather several copies of the instance created in the previous proof together with a small instance where no stable solution exists. By doing so, we create a MIN BP 3-SRTI instance $\mathcal{I}$ in which $bp(\mathcal{I})$ is large if the Boolean formula is not satisfiable, and $bp(\mathcal{I}) = 1$ otherwise. Therefore, finding a good approximation is equivalent to finding an optimal solution.

The smallest instance not admitting any weakly stable solution is a 3-cycle with cyclic strict preferences. Aside from this, k copies of the above described instance are created to the same Boolean formula B. We have to ensure that k is large enough: let $c = \lceil 2/\varepsilon \rceil$ and $k = n^c$, where n is the number of variables in B. We use m to denote the number of clauses in B. Due to the proof of Theorem 3.3 above, this instance has either a single blocking edge or it has at least $k+1$ blocking edges.

In the rest of the proof we show that $|V|^{1-\varepsilon} \le k$. Therefore, any $|V|^{1-\varepsilon}$-approximation is an optimal matching, admitting a single blocking edge. With Inequalities (3.1)-(3.4) we give an upper bound for $|V|$. This is used in Inequalities (3.6)-(3.9) as we establish k as an upper bound for $|V|^{1-\varepsilon}$. Explanations for the steps are given as and when it is necessary after each set of inequalities.

$$\begin{aligned} |V| &= k(4n + 20m) + 3 && (3.1)\\ &= k(4n + 20\frac{4n}{3}) + 3 && (3.2)\\ &\le 32kn && (3.3)\\ &= 32n^{c+1} && (3.4)\end{aligned}$$

(3.1): $|V|$ is the number of vertices in the large MIN BP 3-SRTI instance
(3.2): $m = \frac{4n}{3}$ in MAX (2,2)-E3-SAT

(3.3): we can assume without loss of generality that $kn > 3$
(3.4): $k = n^c$
Since c was defined as $\lceil 2/\varepsilon \rceil$, the following inequality also holds.

$$\frac{c-1}{c+1} = 1 - \frac{2}{c+1} \geq 1 - \varepsilon \tag{3.5}$$

$$\begin{aligned} |V|^{1-\varepsilon} &\leq |V|^{\frac{c-1}{c+1}} && (3.6)\\ &\leq 32^{\frac{c-1}{c+1}} n^{c-1} && (3.7)\\ &\leq n^c && (3.8)\\ &= k && (3.9)\end{aligned}$$

(3.6): Inequality (3.5)
(3.7): Inequalities (3.1)-(3.4)
(3.8): $\frac{c-1}{c+1} < 1$ and we can assume without loss of generality that $n \geq 32$
(3.9): definition of k □

To complete the study of cases in MIN BP d-SRTI, we establish a positive result for instances with degree at most 2.

Theorem 3.6. MIN BP 2-SRTI *is solvable in* $\mathcal{O}(|V|)$ *time.*

Proof. We claim that for an instance $\mathcal{I}$ of MIN BP 2-SRTI, $bp(\mathcal{I})$ equals the number of odd parties in G. An *odd party* is a cycle $C = \langle v_1, v_2, ..., v_k \rangle$ of odd length, where the v_i strictly prefers v_{i+1} to v_{i-1} (addition and subtraction are taken modulo k).

Since an odd party never admits a weakly stable matching, $bp(\mathcal{I})$ is bounded by the number of odd parties by below. This bound is tight: by taking an arbitrary maximum matching in an odd party component, an almost stable solution is already reached. Now we show that a weakly stable matching M can be constructed in all other components.

Consider a component that is not an odd party. Every edge uv is added to M that would block M regardless of the rest of M. This scenario occurs if 1) $\text{rank}_u(v) = 1$ and there is no other vertex v' so that $\text{rank}_u(v') = 1$ and 2) $\text{rank}_v(u) = 1$ and there is no other vertex u' so that $\text{rank}_v(u') = 1$. Edges adjacent to these fixed edges are deleted. These two steps are iterated until the graph contains no edges to fix. Now we search for a maximum matching in each component; a path or a cycle comprising an even number of edges. Since maximum matchings cover all vertices in these components and no edge is preferred strictly by both of its end vertices, this already delivers a weakly stable solution. If the component is an odd path, selecting every second edge is sufficient to construct a stable matching. Regarding the odd cycle components, since they are not odd parties, there is at least one vertex not strictly preferred by either of its adjacent vertices. Leaving this vertex uncovered and adding a perfect matching in the rest of the cycle results in a weakly stable matching. □

3.3 Conclusion and open problems

In this chapter we gave an alternative proof for the hardness of MAX SM FREE and discussed the degree constrained version of WEAK SRT. For the latter problem we proved that WEAK 3-SRTI is already NP-complete, presented a polynomial time algorithm for WEAK 2-SRTI and an inapproximability result for MIN BP 3-SRTI.

The NP-hardness result for MAX SM FREE immediately implies that the analogously defined MAX SA FREE and MAX SF FREE problems are also computationally hard. It could be interesting to extend the 2/3-approximation algorithm of Askalidis et al. [5] to more complex instances.

4 Paths to stable allocations

The stable allocation problem is one of the broadest extensions of the stable marriage problem. In an allocation problem, edges of a bipartite graph have capacities and vertices have quotas to fill. Here we investigate the case of uncoordinated processes in stable allocation instances. In this setting, a feasible allocation is given and the aim is to reach a stable allocation by raising the value of the allocation along blocking edges and reducing it on worse edges if needed. Do such myopic changes lead to a stable solution?

In this chapter, we analyze both better and best response dynamics from an algorithmic point of view. With the help of two deterministic algorithms we show that random procedures reach a stable solution with probability one for all rational input data in both cases. Surprisingly, while there is a polynomial path to stability when better response strategies are played (even for irrational input data), the more intuitive best response steps may require exponential time. We also study the special case of correlated markets. There, random best response strategies lead to a stable allocation in expected polynomial time.

The results presented in this chapter are joint work with Martin Skutella and have been published in [29].

4.1 Introduction

Motivation. *Capacitated matching markets* without prices model various real-life problems such as, e.g., employee placement, task scheduling or kidney donor matching. Stability is probably the most widely used optimality criterion in that case.

Finding equilibria in markets that lack a central authority of control is another widely studied, challenging task. Besides modeling *uncoordinated markets*, like third-generation (3G) wireless data networks [48], selfish and uncontrolled agents can also represent modifications in coordinated markets, e.g., the arrival of a new participant or slightly changed preferences [18]. In this chapter, those two topics are combined: we study uncoordinated capacitated matching markets.

A natural extension of matching problems arises when capacities are introduced. The stable allocation problem is defined on a bipartite graph with edge capacities and quotas on vertices. The exact problem formulation and a detailed example are provided in Section 4.2.

Central planning is needed in order to produce a stable matching in SM instances with the Gale-Shapley algorithm. In many real-life situations, however, such a coordination is not available. Agents play their selfish strategy, trying to reach a best possible solution. A *path to stability* is a series of myopic operations leading to a stable solution. The intuitive picture of a myopic operation in SM is the following. If a man and a woman block a marriage scheme, then they both agree to form a couple together, even if they divorce their current partners to that end. This step may induce new blocking pairs. A sequence of such changes are made until a stable matching is reached.

Literature review. The study of uncoordinated matching processes has a long history. In the case of one-to-one matchings, two different concepts have been studied: better and best response dynamics. One side of the bipartite graph is chosen to be the *active* side. These vertices submit proposals to the *passive* vertices. According to *best response dynamics*, the best blocking edge of an active vertex is chosen to perform a myopic change. In *better response dynamics*, any blocking edge can play this role.

The core questions regarding these uncoordinated processes rise naturally. Can a series of myopic changes result in returning back to the same unstable matching? If yes, is there a way to reach a stable solution? How do random procedures behave? The first question about uncoordinated two-sided SM markets was brought up by Knuth [70] in 1976. He also gives an example of a matching problem where better response dynamics cycle. More than a decade later, Roth and Vande Vate [92] came up with the next result on the topic. They show that random better response dynamics converge to a stable matching with probability one. Analogous results for best response dynamics were published in 2011 by Ackermann et al. [4]. They show an instance in which best response dynamics cycle and give a deterministic algorithm that constructs a sequence of myopic operations reaching a stable solution in polynomial time. They also prove that the convergence time is exponential in both random cases.

Besides these works on uncoordinated SM, there is a number of papers investigating variants of it from the paths-to-stability point of view. For SR it is known that there is a series of myopic operations that leads to a stable solution, if one exists [35]. A path to stability also exists in a more application-oriented model of SM where payments, flexible salaries and productivity are present [23]. In the Hospitals / Residents problem, when couples are present, the existence of such a path is only guaranteed if the preferences are weakly responsive [69]. Weak responsiveness in the Hospitals / Residents problem with couples ensures consistence between the preferences of each partner and the couple's preference list on pairs of hospitals. In many-to-many markets, supposing we are given substitutable preferences on one side and responsive preferences on the other side, a path to stability can be found [59]. Both substitutable and responsive preferences are defined in instances where preferences are given on sets of vertices. Although many variants of SM have been studied, for the best of our knowledge no paper discusses the case of allocations (instead of matchings or b-matchings), where edges are capacitated, and thus might be partially contained in stable solutions. In this chapter we make an attempt to fill this gap in the literature.

Our contribution and structure. In the next section, the essential theoretical basis is provided: stable allocations, and better and best response modifications on such instances are defined. In Section 4.3, a special case of allocation instances are investigated. We show that although random best response processes generally run in exponential time, in the case of correlated markets, polynomial convergence is expected. Better and best response dynamics in the general case on rational input are extensively studied in Section 4.4. We describe two deterministic algorithms that generalize the result of Ackermann et al. on SM to stable allocation instances and also show algorithmic differ-

	shortest path to stability	random path to stability
best response dynamics	exponential length	converges with probability 1
better response dynamics	polynomial length	converges with probability 1

Table 4.1: Our results for a shortest and a random path to a stable allocation on instances with rational input.

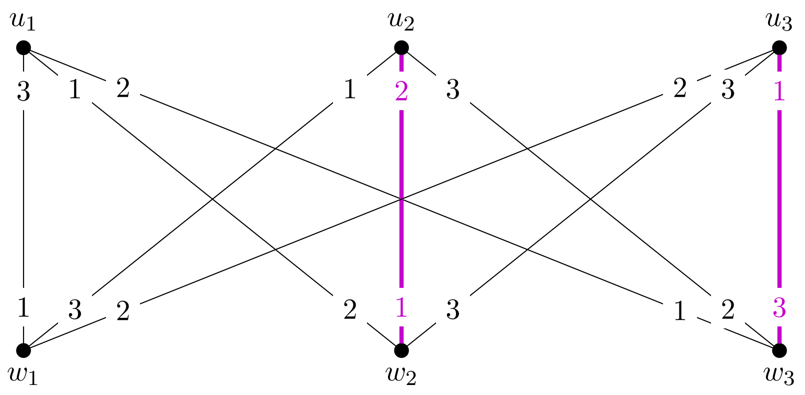


Figure 4.1: A stable marriage instance and a cycle of best response blocking edges.

ences between better and best response strategies. In the case of random procedures, convergence is shown for both strategies. Section 4.5 focuses on running time efficiency. There, a better response algorithm is presented that terminates with a stable solution in $\mathcal{O}(|V|^2|E|)$ time, even for irrational input data. A counterexample proves that such an acceleration cannot be reached for the best response dynamics. Our contribution is summarized in Table 4.1.

4.2 Preliminaries

In this section, we demonstrate that best response dynamics can cycle in SM instances, define stable allocations and extend the notion of better and best response strategies to them.

Example 4.1. *Best response cycle on an* SM *instance [4].*

Figure 4.1 demonstrates that best response dynamics can cycle in SM. Starting with the unstable colored matching $\{u_2w_2, u_3w_3\}$, and saturating the blocking edges u_1w_3, u_2w_1, u_3w_1, u_1w_2, u_2w_2, u_3w_3 in this order leads back to the same unstable matching. In each round, the chosen blocking edge is the best blocking edge of the corresponding vertex u_i.

4.2.1 Stable allocations

SM has been extended in several directions. A great deal of research effort has been spent on *many-to-one* and *many-to-many matchings*, sometimes also referred to as b-matchings. Their extension is called the *stable allocation problem*, also known as the ordinal transportation problem, since it is a direct analog of the classical cost-based transportation problem. In this problem, the vertices of a bipartite graph $G = (V, E), V = J \cup M$ have *quotas* $q : V(G) \to \mathbb{R}_{\geq 0}$, while edges have *capacities* $c : E(G) \to \mathbb{R}_{\geq 0}$. Both functions are *real-valued*, unlike the respective functions in many-to-many instances, where capacities are unit, while quotas are integer-valued. Therefore, allocations can model more complex problems, for example those in which goods can be divided unequally between agents.

In order to avoid confusion caused by terms associated with the marriage model, we call the vertices one side of the graph *jobs* and the remaining vertices *machines*. We follow the convention of denoting machines with M throughout Chapters 4 and 5, where stable allocations are discussed. Since no matching instance is studied in these two chapters, this notational convenience cannot be a source of confusion. For each machine, its quota is the maximal time spent working. A job's quota is the total time that machines must spend on the job in order to complete it. In addition, machines have a limit on the time spent on a specific job; this is modeled by edge capacities. A feasible allocation is a set of contracts where no machine is overloaded and no job is worked on after it has been completed. Note that feasibility allows jobs not to be fully completed.

Definition 4.2 (allocation)**.** *Function $x : E(G) \to \mathbb{R}_{\geq 0}$ is called an* allocation *if for every edge $e \in E(G)$ and every vertex $v \in V(G)$:*

1. *$x(e) \leq c(e)$;*
2. *$x(v) := \sum_{e \in \delta(v)} x(e) \leq q(v)$, where $\delta(v)$ is the set of edges incident to v.*

We refer to $|x| := \sum_{e \in E(G)} x(e)$ as the *size* of the allocation, while $x(e)$ is the *allocation value* on edge e. If $x(e) = 0$ for some edge $e \in E(G)$ or $x(v) = 0$ for some vertex $v \in V(G)$, then we say that e or v, respectively, is *empty* in x. To define stability we need *preference lists* as well. Analogous to SM, all vertices rank their incident edges strictly. Vertex v prefers uv to wv, if uv is ranked better on v's preference list than wv: $\text{rank}_v(uv) < \text{rank}_v(wv)$. A stable allocation instance $\mathcal{I}$ consists of four elements: $\mathcal{I} = (G, q, c, O)$, where O is the set of all preference lists.

Definition 4.3 (blocking edge, stable allocation)**.** *An allocation x is* blocked *by an edge jm if all of the following properties hold:*

1. *$x(jm) < c(jm)$;*
2. *$x(j) < q(j)$ or j prefers jm to its worst edge with positive value in x;*
3. *$x(m) < q(m)$ or m prefers jm to its worst edge with positive value in x.*

A feasible allocation is stable *if no edge blocks it.*

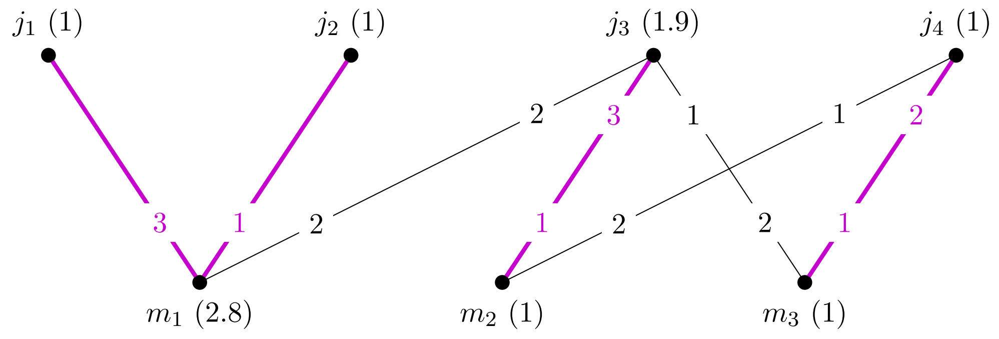


Figure 4.2: A stable allocation instance with unit capacities and a feasible, but unstable allocation, marked by colored edges.

In other words, edge jm is blocking if it is unsaturated and neither end vertex of jm has filled up its quota with at least as good edges as jm. If an unsaturated edge fulfills the second criterion, then we say that it *dominates* x at j. Similarly, if the third criterion is fulfilled, then we talk about an edge dominating x at m.

Problem 14. SA
Input: $\mathcal{I} = (G, q, c, O)$; *a bipartite graph* G, *quotas* $q : V(G) \to \mathbb{R}_{\geq 0}$, *capacities* $c : E(G) \to \mathbb{R}_{\geq 0}$ *and preference orderings of vertices over their incident edges* O.
Question: Is there an allocation not blocked by any edge?

Example 4.4. SA *instance.*

Figure 4.2 illustrates an SA instance. Just like in Chapter 2, we use the same example throughout the entire chapter to demonstrate different notions defined here. For the sake of simplicity, all edge capacities are unit. The numbers within parenthesis over and under the vertices represent the quota function. The preferences can be seen on the edges: the more preferred edges carry a better rank, i.e., a smaller number. For example, machine m_1's most preferred job is j_2, its second choice is j_3, while its least preferred, but still acceptable job is j_1. The function $x = 1$ on the colored edges and $x = 0$ on the remaining edges is a feasible allocation, since no quota or capacity constraint is violated. The unique blocking edge is easy to find: j_3m_1 blocks x, because it is unsaturated and both end vertices have free quota.

Baïou and Balinski [8] prove that stable allocations always exist. They also give two algorithms for finding them, an extended version of the Gale-Shapley algorithm and an inductive algorithm. The worst case running time of the first algorithm is exponential, but the latter one runs in strongly polynomial time. Dean and Munshi [34] speed up the polynomial algorithm using sophisticated data structures: their version runs in $\mathcal{O}(|E| \log |V|)$ time for any real-valued instance.

4.2.2 Better and best response steps for allocations

First, we provide some basic definitions and notation that we will use throughout the chapter. The instance can be written as $\mathcal{I}= (G, q, c, O, x)$, where a feasible but possibly unstable allocation x is given at the beginning. In our instance $\mathcal{I}$, jobs form the active side J, while machines M are passive players. For the sake of simplicity we denote the residual capacity $c(jm) - x(jm)$ of edge jm by $\bar{x}(jm)$ and similarly, the residual quota $q(v) - x(v)$ of vertex v by $\bar{x}(v)$. The definition of better and best response strategies is not as straightforward as it is in the matching instance with unit quotas and capacities. Here, the possible outcomes for a player are ordered lexicographically. We say that machine m prefers allocation x_1 to allocation x_2 if $x_1(j'm) > x_2(j'm)$ the for the best ranked edge $j'm$ among edges with $x_1(jm) \neq x_2(jm)$.

Although lexicographic order seems to be a natural choice, it is somewhat against the convention when discussing stable allocations. In most cases, when comparing the position of a vertex in two stable allocations, the so called *min-min criterion* is used [8]. According to this rule, the vertex prefers the allocation in which its worst positive edge is ranked better. In order to make use of such an ordering relation, each vertex has to have the same allocation value in all stable solutions. The Rural Hospitals Theorem holds forSA, but since x in $\mathcal{I}$ only becomes stable when the output is reached, it is irrelevant when comparing allocations along the path to stability. Therefore here, when studying and comparing arbitrary feasible allocations, this concept proves to be counter-intuitive. Later on, in Chapter 5 lexicographic order will again be used as a measure of optimality, because the model discussed there also fails to satisfy the Rural Hospitals Theorem. On the other hand, in Chapter 6, the conventional min-min criterion will decide which allocation is better for a vertex, because the Rural Hospitals Theorem applies to the problem tackled there.

An active player j having some blocking edges is chosen to perform a *best response step* on the current allocation x. Amongst j's blocking edges, let jm be the one ranked best on j's preference list. The aim of player j is to reach its best possible lexicographic position via increasing $x(jm)$. To this end, j is ready to allocate all its remaining quota $\bar{x}(j)$ to jm, moreover, it may reassign allocation from all edges worse than jm to jm. Thus, j aims to increase $x(jm)$ by $\bar{x}(j) + x(\text{edges dominated by } jm \text{ at } j)$. To preserve feasibility, $x(jm)$ is not increased by more than $\bar{x}(jm)$. The passive player m agrees to increase $x(jm)$ as long as it does not lose allocation on better edges. This constraint gives the third upper bound, $\bar{x}(m) + x(\text{edges dominated by } jm \text{ at } m)$. To summarize this, in a best response step $x(jm)$ is increased by the following amount.

$$\begin{aligned} A := \min\{&\bar{x}(j) + x(\text{edges dominated by } jm \text{ at } j), \bar{x}(jm), \\ &\bar{x}(m) + x(\text{edges dominated by } jm \text{ at } m)\} \end{aligned}$$

Once this A and the new $x(jm)$ is determined, j and m fill their remaining quota, then refuse allocation on their worst allocated edges, until x becomes feasible.

Better response steps are much less complicated to describe. The chosen active vertex j increases the allocation on an arbitrary blocking edge jm. Both j and m are allowed to refuse allocation on worse edges than jm. This rule guarantees that j's lexicographic situation improves and that the change is myopic for both vertices. By definition, best response steps are always better response steps at the same time. The execution of a single better response step consists of modifications on at most $|\delta(j)|+|\delta(m)|-1 \leq |V|-1$ edges.

In our example in Figure 4.2, j_3 and m_1 mutually agree to allocate value 1 to j_3m_1. If best response strategies are played, m_1 refuses 0.2 amount of allocation from j_1m_1, while j_3 reduces $x(j_3m_2)$ to 0.9. Through this step, they induce blocking elsewhere in G: now j_4m_2 blocks the new x, because m_2 lost some allocation. Thus, another myopic change would be to increase $x(j_4m_2)$, and so on. A better response step of the same vertex j_3 would be for example to increase $x(j_3m_1)$ to 1, while refusing j_3m_2 entirely. To keep feasibility, m_1 has to refuse 0.2 amount of allocation from j_1m_1.

4.3 Correlated markets

Before tackling the general paths to stability problem, we first restrict ourselves to instances with special preference profiles. In this section, we study the case of stable allocations on an uncoordinated market with correlated preferences. Later we will prove that the convergence time of random best and better response strategies is exponential on general instances. By contrast, here we show that on correlated markets, random best response strategies terminate in expected polynomial time, even in the presence of irrational data. At the end of this section we also elaborate on the behavior of better response dynamics.

Definition 4.5 (correlated market)**.** *An allocation instance is* correlated*, if there is a function $f : E(G) \to \mathbb{N}$ such that $\mathrm{rank}_v(uv) < \mathrm{rank}_v(wv)$ if $f(uv) < f(wv)$ for every $u, v, w \in V(G)$ and no two edges have the same f value.*

Correlated markets are also called *instances with globally ranked pairs* or *acyclic markets*. The latter property means that there is no cycle of incident edges such that every edge is preferred to the previous one by their common vertex. Abraham et al. [3] show that acyclic markets are correlated and vice versa. It is easy to see that every correlated one-to-one matching market admits a unique stable matching M: the edge with the lowest f-value must be in all stable matchings, therefore the edges incident to it cannot be in them, then we take the edge in the remaining graph with the lowest f-value and so on. This argument carries over to the allocation case as well, as we show in detail in the proof of Theorem 4.6. The instance depicted in Figure 4.2 is not correlated: edges $\langle j_3m_3, j_4m_3, j_4m_2, j_3m_2\rangle$ form a preference cycle. Ackermann et al. [4] were the first to prove that random better and best response dynamics reach a stable matching on correlated markets in expected polynomial time. Using a similar argumentation, we extend their result to allocation instances.

Theorem 4.6. *On correlated allocation instances with real-valued input data, random best response dynamics reach a stable solution in expected time $\mathcal{O}(|V|^2|E|)$.*

Proof. Before studying paths to stability we show that on correlated instances, the set of stable solutions has cardinality one. There is an absolute minimum of $f(jm)$. The single edge jm with this minimal f value must be in all stable allocations with value $\min\{c(jm), q(j), q(m)\}$, otherwise it is blocking. Fixing x on jm and decreasing the quotas of j and m respectively leads to another correlated allocation instance. In this instance, the stable solutions are exactly the stable solutions of the original instance without jm. This leads to an inductive algorithm that proves that there is a unique stable allocation on correlated markets. We will show that random best response dynamics reach this unique solution in expected polynomial time.

Whenever a job j with an unsaturated edge jm of an absolute minimal $f(jm)$ is chosen to submit an offer, its best response strategy is to increase x on jm. Due to this single best response operation performed by j, $x(jm) = \min\{c(jm), q(j), q(m)\}$ is reached. The probability that a vertex $j \in J$ is chosen to take the next step is at least $\frac{1}{|J|}$. As mentioned in Section 4.2.2, one best response step requires at most $\mathcal{O}(|V|)$ modifications. Thus, in order to reach $x(jm) = \min\{c(jm), q(j), q(m)\}$ on the best edge in G, $|J| \cdot |V| = \mathcal{O}(|V|^2)$ modifications are needed in expectation. After this, jm with minimal f value reached its final position in the unique stable allocation, $x(jm)$ will never be reduced, because neither j, nor m have a better neighboring edge. Thus, $x(jm)$ can be fixed, and a new minimum of f can be chosen for the same procedure as before. The number of iterations is bounded from above by the number of edges in the graph. The unique stable allocation is thus reached in $\mathcal{O}(|V|^2|E|)$ time in expectation. □

In order to establish a similar result for better response dynamics in real-valued instances, an exact interpretation of random events would be needed. In the matching case, best and better response dynamics differ exclusively in the rank of the chosen blocking edge: when playing best response strategy, the best blocking edge is chosen by an active vertex j. In contrast to this, here, better response steps differ also in the amount of modification and in the edges chosen to refuse allocation along. The first factor indicates a continuous example space.

If we assume that any better response step results in reassigning the highest possible allocation value to an arbitrary blocking edge, an analogous proof can be derived.

Theorem 4.7. *On correlated allocation instances with real-valued input data, random better response dynamics reach a stable solution in expected time $\mathcal{O}(|V|^3|E|)$.*

Proof. The only difference to the setting with best response steps is that after j is chosen, the expected time of reaching $x(jm) = \min\{c(jm), q(j), q(m)\}$ is larger. In this case, j chooses jm with probability at least $\frac{1}{|\delta(j)|}$. This implies that reaching the stable allocation value on the best edge takes $|\delta(j)| \cdot (|\delta(j)| + |\delta(m)| - 1) = \mathcal{O}(|V|^2)$ steps in expectation. In total, for all vertices $j \in J$ and all edges the algorithm takes $\mathcal{O}(|V|^3|E|)$ steps in expectation. □

4.4 Best and better responses with rational data

In this section, the case of allocations in an uncoordinated market *with rational data* is studied. As already mentioned, better and best response dynamics can cycle in such instances. We describe two deterministic methods, a better response and a best response algorithm that yield stable allocations in finite time. Our best response algorithm is by definition a better response algorithm as well, yet we present a different, better but not best response strategy in Section 4.4.1, because it can be accelerated to reach a stable solution in polynomial time, while the best response strategy cannot, as shown in Section 4.4.2. The main idea of our algorithms is to distinguish between blocking edges based on the type of blocking at the job: dominance or free quota.

A blocking edge can be of two types. Recall point 2 of Definition 4.3: if jm blocks x, then $x(j) < q(j)$ or j prefers jm to its worst edge with positive value in x. We talk about *blocking of type I* in the latter case, if jm blocks x because j prefers jm to its worst edge having positive value in x. *Blocking of type II* means that j has no allocated edge that is worse than jm, but j has not filled up its quota yet, $x(j) < q(j)$. Note that the reason of the blocking property at m is not involved when defining the two types.

4.4.1 Better response dynamics

First, we provide a deterministic algorithm (Algorithm 1 below) that constructs a finite path to stability from any feasible allocation. In the first phase of our algorithm, only blocking edges of type I are chosen to perform myopic changes along. The active vertices (jobs) choose one of their blocking edges of type I, not necessarily the best one. In all cases, withdrawal is executed along worst allocated edges. The amount of new allocation added to the blocking edge is determined in such a way that at least one edge or a vertex becomes saturated or empty. Thus, in the first phase, active vertices replace their worst edges with better ones, even if they have free quota. When no blocking edge of type I remains, the second phase starts. The allocation value is increased on blocking edges of type II such that they cease to be blocking.

The running time of our algorithm is exponential. Later, in Section 4.5 we will show that this algorithm can be accelerated such that a stable solution is reached in strongly polynomial time.

Theorem 4.8. *For every allocation instance with rational data and a given rational feasible allocation x, there is a finite sequence of better responses that leads to a stable allocation.*

The main idea of the proof is the following. We need to keep track of the change in the size of the allocation and in the lexicographic position of the active vertices simultaneously. In one step of the first phase along edge jm, either both j and m refuse edges, thus, the size of the allocation $|x| = \sum_{j \in J} x(j)$ decreases, or only j does so, leaving $|x|$ unchanged and improving j's situation lexicographically. Since both procedures are monotone and the second one does not impair the first one, the first phase terminates. Termination of the second phase is implied by the fact that passive vertices improve their lexicographic situation in each step. The technical details of this proof sketch are presented as Claims 20 and 21.

Recall our example in Figure 4.2. The unique blocking edge j_3m_1 is of type I, because j_3, its active vertex, prefers edge j_3m_1 to its worst allocated edge j_3m_2.

In the first phase, the jobs propose along *arbitrary* blocking edges of type I. We will show that this process ends with an allocation where no job has a blocking edge of type I. In the second phase, the jobs propose along their *best* blocking edges of type II. Later we will see that during this phase until termination, no job gets a blocking edge of type I. A pseudocode is provided after the description of both phases.

First phase. In one step, an arbitrary blocking edge jm of type I is chosen. Both end vertices, j and m may refuse some allocation along worse edges when increasing x on jm. Job j has a *refusal pointer* $r(j)$ that denotes the worst edge allocated to j, if any exists. Similarly, $r(m)$ stands for the worst currently allocated edge of m. A step of Phase I consists of two or three operations, each along $jm, r(j)$ and possibly along $r(m)$. Two operations take place, if m has not filled up its quota yet. In this case, $x(r(j))$ is decreased by $A := \min\{x(r(j)), \bar{x}(jm), \bar{x}(m)\}$. At the same time, $x(jm)$ is increased by the same amount. Depending on which expression is the minimal one, edge $r(j)$ becomes empty or jm becomes saturated or m fills up its quota. Note that $r(m)$ plays no role because m does not refuse any allocation. In the remaining case, if m has a full quota, three operations take place, since m has to refuse some allocation. The amount of allocation we deal with is now $A := \min\{x(r(j)), \bar{x}(jm), x(r(m))\}$. The allocation on the blocking edge jm will be increased by A, on the other two edges it will be decreased by A, until one of them becomes empty or saturated. We emphasize that whenever a job j with free quota adds a new edge better than its worst allocated edge to x, it withdraws some allocation from the worst edge.

We return to our example again. It has already been mentioned that the unique blocking edge j_3m_1 is of type I. The refusal pointer $r(j_3)$ is j_3m_2. Since m_1 has not filled up its quota yet, its refusal pointer j_1m_1 is irrelevant at the moment. Due to the same reason, two operations take place. We augment with $\min\{x(j_3m_2), \bar{x}(j_3m_1), \bar{x}(m_1)\} = 0.8$ amount of allocation. After this operation, $x(j_3m_1) = 0.8, x(j_3m_2) = 0.2$, and j_3m_1 is still a Phase I blocking edge. Since $x(m_1) = q(m_1)$ holds now, three operations are executed with $A = \min\{x(j_3m_2), \bar{x}(j_3m_1), x(j_1m_1)\} = 0.2$. Now j_3m_1 is saturated, hence it ceases to be blocking. During the first operation, j_4m_2 became blocking of type I, because m_2 lost allocation. In the next step, one unit of allocation is reallocated to j_4m_2 from j_4m_3. But $j_3m_3$then becomes blocking of type I, and so on.

Claim 20. *Phase I terminates in finite time.*

Proof. We use the following potential function in order to show that the process does not cycle:

$$\Theta(x) := \sum_{j \in J} \sum_{jm \in E(G)} x(jm) \operatorname{rank}_j(jm)$$

Recall that $\operatorname{rank}_j(jm)$ stands for the rank of jm on j's preference list. The smaller $\operatorname{rank}_j(jm)$ is, the better is m for j. The expression above is bounded for any feasible allocation x:

$$0 \le \Theta(x) \le |J| \cdot \max_{jm \in E(G)} c(jm) \cdot \max_{j \in J} |\delta(j)|.$$

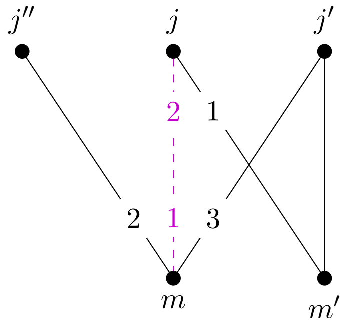


Figure 4.3: Edges affected by one myopic operation along the blocking edge jm of type II.

We will show that $\Theta(x)$ decreases in each step of the procedure. The process terminates if the amount of decrement is always greater than a fixed positive constant. If all data are rational, this is guaranteed.

Considering the potential function, we need to keep track of those two jobs that proposed or got refused, since the allocation of all other jobs remains the same, thus their contributions to the summations of $\Theta(x)$ do not change.

As mentioned above, a step consists of either two or three edges changing their value in x. In the first case, when only two edges change their value in x, there is only one job j that modifies its contribution. Thus $\Theta(x)$ decreases, because some allocation will move from a less preferred edge to jm. In the second case, where three edges are involved, there is a job j that improves its lexicographic position, and another job j' that loses allocation. The effect of the first change at j is just as above, $\Theta(x)$ decreases. Losing allocation for j' also decreases $\Theta(x)$, since $x(j')$ decreases. ■

Second phase. In the second phase, we are allowed to increase $x(j)$. When improving the allocation along a blocking edge jm of type II, m may refuse some allocation, but j cannot, since the reason of blocking is that j has not filled up its quota yet. Thus, we do not need the pointer $r(j)$ any more. One step consists of changes along one edge if $x(m) < q(m)$, or along two edges otherwise. If m has not filled up its quota yet, then we simply assign as much allocation to jm as possible without $x(j), x(m)$ and $x(jm)$ exceeding $q(j), q(m)$ and $c(jm)$, respectively. If m has to refuse something from a job j' in order to accept better offers from j, we improve m's position until $j'm$ becomes empty or jm becomes saturated or j gets its quota filled up.

Claim 21. *No step in Phase II can induce a blocking edge of type I.*

Proof. One step in Phase II leaves all vertices but j, m and the possibly refused j' unchanged. Thus, if there is a blocking edge of type I after the modification, it must be incident to one of those vertices. The three cases, illustrated in Figure 4.3, are the following.

- Edge $j''m$ blocks x. The position of m became lexicographically better, thus, no new blocking edge incident to m was introduced. The existing blocking edges $j''m$ of type II cannot become of type I, because j'''s position remained unchanged.

- Edge jm' (or jm) blocks x. The only change at j is that $x(jm)$ increases, thus, j also improves its lexicographic position. Thus, no new blocking edge incident to j appeared. Blocking edges of type II can change their type of blocking only if j increased its allocation on a worse edge. But this cannot happen since we chose the best blocking edge jm in Phase II.
- Edge $j'm'$ (or $j'm$) blocks x. The only change in j''s neighborhood is that $x(j'm)$ decreases. After this step, consider an unsaturated edge $j'm'$ preferred by j' to its worst allocated edge. Since no machine worsens its lexicographic position in Phase II, if $j'm'$ dominates the new allocation x, it already dominated the previous allocation. Thus, $j'm'$ must have been a blocking edge of type II prior to the modification and thus remains of type II.

We have argued that once Phase II has started, Phase I can never return. ■

The last step ahead of us is to show that Phase II may not cycle. But this follows from the fact that in each step exactly one machine strictly improves its lexicographic situation, while all other machines maintain the same allocation as before. In case of a rational input, this improvement is bounded from below, thus, the second phase of the algorithm terminates.

With this we finished the proof of Theorem 4.8.

Algorithm 1 Two-phase better response algorithm

```
while $\exists j \in J$ with a blocking edge of type I do
    IMPROVEMENT_I($j$)
end while
while $\exists j \in J$ with a blocking edge of type II do
    IMPROVEMENT_II($j$)
end while
```

The duration of both phases strongly depends on the capacities and quotas. The examples in Figure 4.4 show two bad instances. The capacity is N on all edges, where N is an arbitrarily large integer. Quotas are marked above and below the vertices. The initial allocation for Phase I is N on j_1m_1 and on j_2m_2 and zero on the remaining two edges. The first phase performs N augmenting steps along the same cycle. Phase II terminates after N iterations in the second instance, starting with the empty allocation.

This algorithm also proves an important result regarding rational random better response processes. If the input is rational (there is a smallest positive number that can be represented as a linear integer combination of all data), it is clearly worthwhile to restrict the set of feasible better response modifications to the ones that reassign a multiple of this unit. For this reason, the set of reachable allocations is finite and they can

```
procedure IMPROVEMENT_I($j$)
    $jm \leftarrow$ blocking edge of type I of $j$
    if $x(m) < q(m)$ then
        $A := \min\{x(r(j)), \bar{x}(jm), \bar{x}(m)\}$
        $x(r(j)) := x(r(j)) - A$
        $x(jm) := x(jm) + A$
    else
        $A := \min\{x(r(j)), \bar{x}(jm), x(r(m))\}$
        $x(r(j)) := x(r(j)) - A$
        $x(jm) := x(jm) + A$
        $x(r(m)) := x(r(m)) - A$
    end if
end procedure
```

```
procedure IMPROVEMENT_II($j$)
    $jm \leftarrow$ best blocking edge of type II of $j$
    if $x(m) < q(m)$ then
        $A := \min\{\bar{x}(jm), \bar{x}(j), \bar{x}(m)\}$
        $x(jm) := x(jm) + A$
    else
        $A := \min\{x(r(m)), \bar{x}(jm), \bar{x}(j)\}$
        $x(jm) := x(jm) + A$
        $x(r(m)) := x(r(m)) - A$
    end if
end procedure
```

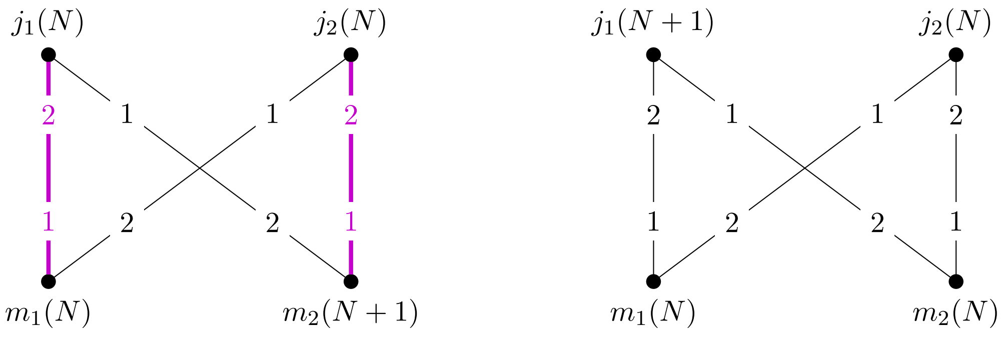


Figure 4.4: Worst-case instances for our better response algorithm. On the graph on the left hand-side, Phase I cycles along $\langle j_1m_2, j_2m_2, j_2m_1, j_1m_1\rangle$ N times. On the second instance, Phase II first assigns N amount of allocation to edges j_1m_2 and j_2m_1 and then cycles N times along $\langle j_1m_1, j_2m_1, j_2m_2, j_1m_2\rangle$.

be seen as states of a discrete time Markov chain. Our algorithm proves that from any state there is a finite path to an absorbing state with a positive probability.

Theorem 4.9. *In the rational case, random better response strategies terminate with a stable allocation with probability one.*

Polynomial time convergence cannot be shown for random better response strategies, since they need exponential time to converge in expectation even in matching instances [4].

4.4.2 Best response dynamics

In this subsection, we derive analogous results for best response modifications to the ones established for better response strategies. The main difference from the algorithmic point of view is that instances can be found in which no series of best response strategies terminate with a stable solution in polynomial time. A simple example shown on the right in Figure 4.4 resembles the instance given by Baïou and Balinski [8] to prove that the Gale-Shapley algorithm requires exponential time to terminate in stable allocation instances. Let G be a complete bipartite graph on four vertices, with quota $q(j_1) = N+1, q(j_2) = q(m_1) = q(m_2) = N$ and initial allocation $x(j_1m_1) = x(j_2m_2) = N$ for an arbitrary large number N. If the preference profile is chosen to be cyclic, such that $\text{rank}_{j_1}(m_1) = \text{rank}_{j_2}(m_2) = \text{rank}_{m_1}(j_2) = \text{rank}_{m_2}(j_1) = 2$, the unique series of best response steps consists of $2N$ operations. A path of exponential length to stability can still be found.

Theorem 4.10. *For every allocation instance with rational data and a given rational feasible allocation x, there is a finite sequence of best responses that leads to a stable allocation.*

Proof. Similar to our method for better response strategies, we prove that there is a two-phase algorithm that terminates with a stable solution.

All blocking edges we take into account are best blocking edges of their job j. Depending on their rank compared to j's worst allocated edge $r(j)$, they are either of type I or type II. A job j's best blocking edge jm is

- of type I(a), if $\text{rank}_j(jm) < \text{rank}_j(r(j))$ and $\bar{x}(j) < \min\{\bar{x}(jm), \bar{x}(m) + x(\text{edges dominated by } jm \text{ at } m)\}$;
- of type I(b), if $\text{rank}_j(jm) < \text{rank}_j(r(j))$ and $\bar{x}(j) \geq \min\{\bar{x}(jm), \bar{x}(m) + x(\text{edges dominated by } jm \text{ at } m)\}$;
- of type II, if $\text{rank}_j(jm) \geq \text{rank}_j(r(j))$.

The intuitive interpretation of the grouping above is given by the steps that we need to execute when jm is chosen to perform a best response operation. If jm is of type I(b), then jm can be saturated without any refusal made by j, since j has sufficient free quota. On the other hand, if j agrees to reduce $x(r(j))$ in order to accommodate more allocation on jm, then jm is a blocking edge of type I(a). The remaining case occurs when jm is worse than $r(j)$, that is, j accepts $\bar{x}(m)$ allocation from m. In this case, no rejection is called by j.

In Phase I, only best blocking edges of type I(a) and I(b) are selected. Then, when only type II blocking edges remain, Phase II starts. In order to prove finite termination, we introduce two potential functions, $\Theta(x)$ and $\Psi(x)$. When proving termination of the first phase, both of them are used, while the second phase is discussed by analyzing the behavior of $\Psi(x)$ only.

The first function, $\Theta(x)$ comprises two components. The first component is the sum consisting of the rank of refusal pointers at jobs. The second term is a sum consisting of the allocation value of refusal pointers at jobs. When we say that $\Theta(x)$ decreases, it is meant in the lexicographic sense. The second function, $\Psi(x)$ is a set of $|M|$ vectors, each of them corresponding to a machine. Each vector contains $|\delta(m)|$ entries, defined as $x(jm)$ for all $j \in J$, ordered as they appear in m's preference list. We denote these vectors by $lex(m)$, because $lex(m)$ increases lexicographically if and only if the lexicographic position of m improves. When we say that $\Psi(x)$ decreases we mean that at least one vector in it increases lexicographically and no vector decreases lexicographically. This also implies that we could add up the i-th elements of these vectors and follow the lexicographic increment of the resulting vector. We choose not to do so for intuitive reasons, but the reader can also think of $\Psi(x)$ as a single vector of $\max_{m \in M} \deg(m)$ scalar components.

$$\Theta(x) := (\Theta_1(x), \Theta_2(x)) := \left(\sum_{j \in J} \text{rank}_j(r(j)), \sum_{j \in J} x(r(j)) \right)$$

$$\Psi(x) := -\left(lex(m_1), lex(m_1), ..., lex(m_{|M|}) \right)$$

Claim 22. *The best response step of job j along edge jm of type I(a) decreases $\Theta(x)$.*

Proof. Due to the type-defining characteristics listed above, there is a rejection on $r(j)$. If $x(r(j))$ becomes 0 through this step, then $\Theta_1(x)$ decreases, while $\Theta_2(x)$ might increase. Otherwise, if $x(r(j)) > 0$ holds even after executing the step, $\Theta_1(x)$ remains unchanged, but $\Theta_2(x)$ decreases. Any other decrement in x, such as allocation refused by m on $r(j')$ for some $j' \neq j$ can only further decrease both components of $\Theta(x)$. ■

Claim 23. *The best response step of job j along edge jm of type I(b) decreases $\Psi(x)$ and does not increase $\Theta(x)$.*

Proof. Since j does not reject any allocation, $x(r(j))$ remains unchanged. If any other $r(j')$ for some $j' \neq j$ is affected, $\Theta(x)$ is decreased. The only machine whose position changes is m itself: it clearly improves its lexicographic position, thus one component of $\Psi(x)$ decreases, while the remaining vectors remain unchanged. ■

For any rational input data, the changes in $\Theta(x)$ or $\Psi(x)$ in each round are bounded from below. Since both functions have an absolute minimum, Phase I terminates in finite time.

Claim 24. *The best response step of job j along edge jm of type II decreases $\Psi(x)$. Moreover, no edge becomes blocking of type I(a) or I(b).*

Proof. During the second phase, no machine loses allocation, thus, their lexicographic position cannot worsen. In addition, for the machine of the current blocking edge jm, $lex(m)$ improves. This also implies that no edge $j'm'$ dominates x at m' that has not already dominated it before the myopic change. Moreover, edges that lost allocation during that step are the worst-choice edges of j, hence they cannot be blocking of type I(a) or I(b). If there is an edge $j'm'$ that became blocking of type I(a) or I(b), then it is better than the worst edge in x at j'. These edges were already unsaturated before the last step and also already dominated x at both end vertices. This contradicts the fact that best blocking edges are chosen in each step. ■ □

The same arguments as above, in Theorem 4.9, imply the result on random procedures.

Theorem 4.11. *In the rational case, random best response strategies terminate with a stable allocation with probability one.*

4.5 Irrational data - a strongly polynomial algorithm

In our previous section, we relied several times on the fact that in each step, x is changed with values greater than a specific positive lower bound. When irrational data are present, e.g., q, c or x are real-valued functions, this can no longer be guaranteed. Hence, our arguments for termination are not any more valid. Moreover, both our algorithms require exponentially many steps to terminate. In this section, we describe a fast version of our two-phase better response algorithm that terminates in polynomial time with a stable allocation for irrational input data as well. We also give a detailed proof of correctness for the first phase and show a construction with which all Phase II steps can be interpreted as Phase I operations on a slightly modified instance.

As usual in graph theory, an *alternating path* with respect to an allocation x is a sequence of adjacent edges that are saturated in x and of those that are unsaturated in x in an alternating manner.

4.5.1 Accelerated first phase

The algorithm and the proof of its correctness can be outlined in the following way (see also Algorithm 2 below). A helper graph is built in order to keep track of edges that may gain or lose some allocation. A potential function is also defined, it stores information about the structure of the helper graph and the degree of instability of the current allocation. In the helper graph we are looking for walks to augment along. The amount of allocation we augment with is specified in such a way that the potential function decreases and the helper graph changes. When using walks instead of proposal-refusal triplets, more than one myopic operation can be executed at a time. Moreover, we also keep track of consequences of locally myopic improvements. For example, we spare running time by avoiding reducing allocation on edges that later become blocking anyway.

First, we elaborate on the structure of the helper graph, define alternating walks and specify the amount of augmentation. The method, the proof of correctness, the pseudocode and a example execution are all described in detail here.

Helper graph

Recall that our real-valued input $\mathcal{I}$ consists of a stable allocation instance (G, q, c, O) and a feasible allocation x. First, we define a helper graph $H(x)$ on the same vertices as G. This graph is dependent on the current allocation x and will be changed whenever we modify x. The edge set of $H(x)$ is partitioned into three disjoint subsets. The first subset $\mathcal{P}$ is the set of Phase I blocking edges. Each job j that has at least one edge with positive x value, also has a worst allocated edge $r(j)$. When a myopic change is made, jobs tend to reduce x along exactly these edges. These *refusal pointers* form $\mathcal{R}$, the second subset of $E(H(x))$. We also keep track of edges that are currently not of blocking type I, but later on they may enter set $\mathcal{P}$. This last subset $\mathcal{P}'$ consists of edges that may become blocking of type I after some myopic changes. An edge $jm \notin \mathcal{P}$ has to fulfill three criteria in order to belong to $\mathcal{P}'$:

1. $c(jm) > x(jm)$;
2. m has at least one refusal edge, i.e., $\delta(m) \cap \mathcal{R} \neq \emptyset$;
3. $jm >_j r(j)$.

Such an edge immediately becomes blocking of type I if m loses allocation along one of its refusal edges. Edges in $\mathcal{P}'$ are called *possibly blocking edges*, the set $\mathcal{P} \cup \mathcal{P}'$ forms the set of *proposal edges*. Note that a job j may have several edges in $\mathcal{P}$ and $\mathcal{P}'$, but at most one in $\mathcal{R}$. Moreover, if j has a proposal edge in $H(x)$, it also has an edge in $\mathcal{R}$. Regarding the machines, if m has a $\mathcal{P}'$-edge, it also has an $\mathcal{R}$-edge. Note that $(\mathcal{P} \cup \mathcal{P}') \cap \mathcal{R} = \emptyset$, because both $\mathcal{P}$ and $\mathcal{P}'$ per definition comprises edges that are ranked better by j than $r(j)$. The following lemma provides an additional structural property of $H(x)$.

Lemma 4.12. *If $jm \in \mathcal{P}$ and $j'm \in \mathcal{P}'$, then $\text{rank}_m(jm) < \text{rank}_m(j'm)$.*
That is, blocking edges are preferred to possibly blocking edges by their common machine m.

Proof. Since $jm \in \mathcal{P}$ is a blocking edge of type I, jm dominates x at m. If the statement is false, then $\text{rank}_m(jm) > \text{rank}_m(j'm)$ for some unsaturated edge $j'm$ that is better than the worst allocated edge of j'. Then also $j'm$ dominates x at m. This, together with the first and last properties of possibly blocking edges implies that $j'm \in \mathcal{P}$. □

Once again we return to our example shown in Figure 4.2. The only blocking edge j_3m_1 alone forms $\mathcal{P}$. The set $\mathcal{R}$ contains all four edges with positive allocation value: j_1m_1, j_2m_1, j_3m_2 and j_4m_3. Edges j_3m_3 and j_4m_2 are possibly blocking. Thus, in this case, $G = H(x)$.

Alternating walks

Our algorithm performs augmentations along alternating walks, so that the allocation value of the refusal edges decreases, while the value of proposal edges increases. This is done in such a way that $\mathcal{R}$, $\mathcal{P}$ or $\mathcal{P}'$ (and thus, $H(x)$) changes. The main idea behind these operations is the same we used in our the proof of Theorem 4.8: reassigning allocation to blocking edges from worse edges, such that the procedure is monotone. The difference between this method and the one presented in Section 4.4.1 is that while our first algorithm tackles a single blocking edge in each step, here we deal with several blocking edges (forming the alternating walk) at once.

When constructing the alternating proposal-refusal walk ρ to augment along, the following rules have to be obeyed:

1. The first edge jm_1 is a $\mathcal{P}$-edge.
2. $\mathcal{P}$ and $\mathcal{P}'$-edges are added to ρ together with the refusal edge they are incident with on the active side.
3. Machines choose their best $\mathcal{P}$ or $\mathcal{P}'$-edge.

4. Walk ρ ends 1) at m if m has no proposal edge or 2) at j or m if it is already on ρ.

As long as there is a blocking edge of type I, the first edge jm_1 of such a walk can always be found. Lemma 4.12 guarantees that point 3 is not violated by this $jm_1 \in \mathcal{P}$. After taking $r(j)$, all that remains is to continue on best proposal edges of machines and refusal edges of jobs they end at. Since $H(x)$ is a finite graph, either of the cases listed in point 4 will appear. According to these rules, proposal-refusal edge pairs are added to the current path until 1) there is no pair to add or 2) the path reaches a vertex already visited. In the first case, ρ is a path. In the latter case, ρ is a union of a path and a cycle, connected at a single vertex. This vertex is the last vertex listed on ρ, where our method halts observing point 4. Such a walk ρ can be a single cycle as well.

Before elaborating on the amount of augmentation, we emphasize that ρ is just a *subset* of the set of edges whose x value changes during an augmentation step. The goal is to reassign allocation from refusal edges to blocking edges, until a stable solution is derived. Naturally, on an alternating walk, refusal edges lose the same amount of allocation that proposal edges gain. However, except if augmentations are performed along a single cycle, the first machine m_1 on ρ gains allocation in total (and the last machine on ρ loses allocation). In order to preserve feasibility, m_1 might have to refuse allocation on edges not belonging to ρ. The exact amount of these refusals is discussed later, together with the amount of augmentation along ρ. Since no other vertex gains allocation in an augmentation step, feasibility cannot be violated elsewhere. Thus, these are the only edges not on ρ that need to be modified.

By contrast, if the augmentation is performed along a cycle C, refusals only happen on $r(j) \in C \cap \mathcal{R}$ edges. Even if the machine m_1 that started C has a full quota, it does not need to refuse any allocation, since $x(m_1)$ remains unchanged during the augmentation. Note that executing several local myopic steps greedily, like in our first algorithm (Algorithm 1), would lead to a different output. A simple example for that can be seen on a slightly modified version of the first instance in Figure 4.4, depicted in Figure 4.5.

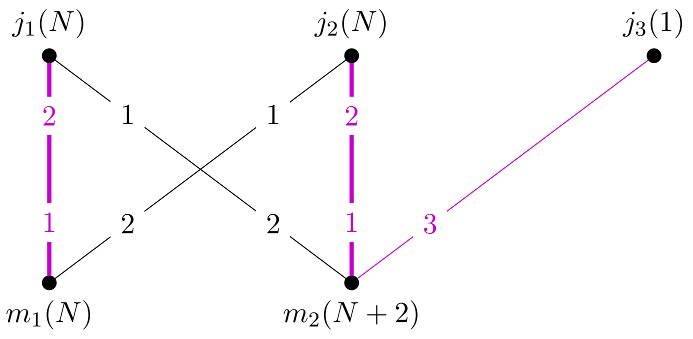


Figure 4.5: An example for a cycle augmentation in the accelerated better response algorithm. In the accelerated version, j_3m_2 remains intact, while in the step-by-step version (Algorithm 1) it gets deleted and then added again.

Let us assume that $q(m_2) = N + 2$ and m_2 has an edge j_3m_2 ranked third, where $c(j_3m_2) = 1$. Let us start with the allocation $x(j_1m_1) = x(j_2m_2) = N, x(j_3m_2) = 1$. Edge $j_1m_2 \in \mathcal{P}$, so we can start the walk at m_1, augment along $C = \langle m_2j_1, j_1j_1, m_1j_1, j_2m_2\rangle$ with allocation value 1 and arrive at a stable solution. On the other hand, our step-by-step better response algorithm presented in Section 4.4.1 would first reject j_3m_1 fully, augment along the same cycle and, in Phase II, add j_3m_1 again.

It is easy to see that the removal of edges like j_3m_1 can cause more superfluous rounds if the instance is more complex. Generally speaking, here we avoid m_1 refusing edges, knowing that it loses allocation later. As a result of that, m_1 would go under its quota, and would possibly create new blocking edges. Both strategies are better response, the difference is that our second algorithm keeps track of changes made as a consequence of a myopic operation.

Amount of augmentation

Once ρ is fixed, the amount of allocation A to augment with has to be determined. It must be chosen so that 1) a feasible allocation is derived and 2) at least one refusal edge becomes empty or at least one proposal edge leaves $\mathcal{P} \cup \mathcal{P}'$. These points guarantee that $H(x)$ changes. To fulfill these two requirements, the minimum of the following terms is determined.

1. Allocation value on refusal edges along ρ: $x(r(j))$, where $r(j) \in \rho \cap \mathcal{R}$.
2. Residual capacity on proposal edges along ρ: $\bar{x}(p)$, where $p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')$.
3. If ρ is not a single cycle, m_1 may refuse a sufficient amount of allocation such that jm_1 does not become saturated, but it stops dominating x at m_1. In this case, the residual quota of m_1 must be filled up and, in addition, the sum of allocation values on edges worse than jm_1 must be refused. With this, jm_1 becomes the worst allocated edge of a full machine. Until reaching this point, jm_1 may gain $\bar{x}(m_1) + x(\text{edges dominated by } jm_1 \text{ at } m_1)$ amount of allocation in total.

To summarize, we augment with

$$A := \min\{x(r(j)), \bar{x}(p) | r(j) \in \rho \cap \mathcal{R}, p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')\}$$

if ρ is a cycle, because the last case with the starting vertex m_1 does not occur. Otherwise, the amount of augmentation is

$$A := \min\{x(r(j)), \bar{x}(p), \bar{x}(m_1) + x(\text{edges dominated by } jm_1 \text{ at } m_1) | \\ r(j) \in \rho \cap \mathcal{R}, p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')\}.$$

Algorithm 2 Accelerated Phase I

```
while $|\mathcal{P}| > 0$ do
    FindWalk($H(x)$)
    if $\rho$ is a cycle then
        AugmentCycle($\rho$)
    else
        AugmentWalk($\rho$)
    end if
    update $\mathcal{R}$, $\mathcal{P}$, $\mathcal{P}'$
end while
```

```
procedure FindWalk($H(x)$)
    $i := 1$, $\rho := \emptyset$, find any $m_1 \in M$ with a $\mathcal{P}$-edge
    while $m_i$ has a best proposal edge $j_i m_i$ and $m_i \notin \rho$ do
        if $j_i \notin \rho$ then
            $\rho := \rho \cup \{j_i m_i\} \cup \{r(j_i)\}$
            $j_i m_{i+1} := r(j_i), i := i + 1$
        else
            $\rho := \rho \cup \{j_i m_i\}$
        end if
    end while
end procedure

procedure AugmentCycle($\rho$)
    $A := \min\{x(r(j)), \bar{x}(p) | r(j) \in \rho \cap \mathcal{R}, p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')\}$
    for $p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')$ do
        $x(p) := x(p) + A$
    end for
    for $r(j) \in \rho \cap \mathcal{R}$ do
        $x(r(j)) := x(r(j)) - A$
    end for
end procedure

procedure AugmentWalk($\rho$)
    $A := \min\{x(r(j)), \bar{x}(p), \bar{x}(m_1) + x(\text{edges dominated by } j_1 m_1 \text{ at } m_1) | r(j) \in \rho \cap \mathcal{R}, p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')\}$
    if $A - \bar{x}(m_1) > 0$ then
        $m_1$ refuses $A - \bar{x}(m_1)$ allocation from its worst edges
    end if
    for $p \in \rho \cap (\mathcal{P} \cup \mathcal{P}')$ do
        $x(p) := x(p) + A$
    end for
    for $r(j) \in \rho \cap \mathcal{R}$ do
        $x(r(j)) := x(r(j)) - A$
    end for
end procedure
```

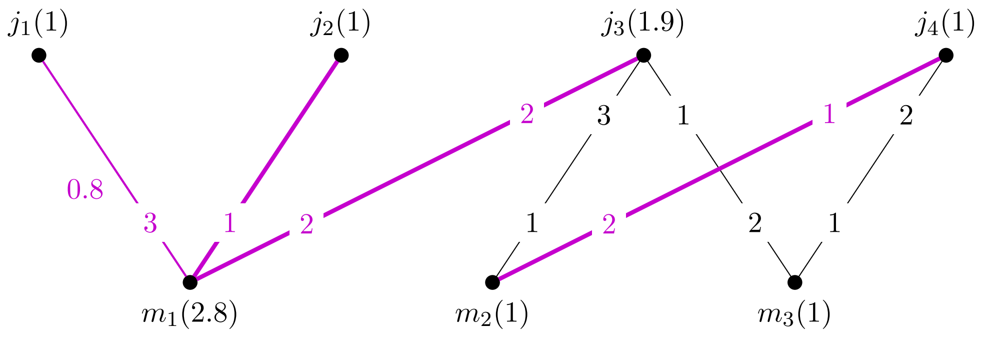


Figure 4.6: After the first round of the accelerated Phase I algorithm.

Recall the example instance in Figure 4.2 again. Checking both proposal and both refusal edges on $\rho = \langle m_1 j_3, j_3 m_2, m_2 j_4, j_4 m_3 \rangle$, the residual capacity of m_1, and the allocation on m_1's worse edges, one can compute that $A = 1$. Thus, allocation x shown in the figure above is obtained after the first augmentation. Edge $j_3 m_1$ leaves $\mathcal{P}$, and $j_3 m_3$ enters it. The set of refusal edges consists of all edges with positive allocation value. $\mathcal{P}'$ is empty. In the second round, ρ is easy to find: it is $\langle m_3 j_3, j_3 m_1 \rangle$. After reassigning allocation of value 1 to $j_3 m_3$, Phase I ends. The allocation derived (see also Figure 4.6) is not yet stable: $j_1 m_1$ and $j_3 m_2$ block it, but they are both of type II.

The second phase of our method can be interpreted as the execution of the first phase on a modified instance. The modification needed consists of introducing a dummy job and swapping the roles of the active and passive sides, as described in detail below.

In total, the algorithm performs $\mathcal{O}(|V||E|)$ rounds, each of them needs $\mathcal{O}(|V|)$ running time. Thus, it runs in $\mathcal{O}(|V|^2|E|)$ time. For a detailed proof of correctness and running time computation, see the proof below.

Theorem 4.13. *For every real-valued allocation instance and given feasible allocation, there is a sequence of better responses leading to a stable allocation in $\mathcal{O}(|V|^2|E|)$ time.*

Proof. **Potential function.** We show with the help of the following multicriteria potential function that the procedure is monotone and finite:

$$\Theta(x) := (\Theta_1(x), \Theta_2(x)) :=$$

$$\left(\sum_{j \in J} \text{rank}_j(r(j)), \quad -\sum_{m \in M} \text{rank}_m(\text{best proposal edge at } m) \right).$$

If $x(j) = 0$, then $\text{rank}_j(r(j))$ can be interpreted as a large number, for example as $|M| + 1$. In lack of proposal edges, the expression in the second component can also be interpreted as a large constant, for example as $|J| + 1$. In order to keep both terms decreasing, a minus sign is added to the second expression. When function $\Theta(x)$ decreases, it does so in the lexicographical sense.

Each round of our algorithm consists of two operations that change x: procedure augment and update. First, we prove that $\Theta(x)$ lexicographically strictly decreases after each augmentation. If no job receives a worse allocated edge than its current edges, and, in addition, at least one job loses its worst allocated edge entirely, then Θ_1 strictly decreases. If no machine receives a proposal edge better than all of its current proposal edges and at least one machine loses its best proposal edge, then the second component Θ_2 decreases. As a second step, we will see that the update operation never increases $\Theta(x)$.

Since $\Theta(x)$ is a bounded, integer-valued function, any procedure that modifies it monotonically, is finite. Later, we elaborate on the running time of our algorithm.

Augmentation. As mentioned above, our goal here is to show that each augmentation step decreases $\Theta(x)$. The amount of allocation we augment with depends only on the extreme points of the min function. Recall the three points we listed when defining the amount of augmentation.

1. $x(r(j))$
   The worst allocated edge of j becomes empty, while $x(j)$ remains unchanged, hence $\Theta_1(x)$ decreases.

2. $\bar{x}(p)$
   If one of the proposal edges reaches its capacity, it stops being blocking. Since it was the best blocking edge of its machine, $\Theta_2(x)$ decreases.

3. In case of walks: $\bar{x}(m_1) + x(\text{edges dominated by } jm_1 \text{ at } m_1)$
   The first blocking edge on ρ, m_1's best proposal edge ceases to dominate x at m_1, hence $\Theta_2(x)$ decreases.

Since the proposal and refusal pointers are monotone, no job receives a worse allocated edge and no machine receives a better proposal edge. Having shown that $\Theta(x)$ decreases every time when either of the augmenting procedures are called, all that remains to be shown is that the second operation, updating $H(x)$, does not destroy this monotonicity. The vertex set of $H(x)$ is fixed. In the following, we study the three subsets of $E(H)$ separately and show that updating them never increases either component of $\Theta(x)$.

Update $\mathcal{R}$

Claim 25. *During the accelerated Phase I algorithm, $r(j) \in \mathcal{R}$ moves monotonically on j's preference list, always pointing to a better machine.*

Proof. Suppose that there is a refusal pointer that moves to a worse edge. Since $r(j)$ is always the worst allocated edge of j, this implies that j increased x along an edge worse than any of its allocated edges and by definition no proposal edge may be worse than the current refusal pointer. □

With this claim we showed that any operation that shifts a refusal pointer improves our potential function $\Theta(x)$. From this point on, we consider a setting where all refusal pointers are fixed. This also implies that Θ_1 does not change, thus, we only concentrate on Θ_2.

Update $\mathcal{P}$

If $\Theta_2(x)$ increases, then there is a machine m whose best proposal edge became better. Since the preference lists are fixed, this is only possible, if an edge that was not in $\mathcal{P} \cup \mathcal{P}'$ becomes blocking or possibly blocking, moreover, it becomes the best proposal edge of its machine. If it becomes blocking (and not possibly blocking), then update $\mathcal{P}$ adds an edge $jm \notin \mathcal{P} \cup \mathcal{P}'$ to $\mathcal{P}$. Blocking edges of type I have to fulfill three criteria, at least one of them was not fulfilled before the augmentation.

1. jm became unsaturated
   - One of the two possibilities for an edge to lose allocation occurs when $jm \in \rho \cap \mathcal{R}$. Since jm is already the worst allocated edge of j, it cannot become a blocking edge of type I.
   - Even if $jm \notin \rho$, it can lose allocation, but only if $x(jm)$ was reduced by $m = m_1$, the starting vertex of the alternating walk. It is the only machine that refuses allocation, and it does so only if $A - \bar{x}(m_1) > 0$. When the refusal happens, $x(m_1) = q(m_1)$ and m_1 has no worse allocated edge than jm_1. In addition, m_1 does not lose any allocation in the current step. Since $q(m_1)$ is full with edges better than jm_1, jm_1 is not blocking.
2. jm became better than the worst allocated edge of j
   Claim 25 shows that j's worst allocated edge never becomes worse during Phase I.
3. jm became better than the worst allocated edge of m or m became unsaturated
   - In the first case, m increased x along an edge worse than jm. This worse edge was in $\mathcal{P} \cup \mathcal{P}'$, hence jm already dominated x at m or m already had a refusal pointer. Thus, changing this property is not sufficient for jm to become blocking.
   - If m lost some allocation, then it was the last vertex on ρ and thus, it had a refusal pointer. According to our definition of $\mathcal{P}'$, all unsaturated edges that dominate x at their job and are incident to a refusal pointer at their machine already belonged to $\mathcal{P}'$. They may leave this set now and be added to $\mathcal{P}$, but $\mathcal{P} \cup \mathcal{P}'$ remains unchanged and thus, the best proposal edge at m remains unchanged as well.

Update $\mathcal{P}'$

$\Theta_2(x)$ may also increase if an edge $jm \notin \mathcal{P} \cup \mathcal{P}'$ is added to $\mathcal{P}'$ and it becomes the best proposal edge of m. Just like above, we check the three criteria that have to be fulfilled by edges in $\mathcal{P}'$.

1. jm became unsaturated

- First we consider the case when $jm \in \rho \cap \mathcal{R}$. But jm is then a refusal edge and worst allocated edges of jobs are never in $\mathcal{P}'$.
- The other option is that $x(jm)$ was reduced by $m = m_1$. As mentioned above, the only machine that refuses allocation is m_1, the first vertex on an alternating walk. Even if jm becomes a $\mathcal{P}'$-edge, m_1 had a better proposal edge at the beginning of the augmentation: the first edge of ρ.

2. jm became better than the worst allocated edge of j
   We can again rely on Claim 25.

3. m gained a refusal edge
   We supposed that set $\mathcal{R}$ is fixed, no refusal pointer moves.

We have investigated the effects of the update operation on $\Theta(x)$ for all three subsets of $E(H(x))$. Each round of the algorithm consists of the following three steps: finding an alternating walk, augmenting along it and then updating $H(x)$. The first procedure does not change $\Theta(x)$, the second strictly decreases it, and the last one never increases it. Thus, $\Theta(x)$ changes strictly monotonically in each round. Since $\Theta(x)$ is integer-valued and bounded, our algorithm terminates. □

Running time

The helper graph $H(x)$ has at most as many edges and vertices as G. In each iteration, $\Theta(x)$ improves. Consider first the case when only Θ_2 changes. The best proposal edge of each machine m can move along all $|\delta(m)|$ edges of m. Since the procedure is monotone, $|E|$ such steps can be executed in total, for all machines. Then, Θ_1 has to improve. Just like Θ_2, Θ_1's monotone behavior also allows $|E|$ steps in total. Yet it is not possible that both components need all $|E|$ rounds. When a refusal pointer $r(j) = jm$ switches to a better edge jm', most of the elements in vector Θ_2 remain unchanged.

Suppose the last augmentation along walk ρ shifted a single refusal pointer $r(j) = jm$. We investigate the change in Θ_2. Clearly, Θ_2 can be increased, since only lexicographic monotonicity of $\Theta(x)$ can be shown. There are at most three special machines in G: m, m' and the last machine in ρ if ρ is not a single cycle. Machines not in ρ remain unchanged. Other machines in ρ reallocate some allocation to better jobs than before. Thus, they improve their lexicographic situation and keep their refusal edges. Machines with the same set of refusal edges do not gain new possibly blocking edges, and machines with a better lexicographic situation do not gain new blocking edges either. Thus, all new edges in $\mathcal{P} \cup \mathcal{P}'$ must be incident to one of the three machines mentioned above. Due to the last operation along ρ that shifted $r(j)$, m possibly ceases to have any refusal edge, thus, it may lose its possibly blocking edges. Regarding the last machine on ρ, it loses allocation, and through that it may receive at most $|J| - 1$ new blocking edges. These edges were all possibly blocking before, moreover, they lead back to ρ. Gaining a refusal edge, m' may become the end vertex of new possibly blocking edges, but there are at most $|J| - 1$ of them. To summarize: when Θ_1 improves in one element, Θ_2 may increase in at most one element by at most $|J| - 1 < |V|$.

This argumentation shows that the number of iterations can be bounded by $\mathcal{O}(|V||E|)$ from above, because $\Theta(x)$ cannot have more different states during the execution of the algorithm. Next, we determine how much time is needed to execute a single augmentation. Procedure FindWalk starts with choosing any machine that has a blocking edge of type I. This can be done in $\mathcal{O}(|V|)$ time. Adding the best proposal edge and the refusal pointer takes constant time, if they are stored for each vertex. Since at most one vertex is visited twice by the walk, after $\mathcal{O}(|V|)$ steps ρ is chosen. Then, either of the two augmenting procedures is called. It modifies x on at most $\mathcal{O}(|V|)$ edges. At last, $\mathcal{R}, \mathcal{P}$, and $\mathcal{P}'$ are updated. As explained above, the change in those sets involves at most $\mathcal{O}(|V|)$ edges.

In total, the algorithm performs $\mathcal{O}(|V||E|)$ rounds, each of them needs $\mathcal{O}(|V|)$ time to be computed. Thus the accelerated Phase I algorithm runs in $\mathcal{O}(|V|^2|E|)$ time.

Our method resembles the well-known notion of rotations [49]. As mentioned in Section 1.1, they can be used when deriving a stable solution from another, by finding an alternating cycle of matching and non-matching edges and augmenting along them. In our algorithm, when we are searching for augmenting cycles or walks, we use an approach similar to rotations: jobs candidate their edges that are better than their worst positive edge, while machines choose the best out of them. However, two differences can be spotted right away. While rotations are always assigned to a stable solution different from the job-optimal, our method works on unstable input. Moreover, besides cycles, we also augment along paths and walks.

Accelerated second phase

The second phase can be accelerated in a very similar manner to the first phase. Instead of describing this new algorithm directly and proving its correctness using the same methods as above, we choose a simpler approach. The main idea in this subsection is that the accelerated second phase of our algorithm is actually the accelerated first phase of the same algorithm on a slightly modified instance. Thus, its correctness and running time have already been proved.

At the beginning of our argumentation we make these modifications on the instance $\mathcal{I}$ given at the termination of the accelerated Phase I algorithm. We show that the set of blocking edges of type I on the modified instance $\mathcal{I}'$ and the set of blocking edges of type II on $\mathcal{I}$ coincide. Then we let our accelerated Phase I algorithm run on $\mathcal{I}'$. At the end, we argue that its output is stable on $\mathcal{I}$.

Modified instance

After termination of the first phase, an allocation x_0 is given so that all blocking edges are of type II. This input of the second phase is modified in the following way. A dummy job j_d and edges between each machine and j_d are added to G. The capacity of these edges equals the maximum quota amongst all machines, and $q(j_d)$ is their sum. While j_d's preference list can be chosen arbitrarily, the new edges are ranked worst on the preference lists of the machines. The new graph is called G'. Not only the graph,

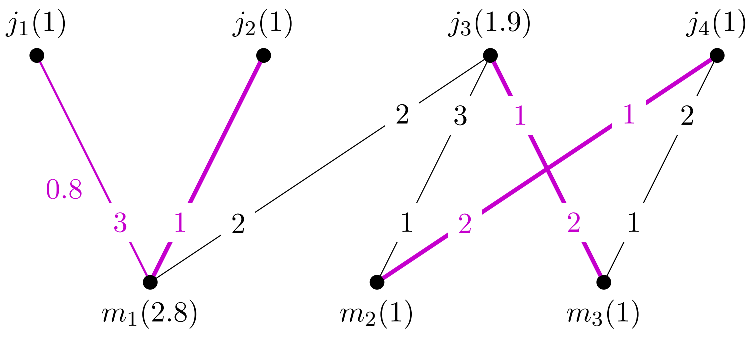


Figure 4.7: Allocation x_0 on $\mathcal{I}$, denoted by colored edges.

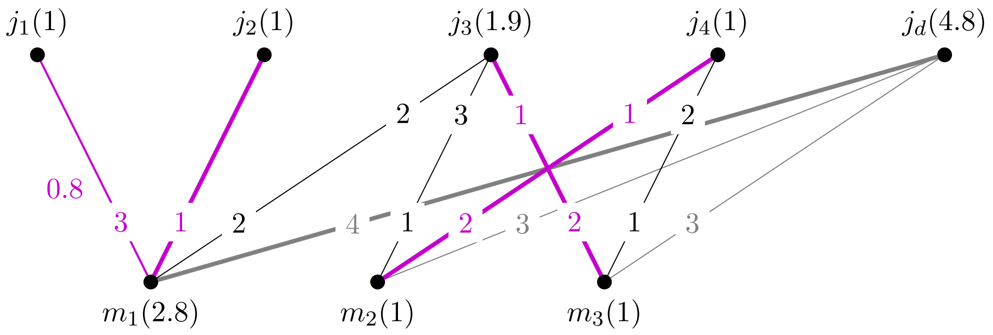


Figure 4.8: Allocation x'_0 on $\mathcal{I}'$, denoted by colored edges and the thick gray edge $j_d m_1$.

but also the allocation x_0 is slightly modified: machines under their quota assign all their free quota to j_d. In this new allocation x'_0 all machines are saturated. The new instance $\mathcal{I}'$ consists of G', q', c', O' and x'_0. A example instance modification is illustrated in Figures 4.7 and 4.8.

As mentioned above, our goal is to perform Phase I operations on $\mathcal{I}'$. In order to be able to do so, we swap the two sides: jobs play a passive role, while machines become the active players. Since each active vertex has a filled up quota, all blocking edges are of type I in $\mathcal{I}'$.

Note that $\mathcal{I}'$ was constructed in such a way that – regardless of the type of blocking – each edge blocking x also blocks x' and vice versa. This is due to the fact that the only difference between the two instances is that machines' free quota appears as allocation on their worst edge in $\mathcal{I}'$. The definition of a blocking edge does not distinguish between those two notions. In particular, given a specific allocation x_0 with no blocking edge of type I, the set of Phase II blocking edges on $\mathcal{I}$ and the set of Phase I blocking edges on $\mathcal{I}'$ trivially coincide.

Let us denote the output of the accelerated Phase I algorithm on $\mathcal{I}'$ by x', and its restriction to $E(G)$ by x.

Claim 26. *Allocation x is stable in $\mathcal{I}$.*

Proof. Suppose edge jm blocks x. On $\mathcal{I}'$, jm is unsaturated and dominates x' at both end vertices, hence jm blocks x' as well. Since x' is the output of the accelerated Phase I algorithm on $\mathcal{I}'$, jm is of type II. Our goal is to show by induction that $x'(m) = q(m)$ for all machines. Thus, a contradiction is derived, because in $\mathcal{I}'$ no Phase II blocking edge can occur.

Initially, $x'(m) = q(m)$ for all machines. The key property of x'_0 is that all unsaturated edges that dominate x'_0 at their (saturated) machine are not better than their job's worst edge in x'_0. Otherwise, they would be blocking edges of type I for x_0. Augmenting along a blocking edge jm in x'_0 can therefore never result in a refusal by the passive vertex j. Thus, after the first round, $x'(m) = q(m)$ still holds. Alternating walks are chosen in such a manner that jobs increase x' only on their best proposal edges. This guarantees that even after the first round, if jm dominates the current allocation x'_1 at m, it is not better than j's worst edge in x'_1. From here on, induction applies. ■

The running time of this phase cannot exceed the running time of the accelerated Phase I algorithm, since the size of $\mathcal{I}'$ does not exceed the size of $\mathcal{I}$ significantly.

With this we finished the proof of the following result.

Theorem 4.14. *For every allocation instance and given feasible allocation x, there is a sequence of better responses that leads to a stable allocation in $\mathcal{O}(|V|^2|E|)$ time.*

4.6 Conclusion and open problems

We solved the problem of uncoordinated processes on stable allocation instances algorithmically. Our first method is a deterministic better response algorithm that finds a stable solution through executing myopic steps. In case of rational input data, the existence of such an algorithm guarantees that random better response strategies terminate with a stable solution with probability one. Analogous results are shown for best response dynamics. We also prove that random best response strategies terminate in expected polynomial time on correlated markets, even in the presence of irrational data. An accelerated version of our first, better response algorithm is provided as well. For any real-valued instance, it terminates after $\mathcal{O}(|V|^2|E|)$ steps with a stable allocation. We also show a counterexample for a possible acceleration for the case of best response dynamics.

Applied to a matching instance, our best response algorithm performs the same steps as the two-phase best response algorithm of Ackermann et al. Our better response variant can also be interpreted as an extended version of the above mentioned method. The only difference is that while our first phase is better response, theirs is best response. However, this seems to be a minor difference, as their proof is also valid for a better response first phase, and our proof still holds if only best blocking edges are chosen. Moreover, stable allocations might be the most complex model in which this approach brings results. The most intuitive extension of Ackermann et al.'s algorithm for stable flows, defined by Fleiner [41], does not even result in feasible myopic changes.

On the other hand, our accelerated better response algorithm generalizes another known method, the polynomial algorithm that finds a stable allocation. Applied directly to an instance with empty allocation, our accelerated Phase II performs augmentations like the augmenting path algorithms of Baïou and Balinski [8], and of Dean and Munshi [34]. Since our accelerated Phase II is a slightly modified variant of our first algorithm, our solution concept offers a bridge between two known methods for solving two different problems, namely the paths to stability problem on SM instances and the stable allocation problem, providing a solution to both of them.

Future research may involve more complex stability problems from the paths-to-stability point of view, such as stable flows.

5 Unsplittable stable allocation problems

Building on the initial work of [33], we study a natural "unsplittable" variant of the stable allocation problem, where each assigned job must be *fully* assigned to a single machine. Such unsplittable bipartite assignment problems generally tend to be NP-hard, including previously-proposed variants of the unsplittable stable allocation problem [79]. Our main result is to show that under an alternative model of stability, the unsplittable stable allocation problem becomes solvable in polynomial time. Although this model is less likely to admit feasible solutions than the model proposed in [79], we show that in the event there is no feasible solution, our approach computes a solution of minimal total congestion (overfilling of all machines collectively beyond their capacities). We also describe a technique for rounding the solution of a stable allocation problem to produce "relaxed" unsplit solutions that are only mildly infeasible, where each machine is overcongested by at most a single job.

The results presented in this chapter are joint work with Brian C. Dean and have been published as [25].

5.1 Introduction

Motivation. Capacitated allocation problems have been extensively studied in the algorithmic literature, where typical objectives are to find a feasible assignment or one of maximum weight. While the fractional (splittable) variants of these problems are easy to solve in polynomial time via network flow techniques, it is NP-hard to find an unsplit allocation of either maximum total size or of maximum weight; the former is a variant of the multiple subset sum problem [20], and the latter is known as the multiple knapsack problem [22].

In this chapter, we study SA in the unsplittable setting, which was shown to be NP-hard in [79] using one natural definition for stability. We show here that by contrast, a different and more strict notion of stability, proposed initially in [33], leads to an $\mathcal{O}(|E|)$ algorithm for the unsplit problem. The tradeoff is that under this different notion of stability, it is unlikely that feasible allocations will exist. However, we show that by relaxing the problem to allow mildly infeasible allocations, our algorithm computes a "relaxed" unsplit stable allocation (in which each machine is filled beyond its capacity by at most the allocation of a single job), in which the total amount of overcongestion across all machines is minimized (so in particular, if there is a feasible allocation with no congestion, we will find it).

In addition to the straightforward application of scheduling jobs in a non-preemptive fashion, a motivating application especially for the unsplittable variant of the stable allocation problem is in assigning personnel with "two-body" constraints. For example, in the National Resident Matching Program [108], a married pair of medical school graduates might act as an unsplittable entity of size 2. This particular application has been studied in substantial detail in the literature; see [14] for further reference.

Literature review. Through the work of several former authors [37, 95, 96], the "relaxed" model has become relatively popular in the context of unsplittable bipartite assignment and unsplittable flow problems. The standard approximation algorithm framework typically does not fit these problems, since even finding any feasible solution is usually NP-hard. Instead, authors tend to focus on pseudo-approximation results with minimal congestion per machine or per edge. Analogous results were previously developed for unsplit stable allocation problems in [33], where an unsplit stable allocation can be found in linear time in which each machine is overcongested by at most a single job. The model of stability proposed in [33] is the one we further develop here, and among all of these prior approaches (including those for standard unsplittable bipartite assignment and flows), it seems to be the only unsplit model studied to date in which minimization of *total* congestion is possible in polynomial time. Hence, there is a substantial algorithmic incentive to consider this model, even though its notion of stability is less natural than in [79].

Contribution and structure. We review preliminary concepts and background material in Section 5.2. Then, in Section 5.3 we present our structural and algorithmic results: we show how to compute in $\mathcal{O}(|E|)$ time a job-optimal allocation that maximizes the total size $|x|$ of all assigned jobs, and a machine-optimal allocation that minimizes $|x|$. We also show that this machine-optimal solution minimizes total congestion. In order to produce potentially other solutions (e.g., that might be more fair to both sides), in Section 5.4 we show also a technique for "rounding" a solution of the fractional stable allocation problem to obtain a relaxed unsplit solution.

5.2 Preliminaries

5.2.1 Problem definition

We use the notation introduced in Chapter 4 for SA. If $x(jm) \in \{0, q(j)\}$ for all $jm \in E(G)$, we say the allocation is *unsplit*, since each assigned job is assigned in its entirety to a single machine. We often forgo the use of edge capacities $c(jm)$ when discussing unsplit allocations, since an edge jm can simply be deleted if $c(jm) < q(j)$. We say that edges with positive x value are *in* x. If any machine m has $q(m) > \sum_{j \in J} c(jm)$, then $q(m)$ is set to $\sum_{j \in J} c(jm)$. Machines with $x(m) = q(m)$ are *saturated*. Later, when $x(m) > q(m)$ occurs in the relaxed version of the problem, we talk about *over-capacitated* machines. If any job prefers machine m to any of its allocated machines, then m is called *popular*, otherwise m is *unpopular*. It is easy to see that all popular machines must be saturated in any stable allocation. Note that the expression "popular'' is not related to the notion of popular matchings discussed in Chapter 7, it is by chance that both concepts are named identically in the literature.

For simplicity, we introduce a "dummy" machine m_d with high capacity, which acts as the last choice for every job. This lets us assume without loss of generality that an unsplittable allocation always exists in which every job is assigned. In this context, we

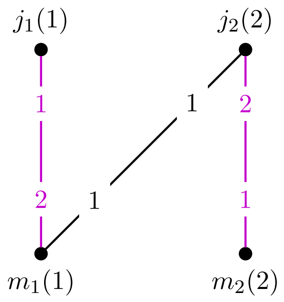


Figure 5.1: On this instance, the unique stable allocation $x(j_2m_1) = x(j_2m_2) = 1$ is fractional. The colored unsplit allocation is stable according to the definition in [79], but it is blocked by edge j_2m_1 according to our definition of stability.

define the *size* $|x|$ of an allocation x so that jobs assigned to m_d do not count, since they are in reality unassigned.

From a complexity standpoint, one of the main results of this chapter is that how we define stability in the unsplit case seems quite important. In [79], the following natural definition was proposed: an edge jm is blocking if j prefers m to its current partner, and if m prefers j over $q(j)$ units of its current allocation or unassigned quota. It was shown in [79] that this definition makes the computation of an unsplit stable allocation NP-hard. We therefore consider an alternative, stricter notion of stability where edge jm is blocking if j prefers m to its current partner, and if m prefers j over *any amount* of its current allocation or if it has free quota. In other words, if j would prefer to be assigned to m over its current partner, then m must be saturated with jobs that m prefers to j. A simple instance demonstrates the difference between these two definitions in Figure 5.1.

Problem 15. UNSPLIT SA
Input: $\mathcal{I} = (G, q, c, O)$; an SA *instance.*
Question: Is there a stable allocation x such that $x(jm) \in \{0, q(j)\}$ for all $jm \in E(G)$?

Aside of the integrality constraint, this definition is fully aligned with the classical definition of a stable allocation. As in the splittable case, popular machines must therefore be saturated. Practice shows [87] that if a hospital is willing to hire one person in a couple, but it has no free job opening for the partner, it will most likely make room for both applicants. This gives sufficient practical motivation to justify our definition of a blocking pair.

The existence of an unsplit stable allocation cannot be guaranteed. A simple instance where the unique stable allocation is fractional is shown in Figure 5.1. Similar to the figures in Chapter 4, the quota of each job and machine is displayed above and below the vertex, while the preference lists are displayed on the edges.

5.2.2 Relaxed unsplit allocations

The downside of our alternative definition of stability is that it is unlikely to allow feasible unsplit stable allocations to exist in most large instances. Therefore, we consider allowing mildly-infeasible solutions where each machine can be over-capacitated by a single job – a model popularized by previous results in the approximation algorithm literature for standard unsplittable assignment problems [37, 95, 96], and introduced in the context of UNSPLIT SA by Dean et al. [33].

Definition 5.1. *Function $x : E(G) \to \mathbb{R}_{\geq 0}$ is a* relaxed unsplit allocation *if the following three properties are fulfilled:*

1. *$x(jm) \in \{0, q(j)\}$ for every edge $jm \in E(G)$;*
2. *$x(j) \leq q(j)$ for every job $j \in J$;*
3. *for each machine m, the removal of the least preferred job assigned to m would cause $x(m) < q(m)$.*

Our definition of stability extends readily to the relaxed setting. We say a relaxed unsplit allocation x is *stable* if for every edge jm with $x(jm) = 0$, either j is assigned to a machine that j prefers to m, or m's quota is filled or exceeded with jobs that m prefers to j. Otherwise, if edge jm with $x(jm) = 0$ is preferred by j to its allocated machine and m's quota is not filled up with better edges than jm, then *jm blocks x*. The model introduced in [33] allows $x(m) \leq q(m)$, but we believe strict inequality is actually a better choice – mathematically and from a modeling perspective. For example, the old definition applied to a hospital-resident matching scenario with married couples might cause a hospital to accept two more residents than its quota, while the new definition would only require accepting one more resident. The results in [33] hold with either definition.

Problem 16. RELAXED UNSPLIT SA
Input: $\mathcal{I} = (G, q, c, O)$; an SA instance.
Question: Is there a stable allocation x with relaxed quota constraints for machines? That is, for each machine m, the removal of the least preferred job assigned to m would cause $x(m) < q(m)$ and $x(jm) \in \{0, q(j)\}$ for all $jm \in E(G)$.

Note that the relaxed unsplit model differs from the non-relaxed unsplit model with capacities inflated by $\max_{j \in J} q(j)$, since stability is still defined with respect to original capacities. It may be best to regard "capacities" here as constraints governing start time, rather than completion time of jobs. A machine below its capacity is always willing to launch a new job, irrespective of job size.

5.3 Machine-optimal relaxed unsplit allocations

In [33], a version of the Gale-Shapley algorithm is described to find the job-optimal relaxed unsplit stable allocation x_{jopt}. In this context, job-optimal means that there is

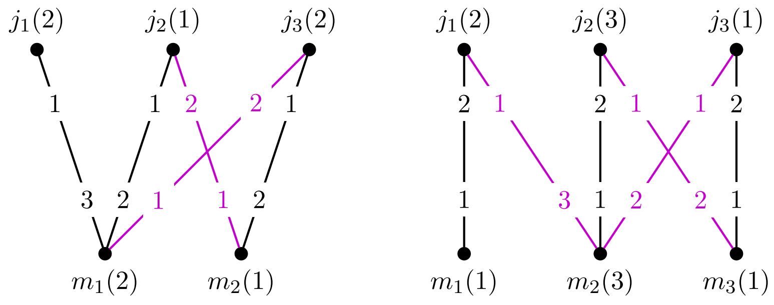


Figure 5.2: The instance on the left admits two relaxed unsplit allocations differing in cardinality. The colored edges form a relaxed unsplit stable allocation of size 3, while the remaining edges build another relaxed unsplit stable allocation of size 5. The second example is evidence against an exact Rural Hospitals Theorem, where m_1 is empty in one relaxed unsplit stable allocation (given by the colored edges) but filled beyond its capacity in another (given by the black edges).

no relaxed unsplit stable allocation x' such that any job is assigned to a better machine in x' than in x_{jopt}. The implementation described in [33] runs in $\mathcal{O}(|E||V|\log|V|)$ time, but this running time could be brought down to $\mathcal{O}(|E|)$. In this section, we show how to define and compute a *machine-optimal* relaxed unsplit stable allocation x_{mopt} also in $\mathcal{O}(|E|)$ time, and we prove the following:

Theorem 5.2. *Among all relaxed unsplit stable allocations x, $|x|$ is maximized at $x = x_{jopt}$ and minimized at $x = x_{mopt}$.*

One of the main challenges with computing a machine-optimal allocation is defining machine-optimality. In SA, existence of a machine-optimal allocation follows from the fact that all stable allocations form a distributive lattice under the standard min-min ordering relationship introduced in [8]. Recall that the min-min criterion elects the allocation with the lower allocation value on the worst allocated edge of m to be the better allocation for m. However, as already mentioned in Section 4, this ordering seems to depend crucially on the existence of a Rural Hospitals Theorem, which no longer holds in the relaxed unsplit case, since relaxed unsplit stable allocations can differ in cardinality, as demonstrated in Figure 5.2. The same figure shows that even an appropriately relaxed version of the Rural Hospitals Theorem seems difficult to formulate over relaxed instances: machines can be saturated or even over-capacitated in one relaxed unsplit stable allocation, while being empty in another one. Nonetheless, we can still prove a result in the spirit of the Rural Hospitals Theorem, which we discuss further in Section 5.3.3.

Without an “exact” Rural Hospitals Theorem, comparing two allocations using the original min-min ordering seems problematic, and indeed one can construct instances where two relaxed unsplit stable allocations are incomparable according to this criterion

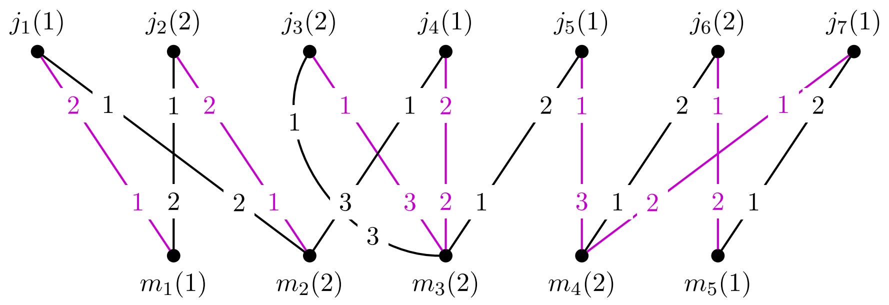


Figure 5.3: The graph shows two relaxed stable unsplit allocations (denoted by colored and black edges, respectively) that are incomparable according to the min-min criterion from the perspective of m_3. In the black allocation, m_3 receives its first and third-choice jobs of size 3 in total, while in the colored allocation, its second and third choice job of size 3 are assigned to it.

Algorithm 3 Reversed relaxed unsplit Gale-Shapley

```
$x(jm_d) := q(j)$ for all $j \in J$, $x(jm) := 0$ for every other $jm \in E(G)$
**while** $\exists m : x(m) < q(m)$ with a non-empty preference list **do**
    $m$ proposes to its best job $j$ with value $q(j)$
    **if** $j$ prefers $m$ to its current partner **then**
        $x(jm) := q(j)$
        $x(jm') := 0$ for $\forall m' \neq m$
    **end if**
    delete $j$ from $m$'s preference list
**end while**
```

(see Figure 5.3). We therefore adopt the same natural ordering relation as in Chapter 4: *lexicographic order*. Note that since jobs are always assigned to machines in an unsplit fashion, the lexicographic and min-min relations are actually the same from the job's perspectives; hence, "job optimal" means the same thing under both. The lexicographic position of the same agent in different allocations can always be compared, and we say a relaxed stable allocation x is *machine-optimal* if it is at least as good for all machines as any other relaxed stable allocation (although we still need to show that such an allocation always exists).

5.3.1 The reversed Gale-Shapley algorithm

For SM, the Gale-Shapley algorithm can be reversed easily with women proposing instead of men to obtain a woman-optimal solution. In Claims 27-29 we show that this idea can be generalized (carefully accounting for multiple assignment and congestion among machines) to compute a machine-optimal relaxed unsplit stable allocation. The pseudocode for the algorithm appears as Algorithm 3.

Claim 27. *The algorithm terminates in $\mathcal{O}(|E|)$ time.*

Proof. In each step a job is deleted from a machine's preference list. ∎

Claim 28. *The algorithm produces an allocation x that is a relaxed unsplit stable allocation.*

Proof. First, we check the three feasibility constraints of Definition 5.1 for x. Since proposals are always made with $q(j)$ and refusals are always full rejections, the quota constraints of the jobs cannot be violated. Moreover, each job is assigned to exactly one machine. Machines can be over-capacitated, but deleting the worst job from their preference list results in an allocation under their quota. Otherwise the machine would not have proposed along the last edge. If x is unstable, then there is an empty edge jm blocking x. During the execution, m must have proposed to j. This offer was rejected, because j already had a better partner in the current allocation. Since jobs monotonically improve their position in the allocation, this leads to a contradiction. ∎

Claim 29. *The output x is the machine-optimal relaxed unsplit stable allocation (i.e., no machine has a better lexicographic position in any other relaxed unsplit stable allocation).*

Proof. Assume that there is a relaxed unsplit stable allocation x', where some machines are better off than in x. To be more precise, in the symmetric difference $x \triangle x'$, the best edge incident to these machines belongs to x'. When running the reversed relaxed unsplit Gale-Shapley algorithm, there is a step when the first such edge jm_1 carries a proposal from m_1 but gets rejected. Otherwise, m_1 has filled up or exceeded its quota in x with only better edges than jm_1. Let us consider only this edge jm_1 first and denote the feasible, but possibly unstable relaxed allocation produced by the algorithm so far by x_0.

When j refused jm_1, it already had a partner m_0 in x_0, which was better than m_1. Even if there is no guarantee that $jm_0 \in x$, it is certain that $jm_0 \notin x'$ and jm_0 does not block x', though $\text{rank}_j(jm_0) > \text{rank}_j(jm_1)$ for $jm_1 \in x'$. It is only possible if m_0 is saturated or over-capacitated in x' with edges better than jm_0. Since $jm_0 \in x_0$, allocation x_0 cannot contain all of these edges, otherwise m_0 is congested in x_0 beyond the level required for a relaxed unsplit allocation. During the execution of the reversed relaxed unsplit Gale-Shapley algorithm, m_0 proposed along all of these edges and got rejected by at least one of them. This edge is never considered again, it cannot enter x later. Thus, jm_1 is not the first edge in $x' \setminus x$ that was rejected in the algorithm. ∎

With this, we completed the constructive proof of the following theorem:

Theorem 5.3. *In* RELAXED UNSPLIT SA, *the machine-optimal relaxed unsplit stable allocation x_{mopt} can be computed in $\mathcal{O}(|E|)$ time.*

5.3.2 Properties of the job- and machine-optimal solutions

In this section we discuss structural properties of RELAXED UNSPLIT SA, related to the Rural Hospitals Theorem and the lattice structure of SA.

Theorem 5.4. *In* RELAXED UNSPLIT SA*, the job-optimal relaxed unsplit stable allocation x_{jopt} is the machine-pessimal relaxed unsplit stable allocation and vice versa, the machine-optimal relaxed unsplit stable allocation x_{mopt} is the job-pessimal relaxed unsplit stable allocation.*

Proof. We start with the first statement. Suppose that there is a relaxed unsplit stable allocation x' that is worse for some machine m than x_{jopt}. This is only possible if m's best edge jm in $x_{\text{jopt}} \triangle x'$ belongs to x_{jopt}. Since x_{jopt} is the job-optimal solution, jm', j's edge in x' is worse than jm. But m is then saturated or over-capacitated in x' with better edges than jm. We assumed that all edges in x' that are better than jm are also in x_{jopt}. Thus, omitting m's worst job from x_{jopt} leaves m at or over its quota.

The second half of the theorem can be proved similarly, using the reversed Gale-Shapley algorithm. Assume that there is a relaxed unsplit stable allocation x' that assigns some jobs to worse machines than x_{mopt} does. Let us denote the set of edges preferred by any job to its allocated machine in x' by $E(G')$. Due to our indirect assumption, $E(G')$ contains some edges of x_{mopt}. When running the reversed Gale-Shapley algorithm in the instance, there is an edge $jm \in E(G')$ that is the first edge in $E(G')$ carrying a proposal. Since j is not yet matched to a better machine, it also accepts this offer. Even if $jm \notin x_{\text{mopt}}$, j's edge in x_{mopt} is at least as good as m, because jobs always improve their position during the course of the reversed Gale-Shapley algorithm. On the other hand, m cannot fulfill its quota in x_{mopt} with better edges than jm, simply because the proposal step along jm took place.

Since $jm \notin x'$, but j prefers jm to its edge in x', m is saturated or over-capacitated with better edges than jm in x'. As observed above, not all of these edges belong to x_{mopt}. Let us denote one of them in $x' \setminus x_{\text{mopt}}$ by $j'm$. Before proposing along jm, m submitted an offer to j' that has been refused. The only reason for such a refusal is that j' has already been matched to a better machine m'. But since $j'm \in x'$, $j'm' \in E(G')$. This contradicts to our indirect assumption that jm is the first edge in $E(G')$ that carries a proposal. □

Theorem 5.2 also follows from the proof above. It is straightforward that x_{jopt} is a relaxed unsplit stable allocation of largest size, because no job exists that is unmatched in x_{jopt}, but matched in some other relaxed unsplit stable allocation. Assume there is a relaxed unsplit stable allocation x with $|x| < |x_{\text{mopt}}|$. Since we know now that $|x_{\text{mopt}}|$ is the worst for every single job individually, no job can occur that is matched in x_{mopt}, but unmatched in some other relaxed unsplit stable allocation.

We note that although we can compute the job-optimal and machine-optimal relaxed unsplit stable allocations, there in general does not appear to be an obvious underlying lattice structure behind relaxed unsplit solutions. For SM or SA, computing the meet or join of two solutions is fairly easy. In order to reach the join (meet) of x_1 and x_2, all jobs choose the better (worse) edge set out of those two allocations [21]. The example in Figure 5.4 illustrates that this property does not carry over to relaxed unsplit allocations. Similar examples can easily be constructed to show that choosing the worse allocation also can lead to instability.

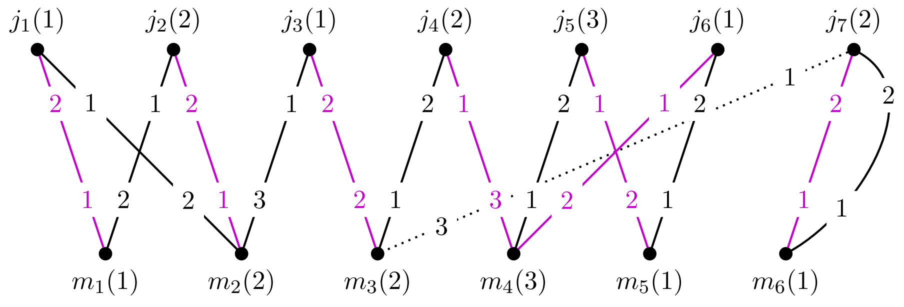


Figure 5.4: This instance is a counterexample showing the difficulty of formulating join and meet operations. If all jobs chose the better allocation, m_3 remains empty and j_7m_3 becomes blocking. If all jobs chose the worse allocation, then m_3 exceeds its quota with so much allocation that it violates the feasibility of relaxed solutions.

Our ability to compute x_{mopt} in $\mathcal{O}(|E|)$ time now gives us a linear-time method for solving UNSPLIT SA.

Lemma 5.5. *If an instance $\mathcal{I}$ of* RELAXED UNSPLIT SA *admits an unsplit stable assignment x, then the machine-optimal relaxed unsplit stable assignment x_{mopt} in the corresponding relaxed instance $\mathcal{I}'$ is also an unsplit stable assignment in $\mathcal{I}$.*

Proof. Suppose the statement is false, e.g., although there is an unsplit stable assignment x, x_{mopt} is no unsplit stable assignment in $\mathcal{I}$. This can be due to two reasons: either the feasibility or the stability of x_{mopt} is violated in $\mathcal{I}$. The latter case is easier to handle. An allocation that is feasible in both instances and stable in $\mathcal{I}'$ cannot be blocked by any edge in $\mathcal{I}$, since the set of unsaturated edges is identical in both instances. The second case, namely if x_{mopt} violates some feasibility constraint on $\mathcal{I}$, needs more care.

$\mathcal{I}$ and $\mathcal{I}'$ differ only in the constraints on the quota of machines. If x_{mopt} is infeasible in $\mathcal{I}$, then there is a machine m for which $x_{\text{mopt}}(m_1) > q(m_1)$. Regarding the unsplit stable assignment x, the inequality $x(m_1) \leq q(m_1)$ trivially holds. Now we use Theorem 5.2 for x and x_{mopt} that are both relaxed unsplit stable assignments in $\mathcal{I}'$. This theorem implies that if there is a machine m_1 with $x_{\text{mopt}}(m_1) > x(m_1)$, then another machine m_2 exists for which $x_{\text{mopt}}(m_2) < x(m_2)$ holds.

This machine m_2 plays a crucial role in our proof. It has a lower allocation value in the machine-optimal relaxed solution x_{mopt} than in another relaxed stable solution x in $\mathcal{I}$. Its lexicographic position can only be better in x_{mopt} than in x if the best edge j_2m_2 in $x \triangle x_{\text{mopt}}$ belongs to x_{mopt}. Moreover, $x \triangle x_{\text{mopt}}$ also contains an edge $j_3m_2 \in x$, otherwise $x_{\text{mopt}}(m_2) > x(m_2)$. Naturally, $\text{rank}_m(j_2m_2) < \text{rank}_m(j_3m_2)$. At this point, we use the property that $x_{\text{mopt}}(m_2) < q(m_2)$. Since m_2 has free quota in x_{mopt} and j_3m_2 is not a blocking edge, j_3 must be matched to a machine better than m_2 in x_{mopt}. Thus, there is a job that comes better off in the machine-optimal (and job-pessimal) relaxed solution than in another relaxed stable solution. This contradiction to Theorem 5.4 finishes our proof. □

Lemma 5.5 shows that if there is an unsplit solution, it can be found in linear time by computing the machine-optimal relaxed solution. Unfortunately, the existence of such an unsplit assignment is not guaranteed. Our next result applies to the case when no feasible unsplit solution can be found. In terms of congestion, with the machine-optimal solution we come as close as possible to feasibility.

Theorem 5.6. *Among all relaxed unsplit stable solutions in an instance of* RELAXED UNSPLIT SA*, x_{mopt} has the lowest total congestion.*

Proof. Let M_u denote the set of machines that remain under their quota in x_{mopt}. Note that $\sum_{m \notin M_u} x_{\text{mopt}}(m)$, the total allocation value on the remaining machines clearly determines the total congestion of x_{mopt}, given by $\sum_{m \notin M_u} x_{\text{mopt}}(m) - q(m)$. Let x be an arbitrary relaxed solution. Due to Theorem 5.2, the size of the allocation is minimized at x_{mopt}. Therefore, for any relaxed unsplit stable allocation x, the following inequalities hold:

$$\begin{aligned}
\sum_{m \in M} x(m) &\geq \sum_{m \in M} x_{\text{mopt}}(m) \\
\sum_{m \notin M_u} x(m) + \sum_{m \in M_u} x(m) &\geq \sum_{m \notin M_u} x_{\text{mopt}}(m) + \sum_{m \in M_u} x_{\text{mopt}}(m) \\
\sum_{m \notin M_u} x(m) - \sum_{m \notin M_u} x_{\text{mopt}}(m) &\geq \sum_{m \in M_u} x_{\text{mopt}}(m) - \sum_{m \in M_u} x(m) \\
\sum_{m \notin M_u} (x(m) - q(m)) - \sum_{m \notin M_u} (x_{\text{mopt}}(m) - q(m)) &\geq \sum_{m \in M_u} x_{\text{mopt}}(m) - \sum_{m \in M_u} x(m)
\end{aligned}$$

At this point, we investigate the sign of both sides of the last inequality. The core of our proof is to show that for each $m \in M_u$ and relaxed stable solution x, the inequality $x_{\text{mopt}}(m) \geq x(m)$ holds. This result, proved below in Lemma 5.7, has two benefits. On one hand, the term on the right hand-side of the last inequality is non-negative. Therefore, the inequality implies that the total congestion on machines in $M \setminus M_u$ is minimized at x_{mopt}. On the other hand, no machine in M_u is over-capacitated in any relaxed solution. Thus, the total congestion is minimized at x_{mopt}. □

Our last observation in this subsection refers to the unsaturated machines.

Lemma 5.7. *For every $m \in M_u$ and relaxed solution x, the inequality $x_{mopt}(m) \geq x(m)$ holds.*

Proof. Suppose that there is a machine $m \in M_u$ for which $x_{\text{mopt}}(m) < x(m)$ for some relaxed solution x. Since m is unsaturated in x_{mopt}, it is unpopular. On the other hand, there is at least one job j for which $jm \in x \setminus x_{\text{mopt}}$. As m is unpopular in x_{mopt}, j is allocated to a better machine in x_{mopt} than in x. Since as established in Theorem 5.7 x_{mopt} is the job-pessimal relaxed solution, thus we derived a contradiction. □

5.3.3 A variant of the Rural Hospitals Theorem

In the relaxed unsplit case, one can find counterexamples against an exact Rural Hospitals Theorem (e.g., where all machines have the same amount of allocation in all relaxed unsplit allocations) or even a weakened theorem stating that all unsaturated and all congested machines have the same status in all relaxed unsplit allocations (see Figure 5.2). Lemma 5.7 above however suggests an alternative variant of the Rural Hospitals Theorem that does hold.

Theorem 5.8. *A machine m that is not saturated in x_{mopt} will not be saturated in any relaxed unsplit stable solution, and a machine m that is over-capacitated in x_{jopt} must at least be saturated in every relaxed unsplit stable solution.*

Proof. The first part is shown by Lemma 5.7. For the second part, consider a machine m that is over-capacitated in x_{jopt} but has $x(m) < q(m)$ in some relaxed unsplit allocation x. Consider any job j in $x_{\text{jopt}} \backslash x$, and note that since x_{jopt} is job-optimal, j prefers m to its partner in x. Hence, jm blocks x. □

As of the jobs' side, Theorem 5.4 already guarantees that if a job is unmatched in x_{jopt}, then it is unmatched in all relaxed stable solutions and similarly, if it is matched in x_{mopt}, then it is matched in all relaxed stable solutions.

5.4 Rounding algorithms

We have seen now how to compute x_{jopt} and x_{mopt} in linear time. We now describe how to find potentially other relaxed unsplit solutions by "rounding" (fractional) solutions of SA. For example, this could provide a heuristic for generating relaxed unsplit solutions that are more balanced in terms of fairness between the jobs and machines. Our approach is based on augmentation around rotations, alternating cycles that are commonly used in SM and SA to move between different stable solutions. We shortly discussed rotations in Chapter 1.1 and advise the reader to [34, 49] for more details on them. We would also like to point it out that the proposal-refusal machinery used in this section resembles the techniques applied in Chapter 4, although due to the different nature of the two problems the technical details are different.

We begin with a stable allocation x with $x(j) = q(j)$ for every job j, thanks to the existence of a dummy machine. For each job j that is not fully assigned to its first-choice machine, we define its *refusal edge* $r(j)$ to be the worst edge jm incident to j with $x(jm) > 0$. Jobs with refusal edges also have *proposal edges* – namely all their edges ranked better than $r(j)$. Recall that a machine with incoming proposal edges is said to be *popular*. We call a machine *dangerous* if it is over-capacitated and has zero assignment on all its incoming proposal edges.

Claim 30. *Consider a popular machine m in some fractional stable allocation x. Among all proposal edges incoming to m, at most one has positive allocation value in x, and this positive proposal edge is ranked worse on m's preference list than any other edge of m with positive allocation.*

Proof. Let $\text{rank}_m(j_1m) < \text{rank}_m(j_2m)$ be proposal edges such that $x(j_1m)$ and $x(j_2m)$ are both positive. Note that j_1m blocks x, since j_1 and m have worse allocated edges in x. A similar argument implies the last part of the claim. ■

Our algorithm proceeds by a series of augmentations around rotations, defined as follows. We start from a popular, non-dangerous machine m (if no such machine exists, the algorithm terminates, having reached an unsplit solution). Since m is popular and non-dangerous, it has incoming proposal edges with positive allocation by definition, and due to Claim 30, it must have exactly one such edge jm. We include jm as well as j's refusal edge jm' in our partial rotation, then continue building the rotation from m' (again finding an incoming proposal edge, etc.). We continue until we close a cycle, visiting some machine m visited earlier (in which case we keep just the cycle as our rotation, not the edges leading up to the cycle), or until we reach a machine m that is unpopular or dangerous, where our rotation ends.

To enact a rotation, we increase the allocation on its proposal edges by ε and decrease along the refusal edges by ε, where ε is chosen to be as large as possible until either (i) a refusal edge along the rotation reaches zero allocation, or (ii) a dangerous machine at the end of the rotation drops down to being exactly saturated from being over-capacitated, and hence ceases to be dangerous. We call case (i) a "regular" augmentation. This concludes the algorithm description.

Claim 31. *The algorithm terminates after $\mathcal{O}(|E|)$ augmentations.*

Proof. Jobs remain fully allocated during the whole procedure, and their lexicographic positions never worsen. With every regular augmentation, some edge stops being a refusal edge, and will never again be increased or serve as a proposal or refusal edge. We can therefore have at most $\mathcal{O}(|E|)$ regular augmentations. Furthermore, a machine can only become dangerous if one of its incoming refusal pointers reaches zero allocation, so the number of newly-created dangerous machines over the entire algorithm is bounded by $|E|$. Hence, the number of non-regular augmentations is at most $\mathcal{O}(|M| + |E|) = \mathcal{O}(|E|)$. ■

Claim 32. *The final allocation x is a feasible relaxed unsplit assignment.*

Proof. Since we start with a feasible assignment and jobs never lose or gain allocation, the quota condition on jobs cannot be violated. If there is any edge jm with $0 < x(jm) < q(j)$, then j has at least two positive edges, the better one must be a positive proposal edge. This contradicts the termination condition, and hence x is unsplit.

We now show that deleting the worst job from each machine results in an allocation strictly below the machine's quota. It is clearly true at the beginning, where no machine is over-capacitated (since x starts out as a feasible stable allocation). The only case when $x(m)$ increases is when m is the first machine on a rotation. As such, m has a positive proposal edge jm, which is also its worst allocated edge, due to our earlier claim. If m is not over-capacitated when choosing the rotation, then even if $x(jm)$ rises as high as $q(j)$, this increases $x(m)$ by strictly less than $q(j)$. Thus, deleting jm, the worst allocated edge of m, guarantees that $x(m)$ sinks under $q(m)$. If m is saturated or over-capacitated when choosing the rotation, then jm would have been the best proposal edge of m earlier, when $x(m)$ was not greater than $q(m)$. Thus, assigning j entirely to m does not violate the relaxed quota condition. Let us consider the last step as $x(m)$ exceeded $q(m)$. Again, m was the starting vertex of an augmenting path, having a positive proposal edge. If it was jm, our claim is proved. Otherwise m became over-capacitated while $x(jm)$ was zero, and then increased the allocation on jm. But between those two operations, m had to become dangerous, because it switched its best proposal edge to jm. Dangerous machines never start alternating paths. Thus, we have a contradiction to the fact that we considered the last step when $x(m)$ exceeded $q(m)$. ■

Claim 33. *The final allocation x is stable.*

Proof. Suppose some edges block x. Since we started with a stable allocation, there was a step during the execution of the algorithm when the first edge jm became blocking. Before this step, either j or m was saturated or over-capacitated with better edges than jm. The change can be due to two reasons: either j gained allocation on an edge worse than jm, or m gained allocation on an edge worse than jm. As already mentioned, j's lexicographic position never worsens: $\text{rank}_j(p) < \text{rank}_j(r(j))$ always holds. The second event also cannot occur, because machines always increase the allocation value on their best positive proposal edge. An edge jm that becomes blocking when allocation is increased on an edge worse than it, was already a proposal edge before. Thus, m would have chosen jm, or an edge better than jm to add it to the augmenting path. ■

Since each augmentation requires $\mathcal{O}(|V|)$ time and there are $\mathcal{O}(|E|)$ augmentations, our rounding algorithm runs in $\mathcal{O}(|E||V|)$ total time. If desired, dynamic tree data structures can be used (much like in [34]) to augment in $\mathcal{O}(\log |V|)$ time, bringing the total time down to just $\mathcal{O}(|E| \log |V|)$.

Although jobs improve their lexicographic position in each rotation, the output of the algorithm is not necessarily x_{jopt}. In fact, even x_{mopt} can be reached via rounding. Ideally, this approach can serve as a heuristic to generate many other relaxed unsplit stable allocations, if run from a variety of different initial stable solutions x.

5.5 Conclusion and open problems

In this chapter, we reformulated the definition of stable unsplit allocations. Several basic properties of SM and SA are discussed on the relaxed setting. Most of them carry over to unsplit allocations, but we also showed examples for certain structural properties that do not hold in the unsplittable case.

We proved that the reversed relaxed unsplit Gale-Shapley algorithm can be used to decide in polynomial time whether a regular instance admits an unsplit stable assignment. If not, relaxed solutions can be searched for. Besides constructing the job-optimal and the machine-optimal solutions, we also showed a method that rounds any fractional stable solution to a relaxed unsplit stable allocation.

On the other hand, we mentioned that the Rural Hospitals Theorem has no generalization for unsplit assignments: relaxed stable solutions can have different cardinality and the same vertex can have various lexicographic positions in them. The set of relaxed solutions also has been investigated: the distributive lattice structure known for SM and SA cannot be observed here. Although rotations can be used to derive an unsplit solution from a fractional one, moving from one unsplit solution to another one is impossible just by rotations along cycles.

This latter obstacle raises a problem about optimizing over the set of solutions. If the instance contains cost on edges, how to find a minimum-value stable solution? Rounding the optimal fractional assignment does not necessarily lead to an optimal unsplit allocation. A similar question can be addressed about fair allocations. Aside from these, any combination with well-known notions in stability problems can be studied: ties, restricted edges, etc. Since unsplittable flows are widely studied and stability is also defined for network flows (as also discussed in Chapter 6), unsplittable stable flows also could be studied extending either of the blocking definitions mentioned in this chapter. Another straightforward generalization would be to define k-splittable allocations and investigate their properties.

6 Stable flows

As most matching problems, stable matchings also can be extended to network flows. In a stable flow instance, all vertices in a directed network express their preferences over their incident edges. A network flow is stable if there is no group of vertices that all could benefit from rerouting the flow along a walk.

We start this chapter with presenting the polynomial version of the Gale-Shapley algorithm for stable flows. Then, stable flows with forced and forbidden edges are discussed. Although a polynomial algorithm for finding a stable flow with restricted edges can easily be derived from methods known for weighted stable allocation problems [34], no direct combinatorial method had been discovered yet. We fill this gap and provide a simple and fast algorithm for the problem. Finally, we study stable multicommodity flows. A stable solution is known to exist [67], but it is PPAD-hard to find one. We show that it is NP-complete to decide whether an integral solution exists.

6.1 Introduction

Motivation. Stable flows can be seen as a generalization of stable marriage, many-to-one matching, many-to-many matching and allocation problems. To the best of our knowledge, it is the most complex generalization of SM, thus it plays a crucial role from the theoretical point of view. Furthermore, it also has a great potential regarding applications.

A directed network with preferences models a market situation. The vertices are vendors dealing with some goods, while edges connecting them represent possible deals. Through this preference list, each vendor specifies how desirable a specific trade would be to him. Sources and sinks model producers and consumers. A feasible network flow is stable, if there is no set of vendors such that they mutually agree to modify the flow in the same manner. A blocking walk represents a set of vendors with a set of possible deals so that all of these vendors would benefit from rerouting some flow along the blocking walk.

Literature review. The notion of stability was extended to so-called "vertical networks" by Ostrovsky in 2008 [82]. Even though the author proves the existence of a stable solution and presents an extension of the Gale-Shapley algorithm, his model is restricted to unit-capacity acyclic graphs. Stable flows in the more general setting were defined by Fleiner [41], who reduced the stable flows problem to SA. He translated the existence of stable solutions, the Rural Hospitals Theorem and the lattice structure from SA to stable flows. In Section 6.2 we elaborate on these results.

The best currently known computation time of a stable flow is $\mathcal{O}(|E| \log |V|)$ on a network with vertex set V and edge set E. This bound is due to Fleiner's reduction to SA and its fastest solution described by Dean and Munshi [34]. Since the reduction takes $\mathcal{O}(|V|)$ time and does not change the instance size significantly, furthermore, weighted SA can be solved in $\mathcal{O}(|E|^2 \log |V|)$ time [34], the same holds for the maximum weight stable flow problem. The Gale-Shapley algorithm can also be extended for stable flows [28]. Stable flows have also been defined for more complex types of network flows, such as flows over time [28] and multicommodity flows [67].

Our contribution and structure. In this chapter we discuss three problems defined on stable flow instances of different complexity. In Section 6.3 we present a polynomial version of the Gale-Shapley algorithm for stable flows. We use the proposal-refusal pointer machinery known from SA to accelerate the pseudo-polynomial Gale-Shapley algorithm. Then, in Section 6.4 stable flows with forced and forbidden edges are discussed. We provide a simple combinatorial algorithm to find a flow fulfilling all constraints on restricted edges and also discuss the algorithm in the SM case. Finally, in Section 6.5 we study stable multicommodity flows. There, a stable solution is known to exist [67], but it is PPAD-hard to find one. We reduce 3-SAT to the integral stable multicommodity flow problem and show that it is NP-complete to decide whether an integral solution exists even if the network in the input has integral capacities only.

6.2 Preliminaries

A *network* (D, c) consists of a directed graph $D = (V, E)$ and a capacity function $c : E(D) \to \mathbb{R}_{\geq 0}$ on its edges. The vertex set of D is divided into two disjoint subsets: the set of *terminals* S and the set of *non-terminals* $V(D) \setminus S$.

Definition 6.1 (flow). *A function $f : E(D) \to \mathbb{R}_{\geq 0}$ is a* flow *if it fulfills both of the following requirements:*

1. *capacity constraints: $f(uv) \leq c(uv)$ for every $uv \in E(D)$;*
2. *flow conservation: $\sum_{uv \in E(D)} f(uv) = \sum_{vw \in E(D)} f(vw)$ for all $v \in V(D) \setminus S$.*

A stable flow instance is a triple $\mathcal{I} = (D, c, O)$. It comprises a network (D, c) and O, the preference ordering of vertices on their incident edges. Each non-terminal vertex ranks its incoming and also its outgoing edges strictly and separately. Similar to the previous models, if v prefers edge vw to vz, then we write $\text{rank}_v(vw) < \text{rank}_v(vz)$ or $vw >_v vz$. Terminals do not rank their edges, because their preferences are irrelevant with respect to the following definition.

Definition 6.2 (blocking walk, stable flow). *A* blocking walk *of flow f is a directed walk $\rho = \langle v_1, v_2, ..., v_k \rangle$ such that all of the following properties hold:*

1. *$f(v_i v_{i+1}) < c(v_i v_{i+1})$, for each edge $v_i v_{i+1}$, $i = 1, ..., k-1$;*
2. *$v_1 \in S$ or there is an edge $v_1 u$ such that $f(v_1 u) > 0$ and $v_1 v_2 >_{v_1} v_1 u$;*
3. *$v_k \in S$ or there is an edge $w v_k$ such that $f(w v_k) > 0$ and $v_{k-1} v_k >_{v_k} w v_k$.*

A flow is stable, *if there is no blocking walk in the graph.*

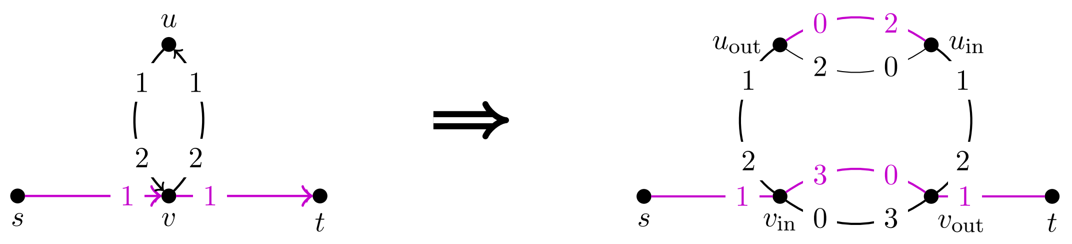


Figure 6.1: Substituting each non-terminal vertex by two parallel edges, putting them to the top and to the bottom of the original lists of incoming and outgoing edges. Edges in both instances have capacity 1. The colored edges mark the vendor-optimal solution. The allocation corresponding to the customer-optimal flow uses the other two added edges.

Walks fulfilling point 2 are said to *dominate* f at start, while walks fulfilling point 3 dominate f at the end. We can say that a walk blocks f if it dominates f at both ends.

Problem 17. SF
Input: $\mathcal{I} = (D, c, O)$; a directed network (D, c) and O, the preference ordering of vertices.
Question: Is there a flow f so that no walk blocks f?

Theorem 6.3 (Fleiner [41])**.** SF *always has a stable solution and it can be found in polynomial time.*

Fleiner's reduction to SA is based on splitting each non-terminal vertex v to v_{in} and v_{out}, and connecting them by two parallel edges, as shown in Figure 6.1. The two added edges become the first and last ranked edges of v_{in}, and the last and first ranked edges of v_{out}, respectively. Their capacity, the quota of v_{in} and v_{out} are the highest amount of flow that can pass through v. Since we are working on an SA instance, the edges lose their orientation given in the SF instance. Fleiner shows that stable allocations guarantee flow conservation at non-terminal vertices.

Due to the transformation, the lattice structure of stable allocations carries over to flows; job- and machine-optimal stable allocations are realized as vendor-optimal and the customer-optimal stable flows. Nonetheless, the hidden allocation edges – between v_{in} and v_{out} – introduce some counterintuitive properties. In Figure 6.1, the two extreme points of the lattice, the vendor-optimal and the customer-optimal stable flows are identical: $f(sv) = f(vt) = 1$. The third stable flow placed between them in the lattice is a better solution lexicographically and also in the Pareto-optimal sense: $f(sv) = f(vt) = f(vu) = f(uv) = 1$. In the allocation instance, it corresponds to the allocation between the two extreme points. The lattice operations meet and join are defined for SA using the min-min criterion. When building the join of two allocations, each job chooses its position in the allocation that is better for it according to the min-min criterion. It was shown that the resulting allocation is stable [8]. The meet works analogously, with jobs choosing their allocation edges in the worse allocation. These two operations carry over to SF, but due to the hidden allocation edges, the choices of the vertices are a lot more complex.

Besides these results, Fleiner also extended the Rural Hospitals Theorem to flows, which will be used later in our proofs.

Theorem 6.4 (Fleiner [41])**.** *For a fixed* SF *instance, each edge incident to a terminal vertex has the same value in every stable flow.*

It is necessary to discuss some technical details mainly regarding notation.

- The only characterizing property of a terminal vertex is that flow conservation does not necessarily hold for this vertex. The terms source and sink are often used in the literature. We do not assign distinguished roles to terminals with only outgoing or only incoming edges, but splitting each $s \in S$ to a source s_{out} and a sink s_{in} does not interfere with any of the previous or our results. Unless otherwise stated, we assume that terminal vertices are split in such a way and $S = S_{\text{in}} \cup S_{\text{out}}$. Introducing a super-source and a super-sink would also be possible, but it would affect the usage of Theorem 6.4, so we forgo that.

- Another technical detail is that in our setting the network (D, c) may contain parallel edges and loops. These edges can be split by a dummy vertex without changing any relevant property of the network. Formally, the number of vertices (with only one incoming and one outgoing edge) can reach $\mathcal{O}(|E|)$ after the change. We assume that all loops and parallel edges have been split in that way in our instances, because this technical change simplifies notation when it comes to denoting walks by the ordered list of vertices appearing in the walk. We would like to emphasize though that all algorithms and proofs presented here carry over for the case with loops and parallel edges without any essential modification.

- There is no difference between a vertex setting up a single list of all of its incident edges – as originally defined by Fleiner – or the same vertex keeping two separate lists: one on the incoming and one on the outgoing edges. The rank of an incoming and an outgoing edge are never compared to one another.

- We also emphasize that we do not interpret the directed edge uv as edge vu with $f(vu) = -f(uv)$, as it is common when dealing with maximum flow problems, due to the notion of residual networks. When talking about edge uv, we mean the edge directed from u to v.

Even though any unsaturated terminal-terminal walk blocks a flow, the size of the flow $|f|$ of stable flows can be much smaller than the size of some maximum flows in the network. Flows with no unsaturated terminal-terminal paths are *maximal* flows. We know that every stable flow is maximal and it is folklore that the ratio of the size of maximal and maximum flows can be of $\mathcal{O}(|E|)$. As the instance in Figure 6.2 demonstrates, this ratio can also be achieved by stable and maximum flows in a graph.

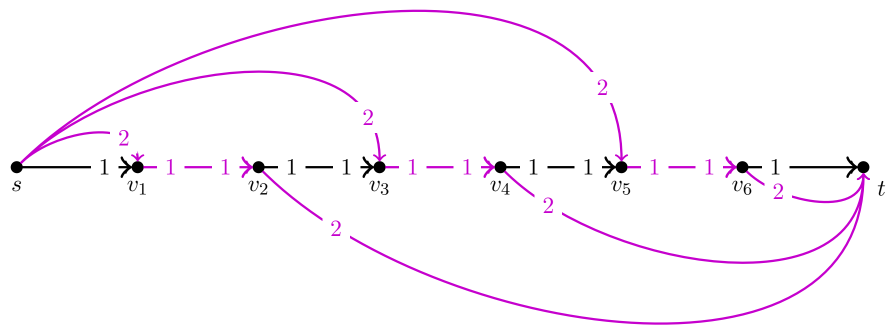


Figure 6.2: The maximum flow (marked by colored edges) has value 3, while the unique stable flow is of value 1 and is sent along the path $\langle s, v_1, v_2, ..., t\rangle$. It is easy to see that this instance can be extended to demonstrate ratio $\Omega(|E|)$.

6.3 A polynomial algorithm for stable flows

6.3.1 Known algorithms for stable flows

As mentioned in Section 6.1, the best currently known computation time of a stable flow is $\mathcal{O}(|E| \log |V|)$, due to Fleiner's reduction to SA. The by far most common solution method for stability problems, the Gale-Shapley algorithm, can also be extended to SF [28]. At start, all terminal vertices $s \in S$ send $c(sv)$ flow on all outgoing edges sv. The vertices that have received flow, gain some *overflow*: the amount of excess flow they still need to forward or reject, if needed. Each vertex v with positive overflow now decides whether it is able to forward the flow on its outgoing edges. If so, v passes on as much flow as it can along its outgoing edges, in the order of its preferences. The rest of the overflow is then refused along the worst incoming edges of v, sending back overflow to the vertex making the offer. Thus v reaches 0 overflow. In each step of the algorithm, vertices with positive overflow submit offers along their best edges, and receivers accept or refuse them as long as there is any overflow at non-terminal vertices.

In [28], two variants of this algorithm are presented: a simultaneous version with collective proposal and refusal rounds, and a preflow-push variant, where a single proposal is immediately accepted and forwarded or rejected. In this latter version, one terminal is activated at a time to send off flow, until the overflow becomes 0, then the next terminal becomes active and so on. The two variants resemble the characteristics of breath-first search and depth-first search, respectively. Their output is the same stable flow.

Note that the Gale-Shapley algorithm runs in pseudo-polynomial time already for SA instances, and the above mentioned extension to SF also requires pseudo-polynomial time to terminate. Here we present a polynomial time algorithm to produce a stable flow. Our method is based on the augmenting path algorithm of Baïou and Balinski [7] and Dean and Munshi [34]. The main idea is to introduce proposal and refusal pointers to keep track of possible Gale-Shapley steps and execute them in bulk. Note that in Section 4.5 we used a similar approach to speed up the better-response algorithm on uncoordinated allocation instances.

6.3.2 Our algorithm

We work in a helper graph $H(f)$ and associate a function $x : E(H(f)) \to \mathbb{R}_{\geq 0}$ with the current flow f. The set of vertices $V(D)$ remains intact in $H(f)$. The edges of $H(f)$ are the current proposal and refusal pointers as defined below. The main idea of the algorithm is to augment along walks of proposal and refusal pointers, thus executing several Gale-Shapley steps simultaneously.

At start, each vertex $v \in V(D) \setminus S$ is assigned a single *proposal pointer* $p(v) \in E(D)$, pointing to v's first choice outgoing edge. During the course of our algorithm, $p(v)$ moves towards the bottom of v's preference list on its outgoing edges. Once it reaches the last element and this edge becomes saturated, it turns into a *refusal pointer*. The first refusal edge is the worst incoming edge of v. From this point on, the refusal pointer $r(v) \in E(D)$ takes a step upwards on v's preference list on incoming edges.

Terminal vertices add all their outgoing edges to the set of proposal edges. When such a proposal edge is fully refused, the terminal vertex does not get refusal edges, because terminal vertices have no incentive to refuse any flow.

While proposal edges preserve their orientation in $H(f)$, refusal edges are represented as backward edges. Due to this, each non-terminal vertex has at most one outgoing edge in $H(f)$, but it can have several incoming edges. If v has a proposal edge or $v \in S_{\text{in}}$, we say that $v \in \mathcal{P}$, otherwise $v \in \mathcal{R}$. Our algorithm performs augmenting steps along walks of proposal and refusal edges, always starting with a proposal edge. In order to specify the amount of flow we augment with, we allocate x values to each pointer, that is, capacities to each edge in $H(f)$. We define x in the following way.

$$\begin{aligned} x(p(v)) &= c(p(v)) - f(p(v)) \\ x(r(v)) &= f(r(v)) \end{aligned}$$

Pointers p and r step further on the preference lists in two cases. Firstly, if $x(p(v))$, or $x(r(v))$, respectively, reaches 0, then the pointer leaves the current edge and is shifted to the next edge as described above. In addition, any edge that has become a refusal pointer at any point cannot be a proposal pointer of its starting vertex, as the proposal pointer immediately steps further in such cases.

In one step of the algorithm a walk starting with a proposal edge at a terminal vertex is chosen. Since each vertex has at most one outgoing pointer in $H(f)$, this walk can be of two types: it is either a path between two terminal vertices or a union of a path and a cycle: it ends in one of its already listed vertices. We augment along the entire path, or, in the second case, along the cycle with the smallest x capacity along it. After this augmentation the pointers have to be updated, because at least one of them moves further.

The process terminates when there is no walk to augment along. Since the pointers are monotone and each augmentation moves at least one pointer, this point is reached in $\mathcal{O}(|E|)$ rounds. In each round we need to find a walk to augment along, compute the lowest x value along it and then update the pointers. This can be done in $\mathcal{O}(|V|)$ time, although it is highly likely that sophisticated data structures allow one to implement these steps in $\mathcal{O}(\log |V|)$ time in the same manner as for SA.

Theorem 6.5. *The output of the algorithm f is a stable flow.*

Proof. We start our proof with the following observations that are immediate consequences of the monotonicity of p and r.

Observation 1. *If $f(uv_1) < c(uv_1)$ and $f(uv_2) > 0$ for some $uv_1 >_u uv_2$, then $p(u)$ has passed through uv_1 and was rejected by v_1.*

Observation 2. *If $f(v_1u) < c(v_1u)$ and $f(v_2u) > 0$ for some $v_1u >_u v_2u$, then $r(u)$ has not reached v_1u and thus the reason for $f(v_1u) < c(v_1u)$ is that v_1 never proposed along v_1u with $c(v_1u)$.*

Assume there is a walk $\rho = \langle v_1, v_2, \ldots, v_k\rangle$ blocking f. Due to point 2 in Definition 6.2, either $v_1 \in S$ or $v_1v_2 >_{v_1} v_1u$ for some $v_1u \in E(D)$ with $f(v_1u) > 0$, thus the conditions of Observation 1 are fulfilled. Since $v_1 \in S$ or $p(v_1)$ passed through the unsaturated v_1v_2, flow along v_1v_2 was rejected by v_2. Therefore, $v_2 \in \mathcal{R}$. If $\rho = \langle v_1, v_2\rangle$, then v_2 rejected a better edge than its worst edge with positive flow on it, which is a contradiction. Similarly, if $\rho = \langle v_1, v_2, v_3\rangle$ and we already know that $v_2 \in \mathcal{R}$, then v_2 must have proposed along v_2v_3, but got rejected by v_3, even though v_3 has a worse edge with positive flow value. For cases $k \geq 4$ we will now show that there is a saturated edge along ρ.

Claim 34. *Along ρ there is at least one edge v_iv_{i+1} with $v_i \in \mathcal{R}$ and $v_{i+1} \in \mathcal{P} \cup S$, for some $1 \leq i \leq k-1$.*

Proof. We already know that $v_2 \in \mathcal{R}$ and if $v_k \in \mathcal{P} \cup S$, we are done. Consider now $v_k \in \mathcal{R}$. According to point 3 in Definition 6.2, either $v_k \in S$ or $v_{k-1}v_k >_{v_k} uv_k$ for some $uv_k \in E(D)$ with $f(uv_k) > 0$. In the second case, the conditions of Observation 2 are fulfilled; thus v_{k-1} has never proposed along $v_{k-1}v_k$ with $c(v_{k-1}v_k)$, that is, $v_{k-1} \in \mathcal{P}$. To summarize this, there is an edge along ρ with $v_i \in \mathcal{R}$ and $v_{i+1} \in \mathcal{P}$. ■

Claim 35. *For every $uv \in E$ with $u \in \mathcal{R}$ and $v \in \mathcal{P} \cup S$, $f(uv) = c(uv)$.*

Proof. Assume that $f(uv) < c(uv)$ for some uv with $u \in \mathcal{R}$ and $v \in \mathcal{P} \cup S$. Since $u \in \mathcal{R}$, u made a proposal with $c(uv)$ along uv during the course of the algorithm. This proposal must have been rejected by v, because $f(uv) < c(uv)$. Then $v \in \mathcal{R}$ as well. ■ □

6.4 Stable flows with restricted edges

In this section we translate the notion of restricted edges (introduced in Chapter 2) to SF. While we discussed mainly approximation concepts for SM and SR, here we present a direct algorithm to solve SF with restricted edges.

Restricted edges in SF can be motivated similarly as in SM and SR. If a certain deal is for some reason particularly important, or to the contrary, not wished by the majority to be realized, but the vendors participating in the deal are free to act according to their will, stable flows with restricted edges correspond to solutions acceptable for all market participants.

In SF, where edges have capacities, the notion of a restricted edge as known from Chapter 2 requires revision. We need to model more complex constraints than in SM, such as a stable flow being acceptable only if its flow value is between two given values. A *lower capacity* function $\ell : E(D) \to \mathbb{R}_{\geq 0}$ and an *upper capacity* function $\mathfrak{u} : E(D) \to \mathbb{R}_{\geq 0}$ are introduced.

Problem 18. SF RESTRICTED
Input: $\mathcal{I} = (D, c, O, \ell, \mathfrak{u})$*; an* SF *instance* (D, c, O)*, a lower capacity function* $\ell : E(D) \to \mathbb{R}_{\geq 0}$ *and an upper capacity function* $\mathfrak{u} : E(D) \to \mathbb{R}_{\geq 0}$.
Question: Is there a stable flow f *so that* $\ell(uv) \leq f(uv) \leq \mathfrak{u}(uv)$ *for all* $uv \in E(D)$*?*

If $\mathfrak{u}(uv) = c(uv)$ for all $uv \in E(D)$, then we call the problem SF FORCED, and analogous, if $\ell(uv) = 0$ for all $uv \in E(D)$, then we talk about SF FORBIDDEN. Note that the introduced upper capacity $\mathfrak{u}$ is not equivalent to decreasing $c(uv)$, because blocking walks can use uv even if $f(uv) = \mathfrak{u}(uv) < c(uv)$ holds.

If $\ell(uv) > \mathfrak{u}(uv)$ for some $uv \in E(D)$, then SF RESTRICTED trivially has no solution. Otherwise, we execute a small technical change for the sake of simpler proofs later. As shown in Figure 6.3, we substitute each edge $uv \in E(D)$ with three parallel paths (to avoid parallel edges): $\langle u, x, v\rangle$, $\langle u, y, v\rangle$ and $\langle u, z, v\rangle$. While uy and yv take over the rank of uv, ux and xv are ranked just above, uz and zv are ranked just below uy and yv. The capacities of the introduced edges are as follows.

$$\begin{aligned}
&\ell(ux) = \ell(xv) = \mathfrak{u}(ux) = \mathfrak{u}(xv) = c(ux) = c(xv) = \ell(uv)\\
&\ell(uy) = \ell(yv) = 0\\
&\mathfrak{u}(uy) = \mathfrak{u}(yv) = c(uy) = c(yv) = \mathfrak{u}(uv) - \ell(uv)\\
&\ell(uz) = \ell(zv) = \mathfrak{u}(uz) = \mathfrak{u}(zv) = 0\\
&c(uz) = c(zv) = c(uv) - \mathfrak{u}(uv)
\end{aligned}$$

In words, this splitting is substituting an edge uv with lower and upper capacities by three paths: the first path $\langle u, x, v\rangle$ forcing $\ell(uv)$ amount of flow, the last path $\langle u, z, v\rangle$ forbidding $\mathfrak{u}(uv)$ amount of flow and the middle path $\langle u, y, v\rangle$ being an unrestricted path. It is straightforward to see that the solutions of SF RESTRICTED on the original instance are in one-to-one correspondence with the solutions of SF RESTRICTED on the modified instance and the input size is increased by at most a constant factor. From

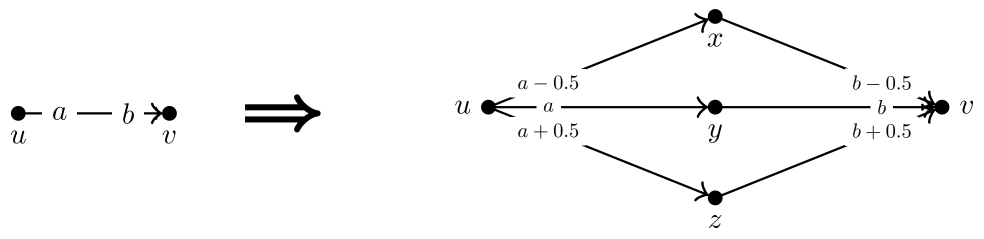


Figure 6.3: Splitting an edge with lower and upper capacities. Due to the preferences and capacities defined on the modified instance, the first $\ell(uv)$ units of flow will saturate $\langle u, x, v\rangle$, then, the coming $\mathfrak{u}(uv) - \ell(uv)$ units of flow will saturate $\langle u, y, v\rangle$, and the remaining $c(uv) - \mathfrak{u}(uv)$ units of flow will use $\langle u, z, v\rangle$.

this point on we assume that the input is given so that $\ell(uv), \mathfrak{u}(uv) \in \{0, c(uv)\}$ for all $uv \in E(D)$. This also allows us to use identical notation to the one in Chapter 2, we use $Q = \{uv \in E | \ell(uv) = c(uv)\}$ and $P = \{uv \in E | \mathfrak{u}(uv) = 0\}$.

Here we describe a polynomial algorithm that finds a stable flow with forced and forbidden edges or proves its nonexistence. We show that forced and forbidden edges can be handled with the help of reductions to unrestricted SF. We would like to emphasize that it is rather straightforward to see that SF RESTRICTED can be solved by transforming the SF RESTRICTED instance first into a weighted SF, and then into a weighted SA instance, both solvable in $\mathcal{O}(|E|^2 \log |V|)$ time [34]. The advantages of our method are that it can be applied directly to the SF RESTRICTED instance and it also gives us insights to solving SR RESTRICTED directly, as pointed out at the end of Sections 6.4.1 and 6.4.2. Moreover, our running time is only $\mathcal{O}(|E| \log |V|)$. The coming two sections are organized in the same manner: first, the case with a single special edge is studied, then the solution for the general case is described.

6.4.1 Forced edges

A single forced edge

We are given an SF FORCED instance with $Q = \{uv\}$. We modify graph D to derive a helper graph D_{st}. This modification consists of deleting the forced edge uv and introducing two new edges to substitute it. One of them starts at a new terminal vertex s and ends at v, the other edge starts at u and ends at a new terminal t. They both have capacity $c(uv)$ and take over uv's rank on u's and on v's preference lists, as shown in Figure 6.4.

Lemma 6.6. *There is a stable flow f in D with $f(uv) = c(uv)$ if and only if there is a stable flow f_{st} in D_{st} with $f_{st}(sv) = f_{st}(ut) = c(sv) = c(ut)$.*

Proof. Assume first that there is a stable flow f in D with $f(uv) = c(uv)$. The stable flow in D_{st} is constructed by copying f to the new graph and replacing $f(uv) = c(uv)$ by $f_{st}(sv) = f_{st}(ut) = f(uv) = c(sv) = c(ut)$. Comparing the dominance situation at

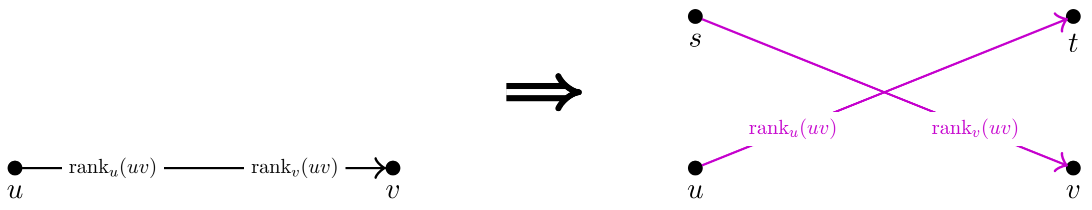


Figure 6.4: Substituting forced edge uv by edges sv and ut in D_{st}.

vertices, it is straightforward that there is no blocking walk to f_{st} that did not block f, because the unsaturated edges are exactly the same in both flows.

To show the opposite direction, suppose that there is a stable flow f_{st} in D_{st} with $f_{st}(sv) = f_{st}(ut) = c(sv) = c(ut)$. If $f_{st}(sv) = f_{st}(ut)$, then feasibility is kept while replacing sv and ut by uv and setting $f(uv)$ to $f_{st}(sv)$. Just as before, the dominance situation remains unchanged. □

Recall Theorem 6.4: testing whether a stable flow exists with specific values on edges incident to terminals can be done in polynomial time. It is sufficient to find any stable flow, because the flow value on sv and ut edges is the same for all stable flows. Lemma 6.6 together with Theorem 6.4 guarantees that SF FORCED with $|Q| = 1$ can be solved simply by computing any stable flow in D_{st} in $\mathcal{O}(|E| \log |V|)$ time.

General case

Being able to handle a single forced edge we are ready to consider the case $|Q| \geq 2$. Substituting more than one forced edge in a manner described above can be done independently, without the forced edges impacting each other. Thus, introducing terminal vertices s and t, deleting all edges in Q and substituting each one of them by two edges, one from s and one to t results in a classical stable flow instance on D_{st}. For this instance, the following theorem holds:

Theorem 6.7. *There is a stable flow f in D with $f(uv) = c(uv)$ for all $uv \in Q$ if and only if there is a stable flow f_{st} in D_{st} with $f_{st}(sv) = f_{st}(ut) = c(sv) = c(ut)$ for each $uv \in Q$.*

Just as in the $|Q| = 1$ case, due to Theorem 6.4, it is enough to compute any stable flow in D_{st} to solve SF FORCED. Thus, SF FORCED can be solved in $\mathcal{O}(|E| \log |V|)$ time.

Our results provide a fairly simple method for SM and SR with forced edges, because the Rural Hospitals Theorem holds for those two cases as well. After deleting each forced edge $uw \in Q$ from the graph, we add uw_s and $u_t w$ edges to each of the pairs. They take over the rank of uw. Unlike in SF, here we need to introduce a separate dummy vertex to each forced edge, simply due to the matching constraints. There is a stable matching containing all forced edges if and only if an arbitrary stable matching covers all of these new vertices w_s and u_t. The running time of this algorithm is $\mathcal{O}(|E|)$, since it is sufficient to construct a single stable solution in an instance with at most $2|V|$ vertices.

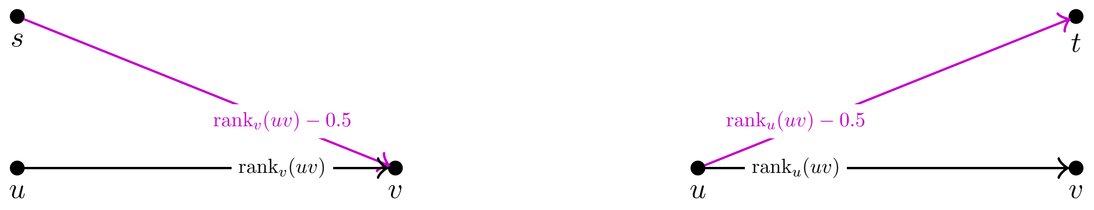


Figure 6.5: Adding edges sv in D_s and ut in D_t to forbidden edge uv.

More vertices cannot occur, because in SM and SR more than one forced edge incident to a vertex immediately implies infeasibility.

6.4.2 Forbidden edges

In order to handle SF FORBIDDEN, we present here an argumentation of the same structure as in the previous section. First, the problem of stable flows with a single forbidden edge is solved and then an algorithm for the general case is described.

A single forbidden edge

Assume that $P = \{uv\}$. First we present two modified instances that will come handy when solving SF FORBIDDEN. The first construction produces graph D_s, by adding a terminal vertex s to $V(D)$ and an edge sv to $E(D)$. We set $c(sv)$ to an arbitrary positive number. The rank of sv on v's preference list is better than the rank of uv, but it is worse than all edges better than uv in D. The second construction is similar to the first one. Graph D_t differs from D in one terminal vertex t and an edge ut. The capacity of ut is positive and ut is right before uv on u's preference list. Both graphs are illustrated in Figure 6.5.

In the following, we characterize SF FORBIDDEN instances with the help of D_s and D_t. Our claim is that SF FORBIDDEN in D has a solution if and only if there is a stable flow f_s in D_s with $f_s(sv) = 0$ or there is a stable flow f_t in D_t with $f_t(ut) = 0$. The latter existence problem can be solved easily in polynomial time, since all stable flows have the same value on edges incident to terminal vertices, as Theorem 6.4 states.

Lemma 6.8. *There is a stable flow f in D with $f(uv) = 0$ if and only if at least one of the following holds:*

Property 1: there is a stable flow f_s in D_s with $f_s(sv) = 0$ or

Property 2: there is a stable flow f_t in D_t with $f_t(ut) = 0$.

Proof. Assume first that there is a stable flow f in D with $f(uv) = 0$. Since f is stable and uv is unsaturated, every unsaturated walk passing through uv dominates the flow on at most one end. Depending on which end it is, we will construct flow f_s or f_t in either of the modified graphs.

Suppose there is no unsaturated walk containing uv that dominates f at its end. In this case, D_s is chosen: $f_s = f$ on all edges, except $f_s(sv) = 0$. Since sv is the only edge that can dominate f_s but does not dominate f, all walks that possibly block f_s must contain sv. The edge sv does not block f_s, because we assumed that no unsaturated walk ending at v dominates f at v, not even uv. Thus v has no incoming edge with positive flow value that is worse than uv (or, equivalently sv). Moreover, since there is no unsaturated walk starting at v that dominates f (and f_s) at its end, f_s is stable.

In the remaining case, when there is no unsaturated walk that dominates f at its beginning, f_t in D_t is constructed. Similarly as above, $f_t = 0$ on ut and $f_t = f$ on all other edges. If there is a blocking walk to f_t, it must pass through ut. But our assumption implies that u's worst outgoing edge with positive f-value is better than uv and that there is no dominating walk ending at u. Thus, neither ut nor any other walk can block f_t.

Now we show the opposite direction. Suppose Property 1 is fulfilled. Since $f_s(sv)$=0 and sv dominates uv, $f_s(uv)$ must be 0 for all stable flows. We construct f from f_s simply by omitting $f_s(sv)$. For this f, the equality $f(uv) = 0$ holds. Moreover, f is stable, since no edge has less flow than in f_s and no edge became dominant to f that did not dominate f_s. An analogous argumentation holds for the case of Property 2. □

General case

The method described above solves SF FORBIDDEN with $|P| = 1$ in $\mathcal{O}(|E| \log |V|)$ time. However, if $|P| > 1$ it is not straightforward how to find sv or ut edges to all forbidden edges. Applying our method greedily for each forbidden edge does not lead to correct results, since the steps can impact each other. An immediate consequence of Lemma 6.8 is that if there is a stable flow in the network with forbidden edges, we can add an sv or a ut edge to each of them so that $f(sv) = f(ut) = 0$ for all of them. However, finding it with the same procedure may require $2^{|P|}$ steps. In the following, we outline a polynomial algorithm that solves SF FORBIDDEN.

We start with two rather straightforward observations that we will refer to several times later. The first one essentially says that deleting edges not used by stable flows from the network cannot create new blocking walks.

Observation 3. *If $f(uv) = 0$ for an edge $uv \in E(D)$ and stable flow f in D, then f remains stable in $D \setminus uv$ as well.*

Another key observation is that during the execution of the Gale-Shapley algorithm, adding more edges from sources to the graph cannot result in more flow on edges already coming from terminals or in less flow on edges running to terminals.

Observation 4. *Let f denote the output of the Gale-Shapley algorithm executed on $\mathcal{I} = (D, c, O)$, where $s'v' \notin E(D)$. For all stable flows f' in $\mathcal{I}' = (D' = D \cup s'v', c', O)$ the inequalities $f(sv) \geq f'(sv)$ and $f(vs) \leq f'(vs)$ hold for every $sv, vs \in E(D)$ and $s \in S$.*

Proof. Since the order of proposals in the Gale-Shapley algorithm in D' does not change the output, we can first find the stable flow in D and then call propose on v'. The proposal-refusal steps started from v' can never increase $f(sv)$, because s is not an active terminal any more and they cannot decrease any $f(vs)$, because terminal vertices do not refuse already existing flow. □

Note that Observation 4 is in general true for any stable flow, not only the output of the Gale-Shapley algorithm. These two basic observations come handy when analyzing the output of the following algorithm for SF FORBIDDEN.

At start, we fix $P_0 = P$ and test whether Property 1 in Lemma 6.8 is fulfilled for edges in P_0. To that end, we add sv to all $uv \in P_0$ and find a stable flow. We say that uv fails the first test if $f(sv) > 0$ for stable flows. We do the same with Property 2 in Lemma 6.8: after adding ut to all $uv \in P_0$ to the original graph D, we find a stable flow. Forbidden edges failing the first test are taken out from P_0 and placed into set P_1, while edges failing the second test are moved to P_2. If an edge fails both tests, then we terminate the algorithm claiming that there is no stable flow avoiding all forbidden edges. Otherwise, at the end of the first round, we check stable flows in the new network D, extended with sv edges for all $uv \in P_1$ and ut edges for all $uv \in P_2$. If they leave all added edges empty (with flow value 0), we proceed to round 2, otherwise no stable flow with forbidden edges exist.

When continuing testing, we require for edges already added to P_1 and P_2 that their sv and ut edges are empty in stable flows. We start testing the remaining forbidden edges in P_0. Note that an edge that passed both tests in the previous round might fail them, now that some edges were put in P_1 or P_2. As in the first round, edges failing exactly one test are put into the corresponding set, and stable flows in the extended network are checked.

The algorithm terminates if there is no edge in P_0 failing any test. This includes the case of P_0 becoming empty. In each round, three stable flows are computed in $\mathcal{O}(|E| \log |V|)$ time. The number of rounds cannot exceed $|P|$, because if the network is not changed in a round, then the algorithm terminates. Thus, our running time is $\mathcal{O}(|P||E| \log |V|)$.

Before showing how to produce a stable flow from sets P_0, P_1 and P_2, we justify that if our algorithm outputs that there is no solution for SF FORBIDDEN, then indeed no stable flow avoids all forbidden edges. Such an output occurs in two cases.

- If both tests fail for an edge uv.
- If after fixing sets P_1 and P_2, stable flows do not avoid all added edges.

In the following lemma we will show that failing a test for sv means that the only chance for the existence a stable flow avoiding uv is adding ut to uv and vice versa. Thus, if either of the two cases above appear, no stable flow avoids uv.

Lemma 6.9. *If a test fails for some sv (or ut, respectively) edge added to $uv \in E(D)$, then no stable flow exists that avoids all forbidden edges and also avoids sv (or ut, respectively) in $D \cup sv$ (or $D \cup ut$, respectively).*

Proof. Assume first that an edge uv fails the first test in a round, when sv edges are added to D. Assume inductively that the edges already fixed in P_1 and P_2 must have an sv edge or ut edge, respectively, added to reach a stable flow avoiding them. Due to Observation 3, applied for added sv edges with $f(sv) = 0$ and Observation 4 applied for added sv edges with $f(sv) > 0$, the test would have failed even if the other sv edges added in this round had not been present in the graph. Thus, there is no stable flow avoiding the edges already in $P_1 \cup P_2$ and avoiding sv as well. The proof is analogous for the case of a test failing for some ut edge. □

Now we show that if the algorithm has not outputted that no stable flow avoiding all forbidden edges exists, then there is indeed one and it is straightforward to output. When this phase terminates, P can be partitioned into three disjoint sets: P_0, P_1 and P_2. The following holds for each $u_0v_0 \in P_0$: in the graph extended with the sv edges of all edges in P_1 and with the ut edges of all edges in P_2, there is a stable flow f so that $u_0v_0 \in P_0$ is not part of any unsaturated walk dominating f at start and there is a stable flow f' so that $u_0v_0 \in P_0$ is not part of any unsaturated walk dominating f' at the end.

We claim that either adding a ut edge to all vertices in P_0 or adding an sv edge to all of them both yield graphs with a stable flow leaving all added edges empty. This is due to the join and meet operations defined on the lattice of stable flows. We demonstrate here with the help of the transformed SA instance how the join is built. When the testing phase halts, there is a stable flow in D for each $u^0v^0 \in P_0$ so that

1. $f(u'v) = 0$ for every $uv \in P_1$ and for all edges $u'v \in E(D)$ with $u'v <_v uv$;
2. $f(uv') = 0$ for every $uv \in P_2$ and for all edges $uv' \in E(D)$ with $uv' <_u uv$;
3. $f(u'v^0) = 0$ for all edges $u'v^0 \in E(D)$ with $u'v^0 \leq_{v^0} uv^0$.

Note that these hold for stable flows in D, not just in extended networks D_s or D_t. If we take the join of these stable flows for each u^0v^0 in P_0, it will again be a stable flow. When constructing the join, each vertex chooses his position regarding its incoming edges in a flow that is best for it, as defined in [41]. It is easier to follow these choices on the SA instance corresponding to the SF instance. For forbidden edges in P_1 and P_2, none of these stable flows uses them, thus, the join will also avoid them. For edges in P_0, every u_{out} gets to chose the best allocation, while every u_{in} receives the worst allocation. Since for each u^0v^0 in P_0 there is a stable flow f so that $f(u'v^0) = 0$ for all edges $u'v^0 \leq_{v^0} uv^0$, the best position of u^0_{out} will also have $u'v^0 \leq_{v^0} uv^0$ for each u^0v^0. Thus $f_{join}(uv) = 0$.

With this algorithm we have proved the following theorem.

Theorem 6.10. SF RESTRICTED *can be solved in* $\mathcal{O}(|P||E|\log|V|)$ *time.*

SF RESTRICTED can also be solved via weighted SA, but that method is less efficient. As mentioned in Section 6.1, SF instances can be converted into SA instances. The maximum weight SA problem was solved by Dean and Munshi [34]. By using rotations, they prove that an optimal solution can be found in $\mathcal{O}(|E|^2 \log|V|)$ time. In our problem, if both forced and forbidden edges are present in an SF RESTRICTED instance, the following strategy leads to a solution. First, the instance is converted into an SA with forced and forbidden edges instance. Then, weight is assigned to each edge: 1 to forced edges, -1 to forbidden edges and 0 to the all remaining edges. If a stable allocation with forced and forbidden edges exists, it is also a maximum weight stable allocation. Therefore, this method also answers the question whether SF RESTRICTED is solvable. Since $P \subseteq E(D)$, our running time cannot be worse.

Just like in the previous subsection, we finish this part with the direct interpretation of our results in SR and SM instances. To each forbidden edge $uw \in P$ we introduce edges uw_s or $u_t w$. According to the preference lists, they are slightly better than uw itself. A stable matching with forbidden edges exists, if there is a suitable set of these uw_s and $u_t w$ edges such that all w_s and u_t are unmatched. Our algorithm for several forbidden edges runs in $\mathcal{O}(|P||E|)$ time, because computing stable solutions in each of the $|P|$ or less rounds takes only $\mathcal{O}(|E|)$ time in SM. With this running time, it is somewhat slower than the best known method [36] that requires only $\mathcal{O}(|E|)$ time, but it is a reasonable assumption that the number of forbidden edges is small.

6.5 Stable multicommodity flows

6.5.1 Problem definition

In many flow-based applications, various goods are exchanged. Such problems are usually modeled by multicommodity flows [58]. Many multicommodity flow problems admit only fractional solutions even if their input is integral. The maximum multicommodity flow problem can be solved in strongly polynomial time [98], but finding a maximum integer multicommodity flow is already an NP-hard task [47]. In this section we will outline the results already achieved in the topic of multicommodity stable flows, including the existence of a stable solution. However, there is no method known for finding one. We show that it is NP-complete to decide whether an integer stable multicommodity flow exists.

A *multicommodity network* $(D, c^i, c), 1 \leq i \leq n$ consists of a directed graph $D = (V, E)$, non-negative commodity capacity functions $c^i : E(D) \to \mathbb{R}_{\geq 0}$ for all the n commodities and a non-negative cumulative capacity function $c : E(D) \to \mathbb{R}_{\geq 0}$ on $E(D)$. For every commodity i, $S^i \subseteq V(D)$ is the set of *terminals for commodity* i.

Definition 6.11 (multicommodity flow). *A set of functions $f^i : E(D) \to \mathbb{R}_{\geq 0}$, $1 \leq i \leq n$ is a* multicommodity flow *if it fulfills all of the following requirements:*

1. *capacity constraints for commodities:*
   $f^i(uv) \leq c^i(uv)$ *for all* $uv \in E(D)$ *and commodity* i*;*

2. *cumulative capacity constraints:*
   $f(uv) = \sum_{1 \leq i \leq n} f^i(uv) \leq c(uv)$ *for all* $uv \in E(D)$*;*

3. *flow conservation:*
   $\sum_{uv \in E(D)} f^i(uv) = \sum_{vw \in E(D)} f^i(vw)$ *for all* $v \in V(D) \setminus S^i$*.*

The concept of stability was extended to multicommodity flows by Király and Pap [67]. A multicommodity stable flow instance $\mathcal{I} = (D, c^i, c, O_E, O_V^i), 1 \leq i \leq n$ comprises a network $(D, c^i, c), 1 \leq i \leq n$, *edge preferences* O_E over commodities, and *vertex preferences* $O_V^i, 1 \leq i \leq n$ over incident edges for commodity i. Each edge uv ranks all commodities with $c^i(uv) > 0$ in a strict order of preference. Separately for every commodity i, each non-terminal vertex ranks its incoming and also its outgoing edges strictly with respect to commodity i. Note that these preference orderings of v can be different for different commodities and they do not depend on the edge preferences (over commodities) of the ranked edges. If edge uv prefers commodity i to commodity j, then we write $i >_{uv} j$. Analogously, if vertex v prefers edge vw to vz with respect to commodity i, then we write $vw >_v^i vz$. We denote the flow value with respect to commodity i by $f^i = \sum_{s \in S^i} \sum_{u \in V(D)} f^i(su)$.

Definition 6.12 (stable multicommodity flow)**.** *A directed walk* $\rho = \langle v_1, v_2, ..., v_k \rangle$ blocks *flow* f with respect to commodity i *if all of the following properties hold:*

1. $f^i(v_jv_{j+1}) < c^i(v_jv_{j+1})$ *for each edge* v_jv_{j+1}, $j = 1, ..., k-1$;
2. $v_1 \in S^i$ *or there is an edge* v_1u *such that* $f^i(v_1u) > 0$ *and* $v_1v_2 >_{v_1}^i v_1u$;
3. $v_k \in S^i$ *or there is an edge* wv_k *such that* $f^i(wv_k) > 0$ *and* $v_{k-1}v_k >_{v_k}^i wv_k$;
4. *if* $f(v_jv_{j+1}) = c(v_jv_{j+1})$, *then there is a commodity* i' *such that* $f^{i'}(v_jv_{j+1}) > 0$ *and* $i >_{v_jv_{j+1}} i'$.

A multicommodity flow is stable, *if there is no blocking walk with respect to any commodity in the graph.*

Note that due to point 4, this definition allows saturated edges to occur in a blocking walk with respect to commodity i, provided that these edges are inclined to trade in some of their forwarded commodities for more flow of commodity i. On the other hand, the role of edge preferences is limited: blocking walks still must start at vertices who are willing to reroute or send extra flow along the first edge of the walk according to their vertex preferences with respect to commodity i.

Problem 19. MSF
Input: $\mathcal{I} = (D, c^i, c, O_E, O_V^i)$, $1 \leq i \leq n$; *a directed multicommodity network* (D, c^i, c), $1 \leq i \leq n$, *edge preferences over commodities* O_E *and vertex preferences over incident edges* $O_V^i, 1 \leq i \leq n$.
Question: Is there a multicommodity flow f *so that no walk blocks* f *with respect to any commodity?*

Theorem 6.13 (Király, Pap [67])**.** *A stable multicommodity flow exists for any instance, but it is* PPAD*-hard to find.*

PPAD-hardness is a somewhat weaker evidence of intractability than NP-hardness [83]. Király and Pap use a polyhedral version of Sperner's lemma [66] to prove this existence result. Note that MSF is one of the very few problems in stability where a stable solution exists, but no extension of the Gale-Shapley algorithm is known to solve it (not even a variant with exponential running time).

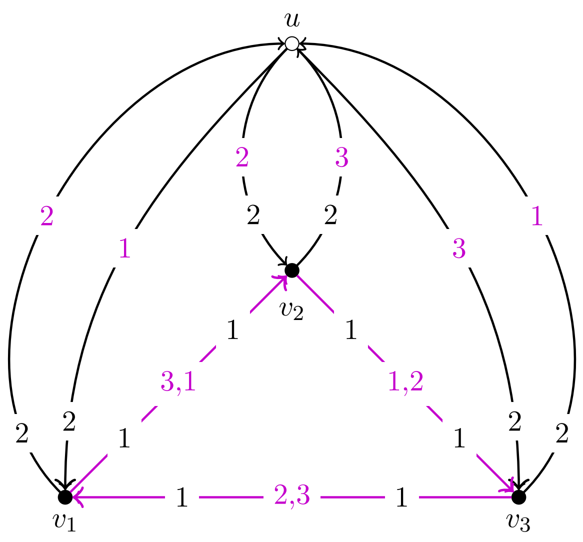


Figure 6.6: The edge preferences are marked with colored labels in the middle of edges, while O^i_V is black and closer to the vertices. For all edges, $c = 1$. The purple edges can forward two commodities, while the black edges can carry only one commodity.

6.5.2 Integral multicommodity stable flows

In this section, we investigate the integral version of MSF.

Problem 20. IMSF
Input: $\mathcal{I} = (D, c^i, c, O_E, O^i_V), 1 \leq i \leq n$; *a directed multicommodity network* $(D, c^i : E(D) \to \mathbb{Z}_{\geq 0}, c : E(D) \to \mathbb{Z}_{\geq 0}), 1 \leq i \leq n$*, edge preferences over commodities* O_E *and vertex preferences over incident edges* $O^i_V, 1 \leq i \leq n$.
Question: Is there an integer multicommodity flow $f : E(D) \to \mathbb{Z}_{\geq 0}$ *so that no walk blocks* f *with respect to any commodity?*

For the single commodity case, Fleiner [41] shows that IMSF always has a solution and it can be found efficiently. In contrast, there are IMSF instances with no integer solution. Király and Pap give an example instance with N commodities and N vertices, where no stable multicommodity flow exists with denominators at most N for any integer N. Here we present a small and slightly modified version of that instance as an example and later use it as a gadget in our hardness proof.

Example 6.14. IMSF *instance with no solution.*

We claim that the instance depicted in Figure 6.6 admits no integer multicommodity flow. Vertex u is the only terminal vertex in the graph. Below we will distinguish two cases: 1) $S^1 = S^2 = S^3 = \{u\}$ (see Lemma 6.15) and 2) $\exists i \in \{1, 2, 3\} : S^i = \emptyset$ (see Lemma 6.16). The edge capacities with respect to commodities are 1 for the commodities that appear in O_E for the specific edge and 0 for the remaining commodities. All edges have cumulative capacity 1. The vertex preferences are the same for all commodities: v_1, v_2 and v_3 are inclined to receive and send the flow along the edges between themselves rather than trading with u. Each commodity i has a unique feasible cycle C^i through u and it is easy to see that due to the choice of the c^i functions, no other cycle or terminal-terminal path exists in the network.

- $C^1 = \langle u, v_1, v_2, v_3, u\rangle$
- $C^2 = \langle u, v_2, v_3, v_1, u\rangle$
- $C^3 = \langle u, v_3, v_1, v_2, u\rangle$

Lemma 6.15. *If $S^1 = S^2 = S^3 = \{u\}$, then there is no integer stable flow.*

Proof. First we show that the following fractional flow is stable. In this flow, each commodity i where $i \in \{1, 2, 3\}$, is sent along cycle C^i with flow value $\frac{1}{2}$.

$$\begin{array}{rrrrrrr} g^3(v_1v_2) = & g^1(v_1v_2) = & g^1(v_2v_3) = & g^2(v_2v_3) = & g^2(v_3v_1) = & g^3(v_3v_1) = & \frac{1}{2} \\ g^1(uv_1) = & g^1(v_3u) = & g^2(uv_2) = & g^2(uv_1) = & g^3(uv_3) = & g^3(uv_2) = & \frac{1}{2} \end{array}$$

Claim 36. *No walk blocks g.*

Proof. It is enough to show that no walk ρ with respect to commodity 1 blocks g, as the instance is symmetric for all three commodities, and thus the nonexistence of blocking walks with respect to the other two commodities can be shown analogously. The only edges allowed to carry commodity 1 are the edges of C^1, thus $\rho \subseteq C^1$. Even though uv_1, v_2v_3 and v_3u fulfill points 1-4 in Definition 6.12, v_1v_2 is saturated and ranks commodity 1 last, violating point 4 in Definition 6.12. ■

Claim 37. *No integral multicommodity flow is stable in the instance.*

Proof. Assume that there is an integral stable flow f in the instance. The empty flow cannot be f, because there is a cycle running through u for each commodity and such cycles block the empty flow. Without loss of generality we can now assume that C^1 is saturated by commodity 1:

$$f^1(uv_1) = f^1(v_1v_2) = f^1(v_2v_3) = f^1(v_3u) = 1,$$

while all other flow values must be 0 due to commodity capacity constraints on edges. This flow is blocked by commodity 3 on the path $\langle u, v_3, v_1, v_2, u\rangle$. It is easy to see that analogous arguments work for C^2 and C^3 as well. Thus, no integer stable flow exists in the graph. ■ □

Lemma 6.16. *If u is a terminal for at most two out of the three commodities, then an integer stable flow exists.*

Proof. Let us now investigate the same instance with a slight modification: $S^1 = S^2 = \{u\}$, but $S^3 = \emptyset$. Then, the following integer flow is stable:

$$f^1(uv_1) = f^1(v_1v_2) = f^1(v_2v_3) = f^1(v_3u) = 1.$$

A blocking walk with respect to commodity 1 cannot exist, because all edges that can carry commodity 1 also carry it to their upper capacity. Commodity 2 could block along C^2, but edge v_2v_3 is saturated with its most preferred commodity. It is trivial that the same flow remains stable if we set $S^1 = u$ and $S^2 = S^3 = \emptyset$. If $S^1 = S^2 = S^3 = \emptyset$, then the empty flow is stable. □

To sum up the established results about Example 6.14: the instance admits an integer stable flow if and only if u has at most two commodities. This argument will help us prove Claim 39 later in our hardness proof.

Theorem 6.17. *Deciding whether* IMSF *has a solution is* NP*-complete. This holds even if all commodities share the same set of terminal vertices and all vertices have the same preferences with respect to all commodities (but edges might have different capacities with respect to different commodities).*

Proof. Testing whether a feasible integral multicommodity flow is stable can be done in polynomial time, as pointed out also in [67]. It is sufficient to check the existence of edges fulfilling points 2 and 3 in Definition 6.12 for every commodity and then execute a breadth-first search for every pair of vertices as v_1 and v_k vertices of the potential blocking walk. Thus IMSF is in NP.

Now we describe IMSF instances $\mathcal{I}'$ constructed to every instance $\mathcal{I}$ of 3-SAT, also illustrated in Figure 6.7. To each of the n variables in the Boolean formula we create 2 commodities, i and $\bar{i}$, corresponding to truth values true and false. To simplify notation, we say that $\bar{\bar{i}} = i$. Every clause in the formula is assigned a clause gadget, identical to the instance presented in Example 6.14, but with u being a non-terminal for all commodities. The three *relevant commodities* are the commodities corresponding to the *negations* of the three literals appearing in the clause. The preferences of u in such a gadget are chosen so that the edges of the gadget are preferred to edges outside of the gadget. Their order with respect to each other is irrelevant due to the c^i constraints.

All commodities share the same terminals s and t. There is a long path running from s to t, consisting of three segments. The first and the third segment are two copies of the same variable gadget, while the second segment consists of the u vertices of clause gadgets. A variable gadget is defined on vertices $\{a, b_1, b_2, ..., b_n, d\}$, where edges with $c = 1$ and unrestricted commodity capacities run from a to each b_i and from each b_i to d. Edge ab_i ranks commodity i best, $\bar{i}$ second, and the rest in arbitrary order, while b_id ranks commodity $\bar{i}$ best, i second, and the rest in arbitrary order. The vertex preferences of a and d are also arbitrary. These three segments are chained together so that the only edge of s ends at a of the first variable gadget, d of the same gadget is connected to the first u vertex of the second segment, the last u of the same segment is adjacent to a of the second variable gadget and d of this gadget has an edge running to t. On the edges

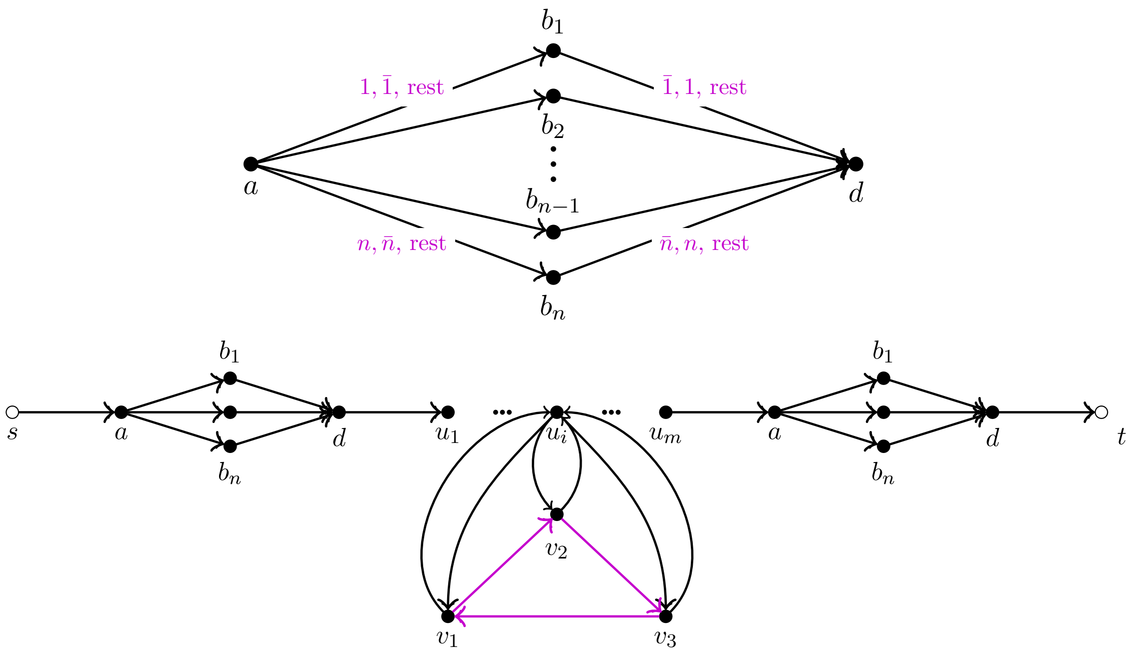


Figure 6.7: A variable gadget and the entire construction for IMSF.

connecting the segments and the u vertices of clause gadgets with each other and with the terminals, $c^i = c = 2n + 1$ for all $1 \le i \le n$, and edge preferences can be chosen arbitrarily.

Lemma 6.18. *If an integral stable multicommodity flow f exists in $\mathcal{I}'$, then there is a truth assignment in $\mathcal{I}$.*

Proof. As defined after Definition 6.11, f^i denotes the total flow value with respect to commodity i.

Claim 38. *For every commodity i, $f^i + f^{\bar{i}} = 1$.*

Proof. If $f^i(ab_i) + f^{\bar{i}}(ab_i) < 1$ for some commodity i and edge ab_i of a variable gadget, then there is an s-t path through b_i, where edges are either unsaturated or they prefer both i and $\bar{i}$ to the commodities they carry. This path blocks f. Since $c(ab_i) = 1$ for every $1 \le i \le n$, $f^i(ab_i) + f^{\bar{i}}(ab_i) = 1$, thus edges ab_i and $b_i d$ of the variable gadgets are saturated with commodities i and $\bar{i}$. This already implies that $f^i + f^{\bar{i}} = 1$ for every $1 \le i \le n$. ■

This claim allows us to assign exactly one truth value to each variable: x_i is true if $f^i = 1$ and it is false if $f^{\bar{i}} = 1$.

Claim 39. *For every clause $C = x_i \vee x_j \vee x_k$, where the variables in C can be in a negated or unnegated form, $f^{\bar{i}} + f^{\bar{j}} + f^{\bar{k}} \le 2$, for every $1 \le i, j, k \le n$.*

Proof. Since u prefers sending flow along its edges in the gadget over forwarding it to the next u vertex on the path, u can be seen as a terminal vertex with respect to the commodities reaching it. As we have shown in Example 6.14, if there is a solution to ISMF, then at most two relevant commodities are present at u. This is why we decided to take the negated version of each literal in the clause: at most two literals are false in each clause. ■ □

Lemma 6.19. *If there is a truth assignment in $\mathcal{I}$, then there is an integral stable multicommodity flow f in $\mathcal{I}'$.*

Proof. The constructed flow to the given truth assignment is the following. For every variable i, $f^i = 1, f^{\bar{i}} = 0$ if i is $\texttt{true}$, and $f^i = 0, f^{\bar{i}} = 1$ otherwise. This rule obviously determines f on all edges not belonging to clause gadgets. Since we started with a valid truth assignment, each clause gadget has at most two out of the three relevant commodities i_1, i_2 and i_3 reaching u. Commodity i_j corresponds to commodity j in Example 6.14. If commodity i_j, $j \in \{1, 2, 3\}$ is not present at u, then we send commodity i_{j+1} (modulo 3) along cycle $C^{i_{j+1}}$ and set all other flow values in the gadget to 0. If two commodities are missing, we send the third along its cycle. If no relevant commodity reaches the gadget, then we leave all edges of the gadget empty.

We need to show now that f is an integral stable flow. Feasibility and integrality clearly follows from the construction. Since the a-d paths in the two variable gadgets are saturated and one of them carries its most preferred commodity to its full cumulative capacity, no blocking walk ρ leaves s or reaches t, in fact, ρ must run between d of the first variable gadget and a of the second variable gadget.

Having eliminated the terminals as starting or end vertices of ρ, we also eliminated the possibility that a commodity i with $f^i = 0$ blocks f. Now we investigate which edge can play the role of the starting edge of a blocking walk ρ.

- Blocking walks cannot start or end with edges outside of clause gadgets, because these edges are the least preferred edges of both of their end vertices.
- Assume now without loss of generality that the first edge of ρ is uv_1 in some clause gadget with relevant commodities i_1, i_2 and i_3, in this order. Then, $f^{i_1} = 1$, but commodity i_1 was not chosen to fill C^1. According to our rules above, the only reason for it is that commodity i_2 is not present at u and commodity i_3 saturates C^3. But the only edge that could be the second edge in ρ is then v_1v_2 in the gadget and it is saturated by its best ranked commodity i_3.
- The last possibility to check is whether ρ can start with an edge of a clause gadget not incident to u. Without loss of generality let us assume this edge is v_1v_2, but it is the only outgoing edge of v_1, thus it cannot fulfill point 2 of Definition 6.12, because each commodity has exactly one outgoing edge at each of vertices v_1, v_2 and v_3. ■

□

6.6 Conclusion and open problems

In this chapter we presented three results:

1. a polynomial version of the Gale-Shapley algorithm for stable flows;
2. a direct algorithm for stable flows with forced and forbidden edges;
3. the NP-completeness of the integral stable multicommodity flow problem.

The most riveting open question regarding Section 6.4 is probably about approximation algorithms. The approximation concepts known from Chapter 2 for MIN BP SR RESTRICTED and SR MIN RESTRICTED VIOLATIONS can be translated to SF RESTRICTED. Even if there is no stable flow containing all forced or avoiding all forbidden edges, how can stability be relaxed such that all edge conditions are fulfilled? Or the other way round: how many edge conditions must be violated by stable flows?

Note that the more complex structure gives rise to alternative interpretations of $bp(M)$ and $|M \cap P| + |Q \setminus M|$. Do intersecting blocking walks count as separate blocking coalitions? If a restricted edge constraint is violated, is it relevant to ask by how much flow?

The big open question of Section 6.5 is clearly algorithms for finding a (possibly fractional) stable multicommodity flow for restricted cases of IMSF. Even though Theorem 6.13 states that it is PPAD-hard to find a solution in the general case, it is natural to ask whether this complexity changes with restricted number of commodities, low degree, and so on. Since the Gale-Shapley algorithm typically executes steps with integer values if the input is integral and we showed the hardness of IMSF, it is likely that a novel approach is needed. Linear programming is a promising direction, but constructing a description of the MSF polytope seems to be an extremely challenging task. At the moment, the most elaborate structure for which a linear program is know is many-to-many stable matchings [40].

Finally, SF RESTRICTED and MSF can be combined with other common notions in stability or flows, such as ties on preference lists, edge weights, unsplittable flows, and so on.

7 Popular matchings

In this last chapter of the thesis we discuss an alternative concept of optimality to stability on matchings under preferences. As in the stable marriage problem, here we are also given a bipartite graph $G = (U \cup W, E)$ where each vertex has a preference list ranking its neighbors. A matching M is *popular* if there is no matching M' such that the vertices that prefer M' to M outnumber those that prefer M to M'. We investigate two problems in this chapter.

The first problem is defined on graphs with strict preferences on both sides. We identify a natural subclass of popular matchings called "dominant matchings" and show that every dominant matching in G can be realized as an image (under a simple and natural mapping) of a stable matching in a modified graph G'. This structural result allows us to find weight-minimal dominant matchings.

In the second setting, every $u \in U$ ranks its neighbors in a strict order of preference, whereas the preference lists of $w \in W$ may contain ties. We show that the problem of deciding whether G admits a popular matching is NP-hard. This is the case even when every $w \in W$ either has a strict preference list or puts all its neighbors into a single tie.

The results presented in Section 7.3 are joint work with Telikepalli Kavitha and have been submitted, while the results presented in Section 7.4 are joint work with Chien-Chung Huang and Telikepalli Kavitha and have been published in [26].

7.1 Introduction

Motivation. We are given a bipartite graph $G = (V, E), V = U \cup W$ and each vertex has a (not necessarily strict) preference list ranking its neighbors. We say that a vertex $v \in V(G)$ *prefers* matching M to matching M' if either v is matched in M and unmatched in M' or v is matched in both and in v's preference list, $M(v)$ is ranked better than $M'(v)$. For matchings M and M' in G, let $\phi(M, M')$ be the number of vertices that prefer M to M'. If $\phi(M', M) > \phi(M, M')$ then we say that M' is *more popular than* M.

Definition 7.1. *A matching M is* popular *if there is no matching that is more popular than M; in other words, $\phi(M, M') \geq \phi(M', M)$ for all matchings M' in G.*

Thus in an election between any pair of matchings, where each vertex casts a vote for the matching that it prefers, a popular matching never loses. It can be therefore regarded as a global, community-optimal solution, while stable matchings capture local optimality.

Another strikingly important feature of popular matchings is that they beat stable matchings in size. The size of a stable matching in G can be as small as $|M_{\max}|/2$, where $M_{\max}$ is a maximum matching in G. Relaxing stability to popularity yields larger matchings and it is easy to show that a largest popular matching has size at least $2|M_{\max}|/3$.

Literature review. The popular matching problem has been studied in the following two models.

- *1-sided model:* here it is only vertices in U that have preferences and cast votes; vertices in W are objects with no preferences or votes.
- *2-sided model:* vertices on both sides have preferences and cast votes.

Popular matchings have been well-studied in the 1-sided model [2, 63, 71, 76, 78, 80]. Abraham et al. [2] gave efficient algorithms to determine if a given instance admits a popular matching or not – their algorithm also works when preference lists of vertices in U admit ties. The notions of *least unpopular* matchings [77] and *popular mixed matchings* [62] were also proposed to deal with instances that had no popular matchings.

In the 2-sided model when all preference lists are strict, it can be shown that any stable matching is popular; thus a popular matching can be found in linear time using the Gale-Shapley algorithm [46]. Moreover, a stable matching is actually a minimum size popular matching and efficient algorithms for computing a maximum size popular matching were given in [51, 60].

When ties are allowed in preference lists on both sides in the 2-sided model, Biró, Irving, and Manlove [12] showed that the popular matching problem is NP-complete. An intermediate variant between the 1-and 2-sided models with strict lists, namely if it is only vertices in U that have preference lists ranking their neighbors, however vertices on both sides cast votes, is studied in [26]. This version of the problem can be solved in polynomial time.

Our contribution and structure. In this chapter we work in the 2-sided model. In Section 7.3 we assume that all vertices rank their neighbors strictly. We define a natural subclass of popular matchings, called “dominant matchings”. A popular matching M is dominant if M is *strictly* more popular ($\phi(M, M') > \phi(M', M)$ for every matching M') than any matching *of larger size* than $|M|$. A characterization of dominant matchings is given in Section 7.3.1. In Section 7.3.2 we show that every dominant matching in G can be realized as an image (under a simple and natural mapping) of a stable matching in a new graph G'. This mapping between stable matchings in G' and dominant matchings in G can also be used to find a minimum weight dominant matching in G efficiently, where we assume that there is a rational weight function on $E(G)$.

In the case with ties we investigate a case between the NP-complete popular matchings with ties problem [12] and the polynomially solvable problem where vertices on one side have a single tie as a preference list [26]. In Section 7.4 we show that the following highly restricted version of the 2-sided popular matching problem with 1-sided ties is NP-complete:

- every $u \in U$ has a strict preference list of length 2 or 4;
- every $w \in W$ has either a strict preference list of length 2 or a single tie of length 2 or 3 as a preference list.

7.2 Preliminaries

Let M be any matching in $G = (V, E)$. We introduce a function to simplify notation used in all our proofs later.

Definition 7.2. *For any $v \in V(G)$ and neighbors x and y of v, define v's* vote *between x and y as:*

$$\mathsf{vote}_v(x, y) = \begin{cases} +1 & \text{if } x >_v y \\ -1 & \text{if } x <_v y \\ 0 & \text{otherwise (i.e., } x = y \text{ or } \mathrm{rank}_v(x) = \mathrm{rank}_v(y)\text{).} \end{cases}$$

If a vertex v is unmatched, then $M(v) = \emptyset$ and we define $\mathsf{vote}_v(v', M(v))$ to be $+1$ for all neighbors v' of v since every vertex prefers to be matched than to be unmatched. Label each edge $e = uw$ in $E(G) \setminus M$ by the pair (α_e, β_e), where $\alpha_e = \mathsf{vote}_u(w, M(u))$ and $\beta_e = \mathsf{vote}_w(u, M(w))$, i.e., α_e is u's vote for w vs. $M(u)$ and β_e is w's vote for u vs. $M(w)$. Note that if an edge uw is labeled $(+1, +1)$, then uw blocks M in the stable matching sense. If an edge uw is labeled $(-1, -1)$, then both u and w prefer their respective partners in M to each other. Let G_M be the subgraph of G obtained by deleting edges that are labeled $(-1, -1)$.

7.2.1 Strict preferences

The notation introduced in this section will be used in Section 7.3. Let us assume that every vertex in $V(G)$ ranks its incident edges in a strict order of preference. We now identify a natural subclass of popular matchings called *dominant* popular matchings or dominant matchings, in short. In order to define dominant matchings, we first define the relation "defeats" as follows.

Definition 7.3. *Matching M* defeats *matching M' if either of these two conditions holds:*

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

(ii) $\phi(M, M') = \phi(M', M)$ and $|M| > |M'|$.

When M and M' gather the same number of votes in the election between M and M', instead of declaring these matchings as incomparable (as done under the "more popular than" relation), it seems natural to regard the larger of M, M' as the *winner* of the election. Condition (ii) of the *defeats* relation captures this notion. We define dominant matchings to be those popular matchings that are not defeated by any matching (as per Definition 7.3).

Definition 7.4. *Matching M is* dominant *if there is no matching that* defeats *it; in other words, M is popular and for any matching M', if $|M'| > |M|$, then M is more popular than M'.*

$u_1 : w_1 \quad w_2 \quad w_3$	$w_1 : u_1 \quad u_2 \quad u_3$	
$u_2 : w_1 \quad w_2$	$w_2 : u_1 \quad u_2$	
$u_3 : w_1$	$w_3 : u_1$	

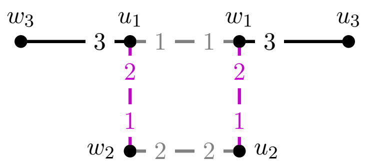


Figure 7.1: There are two maximum size popular matchings here: $M_1 = \{u_1w_1, u_2w_2\}$ and $M_2 = \{u_1w_2, u_2w_1\}$. The matching M_1 is not dominant since the larger matching $M_3 = \{u_1w_3, u_2w_2, u_3w_1\}$ defeats it. The matching M_2 is dominant since M_2 is more popular than M_3.

Note that a dominant matching has to be a maximum size popular matching since smaller-sized popular matchings get defeated by a popular matching of maximum size. However not every maximum size popular matching is a dominant matching, as the example (from [51]) in Figure 7.1 demonstrates.

Analogous to Definition 7.4, we can define another interesting subclass of popular matchings: those popular matchings M such that *for any matching M', if $|M'| < |M|$ then M is more popular than M'.* It is easy to show that this class of matchings is exactly the set of stable matchings. That is, we can show that any popular matching M that is more popular than every *smaller-sized* matching is a stable matching and conversely, every stable matching is more popular than any smaller-sized matching. The first direction follows from the fact that if M is blocked by the edge uw, then for $M' = M \cup uw \setminus uM(u) \setminus wM(w)$, that is, the matching derived from M by adding the blocking uw, it is the case that $\phi(M, M') = \phi(M', M) = 2$, even though $|M'| < |M|$. The other direction is trivial, since every stable matching is a minimum size popular matching. We can say that dominant matchings are to the class of maximum size popular matchings what stable matchings are to the class of minimum size popular matchings: these are popular matchings that carry the proof of their maximality (minimality) by being more popular than every matching of larger (smaller) size.

7.2.2 Ties in preferences

In the second half of this chapter, in Section 7.4, ties are allowed in preference lists. The general problem we investigate is popular matchings with ties.

Problem 21. PMT
Input: $\mathcal{I} = (G, O)$; a bipartite graph $G = (V, E)$ and preference lists O on both sides, possibly with ties.
Question: Is there a popular matching M?

Here we discuss a variant of PMT where vertices in U have strict preferences while vertices in W are allowed to have ties in their preference lists. Thus each man ranks all women that he finds acceptable in a strict order of preference, while each woman need not come up with a total order on all acceptable men. This model captures characteristics of students applying to universities or applicants to posts, as in NRMP [108]. While it is reasonable to assume that applicants submit a strictly ordered list of posts, they may get grouped together in terms of their suitability, thus equally competent applicants are tied together at the same rank.

Observe that popular matchings need not always exist in such instances. Consider an instance where $U = \{u_1, u_2, u_3\}$ and $W = \{w_1, w_2, w_3\}$ and for $i \in \{1, 2, 3\}$, each u_i has the same preference list: $\langle w_1, w_2, w_3 \rangle$, while each w_i ranks u_1, u_2, u_3 the same, i.e., u_1, u_2, u_3 are tied together in w_i's preference list. It is easy to see that for any matching M here, there is another matching M' such that M is more popular than M, thus this instance admits no popular matching.

7.3 Dominant popular matchings

Throughout this section, we assume that every vertex in $G = (V, E)$ has a strictly ordered preference list. In Theorem 7.5 we present a known characterization of popular matchings in such instances, and then in Corollary 7.7 we extend this to dominant matchings.

Besides forming a natural subclass of popular matchings, dominant matchings can also be used to solve the following two problems [27].

- The popular edge problem, an analogous variant of the stable edge problem, asks whether a popular matching exists in G that contains a fixed edge e. One can show that if the answer is yes, then there exists either a stable matching in G that contains e or a dominant matching in G that contains e. These are easy to compute, e.g., using the weighted versions of the problems.
- When all popular matchings in G have the same size, it could be the case that every popular matching in G is also stable. One can use dominant matchings to efficiently check if this is the case or not. If there exists an unstable popular matching in G, then there has to exist an unstable dominant matching in G. This latter is easy to check by iterating through all edges, assuming they block a dominant matching.

7.3.1 A characterization of dominant matchings

Recall from Chapter 4 that an alternating path with respect to a matching M is a sequence of incident edges that are in M and outside of it in an alternating manner. An alternating path is called augmenting if its first and last edges are both outside of M. As defined earlier, G_M is the subgraph of G obtained by deleting edges that are labeled $(-1, -1)$. The following theorem characterizes popular matchings.

Theorem 7.5 (from [51]). *A matching M is popular if and only if the following three conditions are satisfied in the subgraph G_M:*

(i) There is no alternating cycle with respect to M that contains a $(+1, +1)$ edge.

(ii) There is no alternating path starting from an unmatched vertex with respect to M that contains a $(+1, +1)$ edge.

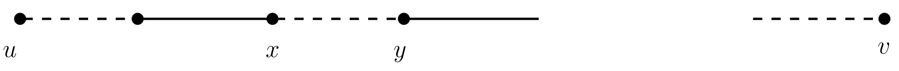


Figure 7.2: The u-v augmenting path ρ in G where the solid edges are in M; at least one edge here (say, xy) is labeled $(-1,-1)$.

(iii) There is no alternating path with respect to M that contains two or more $(+1,+1)$ *edges.*

Lemma 7.6 characterizes those popular matchings that are dominant. The "if" side of Lemma 7.6 was shown in [60]: it was shown that if there is no augmenting path with respect to a popular matching M in G_M then M is more popular than all larger matchings, thus M is a maximum size popular matching.

Here we show that the converse holds as well, i.e., if M is a popular matching such that M is more popular than all larger matchings, in other words, if M is a dominant matching, then there is no augmenting path with respect to M in G_M.

Lemma 7.6. *A popular matching M is dominant if and only if there is no augmenting path with respect to M in G_M.*

Proof. Let M be a popular matching in G. Suppose there is an augmenting path ρ with respect to M in G_M. Let us use $M \oplus \rho$ for the matching derived from M by augmenting along ρ, and $M \approx M'$ to denote both matchings getting the same number of votes in an election between them, i.e., $\phi(M, M') = \phi(M', M)$. We will now show that $M \oplus \rho \approx M$. Since $M \oplus \rho$ is a larger matching than M, if $M \oplus \rho \approx M$, then it means that $M \oplus \rho$ defeats M, thus M is not dominant.

Consider $M \oplus \rho$ versus M: every vertex that does not belong to the path ρ gets the same partner in both these matchings. Hence vertices outside ρ are indifferent between these two matchings. Consider the vertices on ρ. In the first place, there is no edge in $\rho \setminus M$ that is labeled $(+1,+1)$, otherwise that would contradict condition (ii) of Theorem 7.5. Since the path ρ belongs to G_M, no edge is labeled $(-1,-1)$ either. Hence every edge in $\rho \setminus M$ is labeled either $(+1,-1)$ or $(-1,+1)$. Note that the $+1$ signs count the number of votes for $M \oplus \rho$ while the -1 signs count the number of votes for M. Thus the number of votes for $M \oplus \rho$ equals the number of votes for M on vertices of ρ, and thus in the entire graph G. Hence $M \oplus \rho \approx M$.

Now we show the other direction: if there is no augmenting path with respect to a popular matching M in G_M then M is dominant. Let M' be a larger matching. Consider $M \oplus M'$ in G: this is a collection of alternating paths and alternating cycles and since $|M'| > |M|$, there is at least one augmenting path with respect to M here. Call this path ρ, running from vertex u to vertex v, without any restriction on which side of the graph they belong to. Let us count the number of votes for M versus M' among the vertices of ρ.

No edge in ρ is labeled $(+1,+1)$ as that would contradict condition (ii) of Theorem 7.5, thus all the edges of M' in ρ are labeled $(-1,+1)$, $(+1,-1)$, or $(-1,-1)$. Since ρ does not exist in G_M, there is at least one edge that is labeled $(-1,-1)$ here (see Figure 7.2).

Thus among the vertices of ρ, M gets more votes than M' (recall that $+1$'s are votes for M' and -1's are votes for M). Thus M is more popular than M' among the vertices of ρ.

By the popularity of M, we know that M gets at least as many votes as M' over all other paths and cycles in $M \oplus M'$; this is because if ρ is an alternating path/cycle in $M \oplus M'$ such that the number of vertices on ρ that prefer M' to M is more than the number that prefer M to M', then $M \oplus \rho$ is more popular than M, a contradiction to the popularity of M. Thus adding up over all the vertices in G, it follows that $\phi(M, M') > \phi(M', M)$. Hence M is more popular than any larger matching and so M is a dominant matching. □

Now we are ready to present a characterization of dominant matchings, following immediately from Theorem 7.5 and Lemma 7.6.

Corollary 7.7. *Matching M is a dominant matching if and only if M satisfies conditions (i)-(iii) of Theorem 7.5 and condition (iv): there is no augmenting path with respect to M in G_M.*

7.3.2 The set of dominant matchings

In this section we show a surjective mapping from the set of stable matchings in a new instance $G' = (U' \cup W', E')$ to the set of dominant matchings in $G = (U \cup W, E)$. The construction of $G' = (U' \cup W', E')$ is as follows.

Corresponding to every man $u \in U$, there will be two men u_0 and u_1 in U' and one woman d_u in W'. The vertex d_u will be referred to as the dummy woman corresponding to u. Corresponding to every woman $w \in W$, there will be exactly one woman in W' – for the sake of simplicity, we will use w to refer to this woman as well. Thus $W' = W \cup D$, where $D = \{d_u : u \in U\}$ is the set of dummy women.

Regarding the other side of the graph, $U' = U_0 \cup U_1$, where $U_i = \{u_i : u \in U\}$ for $i \in \{0, 1\}$, and vertices in U_0 are called *level 0 vertices*, while vertices in U_1 are called *level 1 vertices*.

We now describe the edge set of G' and the preferences of the vertices. For each $u \in U$, the vertex d_u has exactly two neighbors: these are u_0 and u_1 and d_u's preference order is u_0 followed by u_1. The dummy woman d_u is u_1's most preferred neighbor and u_0's least preferred neighbor. The preference list of u_0 is all the neighbors of u (in u's preference order) followed by d_u. On the other hand, the preference list of u_1 is d_u followed by the neighbors of u (in u's preference order) in G.

For any $w \in W$, its preference list in G' is level 1 neighbors in the same order of preference as in G followed by level 0 neighbors in the same order of preference as in G. For instance, if w's preference list in G is u followed by u', then w's preference list in G' is top-choice u_1, then u'_1, and then u_0, and the last-choice is u'_0. We show an example in Figure 7.3.

We now define the mapping T : {stable matchings in G'} $\to$ {dominant matchings in G}. Let M' be any stable matching in G.

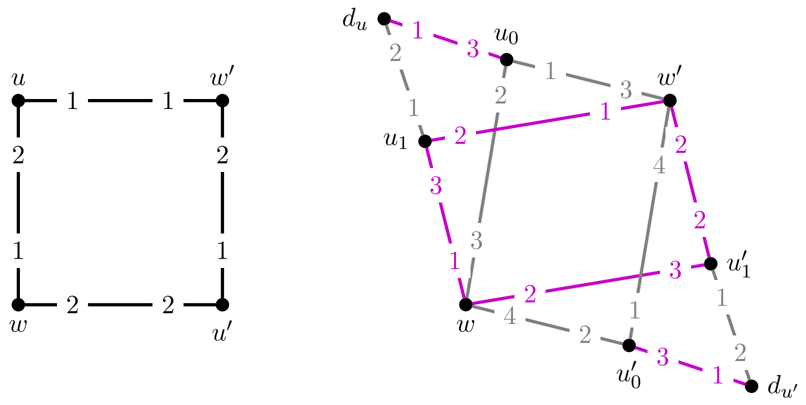


Figure 7.3: The graph G' on the right corresponding to G on the left. We used purple to color edges in $(U_1 \times W) \cup (U_0 \times d_u)$ and gray to color edges in $(U_0 \times W) \cup (U_1 \times d_u)$.

- $T(M')$ is the set of edges obtained by deleting all edges involving vertices in D (i.e., dummy women) from M' and replacing every edge $u_i w \in M'$, where $w \in W$ and $i \in \{0, 1\}$, by the edge uw.

It is easy to see that $M = T(M')$ is a matching in G. This is because M' has to match d_u, for every $u \in U$, since d_u is the top-choice for u_1. Thus for each $u \in U$, one of u_0, u_1 has to be matched to d_u. Hence at most one of u_0, u_1 is matched to a non-dummy woman w and thus $M = T(M')$ is a matching in G. First we show that $M = T(M')$ is a dominant matching in G if M' was a stable matching in G'. Then, we will prove that T is surjective: corresponding to any dominant matching M in G, there is a stable matching M' in G' such that $T(M') = M$.

M is a dominant matching in G.

Lemma 7.8. *If M' is a stable matching in G', then $M = T(M')$ is a dominant matching in G.*

Proof. This proof is similar to the proof of correctness of the maximum size popular matching algorithm in [60]. As described in Section 7.3.1, in the graph G, label each edge $e = uw$ in $E(G) \setminus M$ by the pair (α_e, β_e), where $\alpha_e \in \{+1, -1\}$ is u's vote for w vs. $M(u)$ and $\beta_e \in \{+1, -1\}$ is w's vote for u vs. $M(w)$.

- It will be useful to assign a value in $\{0, 1\}$ to each $u \in U$. If $M'(u_1) = d_u$, then $f(u) = 0$ else $f(u) = 1$. In particular, if $u \in U$ is unmatched in M then $u_0 d_u \in M'$ and so $f(u) = 1$.
- We will now define f-values for vertices in W as well. If $M'(w) \in U_1$ then $f(w) = 1$, else $f(w) = 0$. So if $w \in W$ is unmatched in M' (and thus in M) then $f(w) = 0$.

Claim 40. *The following statements hold on the edge and vertex labels for every $u, y \in U$ and $w, z \in W$:*

(1) If the edge uw is labeled $(+1, +1)$, then $f(u) = 0$ and $f(w) = 1$.

(2) If yz is an edge such that $f(y) = 1$ and $f(z) = 0$, then yz has to be labeled $(-1, -1)$.

Proof. We show part (1) first. The edge uw is labeled $(+1, +1)$. Let $M(u) = z$ and $M(w) = y$. Thus $w >_u z$ and $u >_w y$. We know from the definition of our function T that $M'(z) \in \{u_0, u_1\}$ and $M'(w) \in \{y_0, y_1\}$. So there are 4 possibilities: M' contains (1) u_0z and y_0w, (2) u_1z and y_0w, (3) u_1z and y_1w, (4) u_0z and y_1w.

We know that M' has no blocking edge in G' since it is a stable matching. In (1), the edge u_0w blocks M', and in (2) and (3), the edge u_1w blocks M'. Thus the only possibility is (4). That is, $M'(w) \in U_1$ and $M'(u_1) = d_u$. In other words, $f(u) = 0$ and $f(w) = 1$.

We now show part (2) of Claim 40. We are given that $f(y) = 1$, so $M'(y_0) = d_y$. We know that d_y is y_0's last choice and y_0 is adjacent to z, thus $M'(z) >_z y_0$. Since we are given that $f(z) = 0$, i.e., $M'(z) \in U_0$, it follows that $M'(z) = u_0$, where $u >_z y$ in G.

In G', $y_1 >_z u_0$ since z prefers any level 1 neighbor to a level 0 neighbor. Thus y_1 is matched to a neighbor that is ranked better than z in y's preference list, i.e., $M'(y_1) = v$, where $v >_y z$. We have the edges yv and uz in M, thus both y and z prefer their respective partners in M to each other. Hence the edge yz has to be labeled $(-1, -1)$. ■

Claims 41 and 42 shown below, along with Lemma 7.6, imply that M is a dominant matching in G.

Claim 41. *There is no augmenting path with respect to M in G_M.*

Proof. Let $u \in U$ and $w \in W$ be unmatched in M. Then $f(u) = 1$ and $f(w) = 0$. If there is an augmenting path $\rho = \langle a, \cdots, b \rangle$ with respect to M in G_M, then in ρ we move from a man whose f-value is 1 to a woman whose f-value is 0. Thus there have to be two consecutive vertices $y \in U$ and $z \in W$ on ρ such that $f(y) = 1$ and $f(z) = 0$. However part (2) of Claim 40 tells us that such an edge yz has to be labeled $(-1, -1)$. In other words, G_M does not contain the edge yz or equivalently, there is no augmenting path ρ in G_M. ■

Claim 42. *M is a popular matching in G.*

Proof. We will show that M satisfies conditions (i)-(iii) of Theorem 7.5.

Condition (i). Consider any alternating cycle C with respect to M in G_M and let $u \in U$ be any vertex in C: if $f(u) = 0$ then its partner $w = M(u)$ also satisfies $f(w) = 0$ and part (2) of Claim 40 tells us that there is no edge in G_M between w and any u' such that $f(u') = 1$. Similarly, if $f(u) = 1$ then its partner $w = M(u)$ also satisfies $f(w) = 1$ and though there can be an edge yw labeled $(+1, +1)$ incident on w, part (1) of Claim 40 tells us that $f(y) = 0$ and thus there is no way the cycle C can return to u, whose f-value is 1. Hence if G_M contains an alternating cycle C with respect to M, then all vertices in C have the same f-value. Since there can be no edge labeled $(+1,+1)$ between two vertices whose f-value is the same (by part (1) of Claim 40), it follows that C has no edge labeled $(+1,+1)$.

Condition (ii). Consider any alternating path ρ with respect to M in G_M and let the starting vertex in ρ be $u \in U$. Since u is unmatched in M, we have $f(u) = 1$ and we know from part (2) of Claim 40 that there is no edge in G_M between such a man and a woman whose f-value is 0. Thus u's neighbor in ρ is a woman w' such that $f(w') = 1$. Since $f(w') = 1$, its partner $u' = M(w')$ also satisfies $f(u') = 1$ and part (2) of Claim 40 tells us that there is no edge in G_M between u' and any w'' such that $f(w'') = 0$, thus all vertices of ρ have f- value 1 and thus there is no edge labeled $(+1,+1)$ in ρ.

Suppose the starting vertex in ρ is $w \in W$. Since w is unmatched in M, we have $f(w) = 0$ and we again know from part (2) of Claim 40 that there is no edge in G_M between such a woman and a man whose f-value is 1. Thus w's neighbor in ρ is a woman u' such that $f(u') = 0$. Since $f(u') = 0$, its partner $w' = M(u')$ also satisfies $f(w') = 0$ and part (2) of Claim 40 tells us that there is no edge in G_M between w' and any u'' such that $f(u'') = 1$, thus all vertices of ρ have f- value 0 and thus there is no edge labeled $(+1,+1)$ in ρ.

Condition (iii). Consider any alternating path ρ with respect to M in G_M. We can assume that the starting vertex in ρ is matched in M (as condition (ii) has dealt with the case when this vertex is unmatched). Suppose the starting vertex is $u \in U$. If $f(u) = 0$ then its partner $w = M(u)$ also satisfies $f(w) = 0$ and part (2) of Claim 40 tells us that there is no edge in G_M between w and any u' such that $f(u') = 1$, thus all vertices of ρ have f- value 0 and thus there is no edge labeled $(+1,+1)$ in ρ. If $f(u) = 1$ then after traversing some vertices whose f-value is 1, we can encounter an edge yz that is labeled $(+1,+1)$ where $f(z) = 1$ and $f(y) = 0$. However once we reach y, we get stuck in vertices whose f-value is 0 and thus we can see no more edges labeled $(+1,+1)$.

Suppose the starting vertex in ρ is $w \in W$. If $f(w) = 1$ then its partner $u = M(w)$ also satisfies $f(u) = 1$ and part (2) of Claim 40 tells us that there is no edge in G_M between u and any w' such that $f(w') = 0$, thus all vertices of ρ have f-value 1 and thus there is no edge labeled $(+1,+1)$ in ρ. If $f(w) = 0$ then after traversing some vertices whose f-value is 0, we can encounter an edge yz labeled $(+1,+1)$ where $f(y) = 0$ and $f(z) = 1$. However once we reach z, we get stuck in vertices whose f-value is 1 and thus we can see no more edges labeled $(+1,+1)$. Thus in all cases there is at most one edge labeled $(+1,+1)$ in ρ. ■ □

T is surjective

Lemma 7.9. *Corresponding to any dominant matching M in G, there is a stable matching M' in G' such that $T(M') = M$.*

Proof. We will work in G_M, the subgraph of G obtained by deleting all edges labeled $(-1,-1)$. We now construct sets $U_0, U_1 \subseteq U$ and $W_0, W_1 \subseteq W$ as described in Algorithm 4. These sets will be useful in constructing the matching M'.

Algorithm 4 Construction of $U_0, U_1 \subseteq U$ and $W_0, W_1 \subseteq W$

```
1: initialize $U_0 := W_1 := \emptyset$
2:            $U_1 :=$ {unmatched men in $M$}
3:            $W_0 :=$ {unmatched women in $M$}
4: for every edge $yz \in M$ that is labeled $(+1, +1)$ do
5:     $U_0 := U_0 \cup y$
6:     $W_0 := W_0 \cup M(y)$
7:     $W_1 := W_1 \cup z$
8:     $U_1 := U_1 \cup M(z)$
9: end for
10: while $\exists$ a matched man $u \notin U_0$ that is adjacent in $G_M$ to a woman in $W_0$ do
11:     $U_0 := U_0 \cup u$
12:     $W_0 := W_0 \cup M(u)$
13: end while
14: while $\exists$ a matched woman $w \notin W_1$ that is adjacent in $G_M$ to a man in $U_1$ do
15:     $W_1 := W_1 \cup w$
16:     $U_1 := U_1 \cup M(w)$
17: end while
```

All unmatched men are in U_1 and all unmatched women are in W_0. For every edge $yz, y \in U, z \in W$ that is labeled $(+1, +1)$, we add y and its partner to U_0 and W_0, respectively while z and its partner are added to W_1 and U_1, respectively. For any man u, if u is adjacent to a vertex in W_0 and u is not in U_0, then u and its partner get added to U_0 and W_0, respectively. Similarly, for any woman w, if w is adjacent to a vertex in U_1 and w is not in W_1, then w and its partner get added to W_1 and U_1, respectively.

The following observations are easy to see (refer to Figure 7.4). Every $u \in U_1$ has an even length alternating path in G_M to either

(1) a man unmatched in M (by lines 1-3 and lines 14-17 in Algorithm 4) or

(2) a man $M(z)$ where z has an edge labeled $(+1, +1)$ incident on it (by lines 4-9 and lines 14-17).

Similarly, every $u \in U_0$ has an odd length alternating path in G_M to either

(3) a woman unmatched in M (by lines 1-3 and lines 10-13) or

(4) a woman $M(y)$ where y has an edge labeled $(+1, +1)$ incident on it (by lines 4-9 and lines 10-13).

The proof of the following claim is based on the characterization of dominant matchings in terms of conditions (i)-(iv) as given by Corollary 7.7. We will also use (1)-(4) observed above.

Claim 43. *$U_0 \cap U_1 = \emptyset$.*

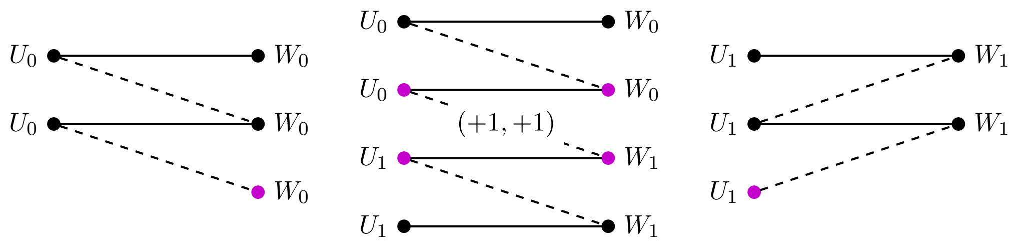


Figure 7.4: Vertices get added to U_1 and U_0 by alternating paths in G_M from either unmatched vertices (first and third paths) or endpoints of edges labeled $(+1,+1)$ (middle path). The solid edges are in M and the colored vertices are the ones that are added to their respective sets at the beginning, in lines 1-9 of Algorithm 4.

Proof. Case 1. Suppose u satisfies reasons (1) and (3) for its inclusion in U_1 and in U_0, respectively. So u is in U_1 because it is reachable via an even alternating path in G_M from an unmatched man u; also u is in U_0 because it is reachable via an odd length alternating path in G_M from an unmatched woman w. Then there is an augmenting path $\langle u, \dots, w\rangle$ with respect to M in G_M – a contradiction to the fact that M is dominant (by Lemma 7.6).

Case 2. Suppose u satisfies reasons (1) and (4) for its inclusion in U_1 and in U_0, respectively. So u is in U_1 because it is reachable via an even alternating path with respect to M in G_M from an unmatched man u; also u is in U_0 because it is reachable via an odd length alternating path in G_M from z, where edge yz is labeled $(+1,+1)$. Then there is an alternating path with respect to M in G_M from an unmatched man u to the edge yz labeled $(+1,+1)$ and this is a contradiction to condition (ii) of popularity.

Case 3. Suppose u satisfies reasons (2) and (3) for its inclusion in U_1 and in U_0, respectively. This case is absolutely similar to Case 2. This will cause an alternating path with respect to M in G_M from an unmatched woman to an edge labeled $(+1,+1)$, a contradiction again to condition (ii) of popularity.

Case 4. Suppose u satisfies reasons (2) and (4) for its inclusion in U_1 and in U_0, respectively. So u is reachable via an even length alternating path in G_M from an edge labeled $(+1,+1)$ and $M(u)$ is also reachable via an even length alternating path in G_M from an edge labeled $(+1,+1)$. If it is the same edge labeled $(+1,+1)$ that both u and $M(u)$ are reachable from, then there is an alternating cycle in G_M with a $(+1,+1)$ edge – a contradiction to condition (i) of popularity. If these are two different edges labeled $(+1,+1)$, then we have an alternating path in G_M with two edges labeled $(+1,+1)$ – a contradiction to condition (iii) of popularity.

These four cases finish the proof that $U_0 \cap U_1 = \emptyset$. ■

We now describe the construction of the matching M'. Initially $M' = \emptyset$.

- For each $u \in U_0$: add the edges $u_0 M(u)$ and $u_1 d_u$ to M'.
- For each $u \in U_1$: add the edge $u_0 d_u$ to M' and if u is matched in M then add

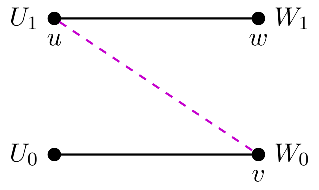


Figure 7.5: If $v >_{u_1} w$ in G', then $v >_u w$ in G; thus the edge uv has to be present in G_M.

$u_1M(u)$ to M'.

- For $u \notin (U_0 \cup U_1)$: add the edges $u_0M(u)$ and u_1d_u to M'.

Note that the men outside $U_0 \cup U_1$ are not reachable from either unmatched vertices or edges labeled $(+1,+1)$ via alternating paths in G_M. Now we show that the constructed matching M' is indeed stable.

Claim 44. *M' is a stable matching in G'.*

Proof. Suppose M' is not stable in G'. Then there are edges u_iv and u_jw in M' where $i, j \in \{0,1\}$, such that in the graph G', the vertices v and u_j prefer each other to u_i and w, respectively. There cannot be a blocking edge involving a dummy woman, thus the edges uv and uw are in M.

If $i = j$, then the edge uv blocks M in G. However, from the construction of the sets U_0, U_1, W_0, W_1, we know that all the blocking edges with respect to M are in $U_0 \times W_1$. Thus there is no blocking edge in $U_0 \times W_0$ or in $U_1 \times W_1$ with respect to M and so $i \neq j$. Since $u_j >_v u_i$ in G', the only possibility is $i = 0$ and $j = 1$. It has to be the case that $v >_u w$, so there is an edge labeled $(+1,-1)$ between $u \in U_1$ and $v \in W_0$ (see Figure 7.5).

So once v got added to W_0, since u is adjacent in G_M to a vertex in W_0, vertex u satisfied the condition in line 10 of our algorithm to construct the sets U_0, U_1, W_0, and W_1. So u would have got added to U_0 as well, i.e., $u \in U_0 \cap U_1$, a contradiction to Claim 43. Thus there is no blocking edge with respect to M' in G'. ■

For each $u \in U$, note that exactly one of u_0d_u, u_1d_u is in M'. In order to form the set $T(M')$, the edges of M' with women in D are pruned and each edge $u_iw \in M'$, where $w \in W$ and $i \in \{0,1\}$, is replaced by uw. It is easy to see that $T(M') = M$.

This conclude the proof that every dominant matching in G can be realized as an image under T of some stable matching in G'. Thus T is surjective. □

Our mapping T can also be used to solve the minimum weight dominant matching problem in polynomial time. Here we are given a weight function $\omega : E(G) \to \mathbb{Q}$ and the problem is to find a dominant matching in G whose sum of edge weight is the most. We will use the mapping T established from {stable matchings in G'} to {dominant matchings in G} to solve the minimum weight dominant matching problem in G. It is easy to extend ω to the edge set of G'. For each edge uw in G, we will assign

$\omega(u_0w) = \omega(u_1w) = \omega(uw)$ and we will set $\omega(u_0d_u) = \omega(u_1d_u) = 0$. Thus the weight of any stable matching M' in G' is the same is the weight of the dominant matching $T(M')$ in G.

Since every dominant matching M in G equals $T(M')$ for some stable matching M' in G', it follows that the minimum weight dominant matching problem in G is the same as the minimum weight stable matching problem in G'. Since a minimum weight stable matching in G' can be computed in polynomial time [38, 39, 54, 93], we can conclude Theorem 7.10.

Theorem 7.10. *Given a graph $G = (V, E)$ with strict preference lists and a weight function $\omega : E(G) \to \mathbb{Q}$, the problem of computing a minimum weight dominant matching can be solved in polynomial time.*

7.4 Popular matching with 1-sided ties

In this section, we study instances of the popular matching problem where ties can occur in preference lists. It is known [12] that when ties are allowed on both sides, the popular matching problem is NP-complete. An intermediate variant between the 1- and 2-sided models with strict lists, namely if it is only vertices in U that have strict preference lists ranking their neighbors, however vertices on both sides cast votes, is studied in [26]. This version of the problem can be solved in polynomial time. Here we build a bridge between these two results with the following theorem.

Theorem 7.11. *Let $G = (V, E)$ be a bipartite graph where each $u \in U$ has a strict preference list while each $w \in W$ either has a strict preference list or puts all its neighbors into a single tie.* PMT *in G is* NP*-complete.*

Given a matching M in $G = (V, E)$, it was shown in [12] that M can be tested for popularity in $\mathcal{O}(\sqrt{|V|} \cdot |E|)$ time, even in the presence of ties. Thus it is known that the problem is in NP, we only need to show NP-hardness.

Our reduction from the (2,2)-E3-SAT problem (Problem 7 in Chapter 2) shows that the following version of PMT in $G = (V, E)$ with 1-sided ties is NP-complete:

- every vertex in U has a strict preference list of length 2 or 4;
- every vertex in W has either a strict preference list of length 2 or a single tie of length 2 or 3 as a preference list.

Recall from Chapter 2 that the (2,2)-E3-SAT problem takes as its input a Boolean formula $\mathcal{I}$ in CNF, where each clause contains three literals and every variable appears exactly twice in unnegated form and exactly twice in negated form in the clauses. The problem is to determine if $\mathcal{I}$ is satisfiable or not and this problem is NP-complete [10].

Constructing a popular matching instance $G = (V, E)$ from $\mathcal{I}$. Let $\mathcal{I}$ have m clauses and n variables. The instance G constructed consists of n variable gadgets, m clause gadgets, and some interconnecting edges between these, as also shown in Figure 7.6.

A *variable gadget* representing variable v_j, for $1 \leq j \leq n$, is a 4-cycle on vertices $u_{j_1}, w_{j_1}, u_{j_2}$, and w_{j_2}, where $u_{j_1}, u_{j_2} \in U$ and $w_{j_1}, w_{j_2} \in W$. A *clause gadget* representing clause C_i, for $1 \leq i \leq m$, is a subdivision graph of a claw. Its edges are divided into three classes: $c_i \in W$ is at the center, the neighbors of c_i are $x_{i_1}, x_{i_2}, x_{i_3} \in U$, and finally, each of $x_{i_1}, x_{i_2}, x_{i_3}$ is adjacent to its respective copy in $Y_i = \{y_{i_1}, y_{i_2}, y_{i_3}\}$, where $Y_i \subseteq W$.

A vertex in Y_i represents an appearance of a variable. For instance, y_{3_1} is the first literal of the third clause. Each of the vertices in Y_i is connected to a vertex in the variable gadget via an *interconnecting edge.* Vertex y_{i_k} is connected to the gadget standing for variable j if the k-th literal of the i-th clause is either v_j or $\neg v_j$. If it is v_j, then the interconnecting edge ends at u_{j_1}, else at u_{j_2}. The preferences of this instance can be seen in Figure 7.6. The constructed graph trivially satisfies both conditions claimed above, i.e., every vertex in U has a strict preference list of length 2 or 4 and every vertex in W has either a strict preference list of length 2 or a single tie of length 2 or 3 as a preference list.

Lemma 7.12. *If there is a popular matching M in G, then one can construct a truth assignment in $\mathcal{I}$.*

Proof. The graph G is as described above. Claim 45 states that any popular matching M in G has a certain structure.

Claim 45. *Any popular matching M in G has to obey the following properties.*

- *M avoids all interconnecting edges.*
- *M is one of the two perfect matchings on any variable gadget; i.e., for each j, the edges of M restricted to the gadget corresponding to variable v_j are either (i) $u_{j_1}w_{j_1}$ and $u_{j_2}w_{j_2}$, or (ii) $u_{j_1}w_{j_2}$ and $u_{j_2}w_{j_1}$.*
- *M leaves exactly one vertex per clause i unmatched and this unmatched vertex y_{i_k} is adjacent to a u_{j_t} that is matched to w_{j_1}.*

We prove Claim 45 below. For now, we assume this claim and show how we construct a satisfying truth assignment for $\mathcal{I}$. We assign true to all variables v_j such that $M \supseteq \{u_{j_1}w_{j_1}, u_{j_2}w_{j_2}\}$ and false to all variables v_j such that $M \supseteq \{u_{j_1}w_{j_2}, u_{j_2}w_{j_1}\}$.

The truth value of every variable is now uniquely defined and all we need to show is that every clause has a true literal. Assume that in clause C_i, all three literals are false. The clause gadget has an unmatched vertex y_{i_k} that is adjacent to a u_{j_t}. If the literal is false, then u_{j_t} prefers y_{i_k} to $M(u_{j_t}) = w_{j_2}$ and $u_{j_t}y_{i_k}$ becomes an edge labeled $(+1, +1)$ with an unmatched end vertex – this contradicts the popularity of M. Hence in every clause, there is at least one true literal and so this is a satisfying assignment.

Proof of Claim 45. Our first observation is that every c_i, for $1 \leq i \leq m$, and every w_{j_1}, for $1 \leq j \leq n$, must be matched in M. That is because these vertices are the top choices for each of their neighbors, hence if one of them is left unmatched, then there would be an edge labeled $(+1, +1)$ incident to an unmatched vertex. This contradicts the popularity of M.

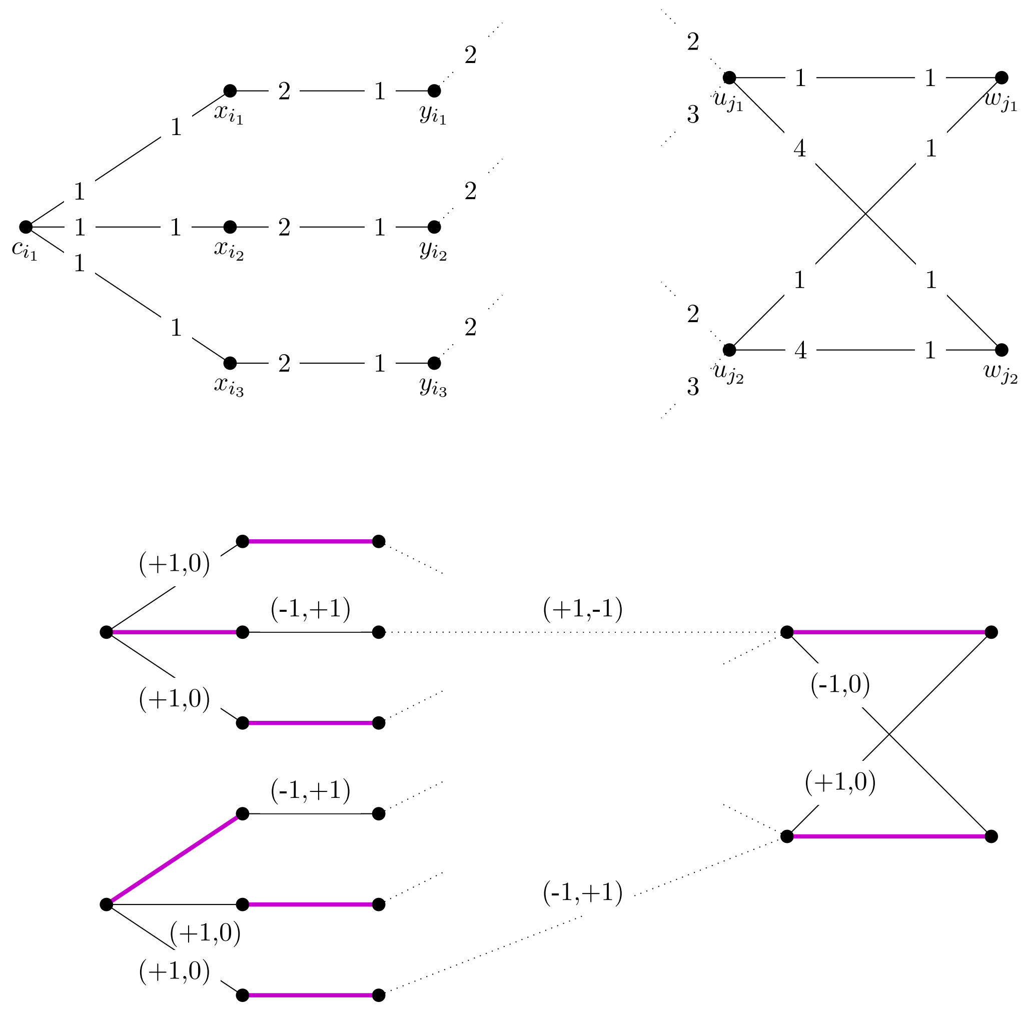


Figure 7.6: A clause gadget, a variable gadget, and the structure of the entire construction with a variable that appears at the second place in the first clause in unnegated form and at the third place in the second clause in negated form. The thick colored matching corresponds to a true variable.

Assume without loss of generality that $\{c_i x_{i_3}, u_{j_1} w_{j_1}\} \subseteq M$. Also, the edges $x_{i_1} y_{i_1}$ and $x_{i_2} y_{i_2}$ must be in M, because they are the top-ranked edges of y_{i_1} and y_{i_2}, respectively. We now claim that $u_{j_2} w_{j_2} \in M$ as well.

Suppose $u_{j_2} w_{j_2} \notin M$. Since M is a maximal matching, $u_{j_2} y_{i_k} \in M$ for some i_k. Based on the above described structure of the clause gadgets, the edges $x_{j_k} c_j, x_{j_{k+1}} y_{j_{k+1}}$, and $x_{j_{k+2}} y_{j_{k+2}}$ are in M, where the subscripts are taken modulo 3. Consider the following augmenting path ρ with respect to M:

$$\rho = \langle w_{j_2}, u_{j_1}, w_{j_1}, u_{j_2}, y_{j_k}, x_{j_k}, c_j, x_{j_{k+1}}, y_{j_{k+1}} \rangle.$$

But now $M \oplus \rho$ is more popular than M, which contradicts the popularity of M. Thus

$u_{j_2}w_{j_2} \in M$.

An analogous argument proves that if $u_{j_2}w_{j_1} \in M$ for some j, then $u_{j_1}w_{j_2}$ has to be in M. The last observation we make is that if y_{i_k} is unmatched in M, then its interconnecting edge leads to a u_{j_t} that is matched to w_{j_1}. Otherwise $y_{i_k}u_{j_t}$ would be labeled $(+1,+1)$ with one vertex unmatched, a contradiction again to the popularity of M. This finishes the proof of Claim 45.■ □

Lemma 7.13. *If there is a truth assignment in $\mathcal{I}$, then there is a popular matching M in G.*

Proof. Here we first construct a matching M in the graph G as described below and then show that it is popular. Initially $M = \emptyset$. For each j, where $1 \leq j \leq n$, if $v_j = \texttt{true}$ in the assignment, then add $u_{j_1}w_{j_1}$ and $u_{j_2}w_{j_2}$ to M, else add $u_{j_1}w_{j_2}$ and $u_{j_2}w_{j_1}$ to M. For each i, where $1 \leq i \leq m$, in the gadget corresponding to clause C_i, any true literal is chosen (say, the k-th literal) and y_{i_k}, representing its appearance, is left unmatched. Moreover, edges $x_{i_k}c_i, x_{i_{k+1}}y_{i_{k+1}}$ and $x_{i_{k+2}}y_{i_{k+2}}$ (where the subscripts are taken modulo 3) are added to M. No interconnecting edge appears in M. This finishes the description of M.

Claim 46. *The matching M is popular in G.*

Proof. Suppose M is not popular. Then there is another matching M' that is more popular than M. This can only happen if $M \oplus M'$ contains a component ρ such that the number of vertices in ρ that prefer M' to M is more than those that prefer M to M'. To achieve this, the matching M' should contain at least one edge labeled either $(+1,+1)$ or $(+1,0)$. We now analyze the cases based on the occurrences of such "positive" edges.

Since we started with a truth assignment, no interconnecting edge can be labeled $(+1,+1)$. In fact, it is easy to check that no edge here can be labeled $(+1,+1)$. We now check for the occurrences of edges labeled $(+1,0)$. These can occur at two places: the edge $u_{j_t}w_{j_1}$ for any $1 \leq j \leq n$ and the edge $x_{i_k}c_i$ for any $1 \leq i \leq m$.

Case 1. Suppose $u_{j_2}w_{j_1}$ is labeled $(+1,0)$. This happens if v_j is $\texttt{true}$ in the truth assignment. We start the augmenting path ρ at $u_{j_2}w_{j_1}$. Augmenting along the 4-cycle is not sufficient to break popularity, therefore, u_{j_1} must be matched along one of its interconnecting edges, say $u_{j_1}y_{i_k}$.

- If y_{i_k} is unmatched, consider the path $\rho = \langle w_{j_2}, u_{j_2}, w_{j_1}, u_{j_1}, y_{i_k}\rangle$. There are two vertices (u_{j_1} and w_{j_2}) that prefer M to $M \oplus \rho$ and two vertices (u_{j_2} and y_{i_k}) that prefer $M \oplus \rho$ to M.

- If y_{i_k} is matched, then extend the path ρ till the unmatched vertex of the i-th variable gadget (call this y_{i_t}). The path ρ is described below:

$$\rho = \langle w_{j_2}, u_{j_2}, w_{j_1}, u_{j_1}, y_{i_k}, x_{i_k}, c_i, x_{i_t}, y_{i_t}\rangle.$$

  So four vertices, i.e., w_{j_2}, u_{j_1}, y_{i_k}, and x_{i_t}, prefer M to $M \oplus \rho$ while three vertices, i.e., u_{j_2}, x_{i_k}, and y_{i_t}, prefer $M \oplus \rho$ to M.

Case 2. Now suppose $x_{i_k}c_i$ is labeled $(+1,0)$. Let us assume that this edge is $x_{i_3}c_i$ and suppose $x_{i_1}c_i \in M$. Consider the alternating path $\rho = \langle y_{i_1}, x_{i_1}, c_i, x_{i_3}, y_{i_3}\rangle$. In the matching $M \oplus \rho$, the vertices x_{i_3} and y_{i_1} are better off while x_{i_1} and y_{i_3} are worse off, i.e., they prefer M to $M \oplus \rho$. In order to collect one more vertex that prefers $M \oplus \rho$, let us extend this alternating path ρ to include $u_{j_k}y_{i_3}$, the interconnecting edge of y_{i_3}. The vertex y_{i_3} still prefers M to $M \oplus \rho$ since y_{i_3} was paired in M to its top-ranked neighbor.

Without loss of generality, let us assume that this interconnecting edge is $u_{j_2}y_{i_3}$. We have two cases here: either $\{u_{j_1}w_{j_1}, u_{j_2}w_{j_2}\} \subseteq M$ or $\{u_{j_1}w_{j_2},\, u_{j_2}w_{j_1}\} \subseteq M$.

- In the first case, the path ρ gets extended to $\langle \cdots, u_{j_2}, w_{j_2}\rangle$. So u_{j_2} prefers $M \oplus \rho$ to M, however w_{j_2} is left unmatched in $M \oplus \rho$, so w_{j_2} prefers M to $M \oplus \rho$.
- In the second case, the path ρ gets extended to $\langle \cdots, u_{j_2}, w_{j_1}, u_{j_1}, w_{j_2}\rangle$. So u_{j_1} prefers $M \oplus \rho$ to M, however both u_{j_2} and w_{j_2} prefer M to $M \oplus \rho$.

We have analyzed all the cases where edges can labeled $(+1,0)$ and we showed that there is no alternating cycle or path ρ containing an edge labeled $(+1,0)$ such that $M \oplus \rho$ is more popular than M. Thus M is popular. ■ □

Theorem 7.11 clearly follows from Lemmas 7.12 and 7.13.

7.5 Conclusion and open problems

In this chapter, we presented an alternative notion of optimality, based on collective voting. We brought up several arguments why popular matchings are closely related to stable matchings and can be seen as a relaxation of them.

First dominant matchings were defined in instances with strictly ordered lists and an even closer connection between stable matchings and dominant matchings was pointed out. Then, the hardness of very restricted instances with ties was shown.

Interestingly, in the case of strictly ordered lists, all polynomial time algorithms currently known for computing *any* popular matching in $G = (V, E)$ compute either a stable matching or a dominant matching in G. The question rises naturally: can one come up with an efficient algorithm that computes a non-dominant and non-stable matching? At the moment, even the complexity of deciding whether such a matching exists, is unknown, although we conjecture it to be a tractable problem.

For the case with ties, when each $w \in W$ either has a single tie of length at most 3 or a strict preference list (and each $u \in U$ has a strict preference list), we showed that the popular matching problem becomes NP-hard. The complexity of the same problem with ties of length at most 2 instead of 3 is open.

Bibliography

[1] D. J. Abraham, P. Biró, and D. F. Manlove. "Almost stable" matchings in the roommates problem. In T. Erlebach and G. Persiano, editors, *Proceedings of WAOA '05: the 3rd Workshop on Approximation and Online Algorithms*, volume 3879 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2006.

[2] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37:1030–1045, 2007.

[3] D. J. Abraham, A. Levavi, D. F. Manlove, and G. O'Malley. The stable roommates problem with globally-ranked pairs. *Internet Mathematics*, 5:493–515, 2008.

[4] H. Ackermann, P. W. Goldberg, V. S. Mirrokni, H. Röglin, and B. Vöcking. Uncoordinated two-sided matching markets. *SIAM Journal on Computing*, 40:92–106, 2011.

[5] G. Askalidis, N. Immorlica, A. Kwanashie, D. F. Manlove, and E. Pountourakis. Socially stable matchings in the Hospitals / Residents problem. In F. Dehne, R. Solis-Oba, and J.-R. Sack, editors, *Algorithms and Data Structures*, volume 8037 of *Lecture Notes in Computer Science*, pages 85–96. Springer Berlin Heidelberg, 2013.

[6] D. Avis. A survey of heuristics for the weighted matching problem. *Networks*, 13:475–493, 1983.

[7] M. Baïou and M. Balinski. Many-to-many matching: stable polyandrous polygamy (or polygamous polyandry). *Discrete Applied Mathematics*, 101:1–12, 2000.

[8] M. Baïou and M. Balinski. Erratum: the stable allocation (or ordinal transportation) problem. *Mathematics of Operations Research*, 27:662–680, 2002.

[9] M. Balinski and T. Sönmez. A tale of two mechanisms: student placement. *Journal of Economic Theory*, 84:73–94, 1999.

[10] P. Berman, M. Karpinski, and A. D. Scott. Approximation hardness of short symmetric instances of MAX-3SAT. Electronic Colloquium on Computational Complexity Report, number 49, 2003.

[11] P. Biró. Matching schemes: online collection, 2014. `http://econ.core.hu/english/res/game_app.html`.

[12] P. Biró, R. W. Irving, and D. F. Manlove. Popular matchings in the marriage and roommates problems. In *Proceedings of CIAC '10: the 7th International Conference on Algorithms and Complexity*, volume 6078 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2010.

[13] P. Biró and S. Kiselgof. College admissions with stable score-limits. *Central European Journal of Operations Research*, pages 1–15, 2013.

[14] P. Biró and F. Klijn. Matching with couples: a multidisciplinary survey. *International Game Theory Review*, 15, article number 1340008, 2013.

[15] P. Biró, D. F. Manlove, and I. McBride. The Hospitals / Residents problem with couples: complexity and integer programming models. In J. Gudmundsson and J. Katajainen, editors, *Experimental Algorithms*, volume 8504 of *Lecture Notes in Computer Science*, pages 10–21. Springer International Publishing, 2014.

[16] P. Biró, D. F. Manlove, and E. J. McDermid. "Almost stable" matchings in the roommates problem with bounded preference lists. *Theoretical Computer Science*, 432:10–20, 2012.

[17] P. Biró, D. F. Manlove, and S. Mittal. Size versus stability in the marriage problem. *Theoretical Computer Science*, 411:1828–1841, 2010.

[18] Y. Blum, A. E. Roth, and U. G. Rothblum. Vacancy chains and equilibration in senior-level labor markets. *Journal of Economic Theory*, 76:362–411, 1997.

[19] S. Braun, N. Dwenger, and D. Kübler. Telling the truth may not pay off: an empirical study of centralized university admissions in Germany. *The B.E. Journal of Economic Analysis and Policy*, 10, article 22, 2010.

[20] A. Caprara, H. Kellerer, and U. Pferschy. A PTAS for the multiple subset sum problem with different knapsack capacities. *Information Processing Letters*, 73:111 – 118, 2000.

[21] K. Cechlárová and T. Fleiner. Stable roommates with free edges. Technical Report 2009-01, Egerváry Research Group on Combinatorial Optimization, Operations Research Department, Eötvös Loránd University, 2009.

[22] C. Chekuri and S. Khanna. A polynomial time approximation scheme for the multiple knapsack problem. *SIAM Journal on Computing*, 35:713–728, 2005.

[23] B. Chen, S. Fujishige, and Z. Yang. Decentralized market processes to stable job matchings with competitive salaries. Discussion papers, Department of Economics, University of York, 2011.

[24] Y. Chen and T. Sönmez. Improving efficiency of on-campus housing: an experimental study. *American Economic Review*, 92:1669–1686, 2002.

[25] Á. Cseh and B. C. Dean. Improved algorithmic results for unsplittable stable allocation problems. *Journal of Combinatorial Optimization*, pages 1–15, 2015.

[26] Á. Cseh, C.-C. Huang, and T. Kavitha. Popular matchings with two-sided preferences and one-sided ties. In M. M. Halldórsson, K. Iwama, N. Kobayashi, and

B. Speckmann, editors, *Automata, Languages, and Programming*, volume 9134 of *Lecture Notes in Computer Science*, pages 367–379. Springer Berlin Heidelberg, 2015.

[27] Á. Cseh and T. Kavitha. Popular edges and dominant matchings. *CoRR*, abs/1508.00614, 2015.

[28] Á. Cseh, J. Matuschke, and M. Skutella. Stable flows over time. *Algorithms*, 6:532–545, 2013.

[29] Á. Cseh and M. Skutella. Paths to stable allocations. In R. Lavi, editor, *Algorithmic Game Theory*, volume 8768 of *Lecture Notes in Computer Science*, pages 61–73. Springer Berlin Heidelberg, 2014.

[30] g. Cseh and D. F. Manlove. Stable marriage and roommates problems with restricted edges: Complexity and approximability. In M. Hoefer, editor, *Algorithmic Game Theory*, volume 9347 of *Lecture Notes in Computer Science*, pages 15–26. Springer Berlin Heidelberg, 2015.

[31] W. Darrach. Letter. *Bulletin of the Association of American Medical Colleges*, 68(2), January 1927.

[32] B. A. Davey and H. A. Priestley. *Introduction to lattices and order.* Cambridge university press, 2002.

[33] B. C. Dean, M. X. Goemans, and N. Immorlica. The unsplittable stable marriage problem. In G. Navarro, L. E. Bertossi, and Y. Kohayakawa, editors, *Proceedings of IFIP TCS '06: the 4th IFIP International Conference on Theoretical Computer Science*, volume 209 of *IFIP — International Federation for Information Processing*, pages 65–75. Springer, 2006.

[34] B. C. Dean and S. Munshi. Faster algorithms for stable allocation problems. *Algorithmica*, 58:59–81, 2010.

[35] E. Diamantoudi, E. Miyagawa, and L. Xue. Random paths to stability in the roommate problem. *Games and Economic Behavior*, 48:18–28, 2004.

[36] V. M. F. Dias, G. D. da Fonseca, C. M. H. de Figueiredo, and J. L. Szwarcfiter. The stable marriage problem with restricted pairs. *Theoretical Computer Science*, 306:391–405, 2003.

[37] Y. Dinitz, N. Garg, and M. X. Goemans. On the single-source unsplittable flow problem. *Combinatorica*, 19:17–41, 1999.

[38] T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45:233–284, 1992.

[39] T. Feder. Network flow and 2-satisfiability. *Algorithmica*, 11:291–319, 1994.

[40] T. Fleiner. On the stable b-matching polytope. *Mathematical Social Sciences*, 46:149–158, 2003.

[41] T. Fleiner. On stable matchings and flows. *Algorithms*, 7:1–14, 2014.

[42] T. Fleiner, R. W. Irving, and D. F. Manlove. Efficient algorithms for generalised stable marriage and roommates problems. *Theoretical Computer Science*, 381:162–176, 2007.

[43] A.-T. Gai, D. Lebedev, F. Mathieu, F. de Montgolfier, J. Reynier, and L. Viennot. Acyclic preference systems in P2P networks. In A. Kermarrec, L. Bougé, and T. Priol, editors, *Proceedings of Euro-Par '07 (European Conference on Parallel and Distributed Computing): the 13th International Euro-Par Conference*, volume 4641 of *Lecture Notes in Computer Science*, pages 825–834. Springer, 2007.

[44] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[45] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11:223–232, 1985.

[46] P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Science*, 20:166–173, 1975.

[47] M. R. Garey and D. S. Johnson. *Computers and Intractability.* Freeman, San Francisco, CA., 1979.

[48] M. X. Goemans, E. L. Li, V. S. Mirrokni, and M. Thottan. Market sharing games applied to content distribution in ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24:1020–1033, 2006.

[49] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms.* MIT Press, 1989.

[50] K. Hamada, K. Iwama, and S. Miyazaki. An improved approximation lower bound for finding almost stable maximum matchings. *Information Processing Letters*, 109:1036–1040, 2009.

[51] C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. *Information and Computation*, 222:180–194, 2013.

[52] R. W. Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6:577–595, 1985.

[53] R. W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics*, 48:261–272, 1994.

[54] R. W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM*, 34:532–543, 1987.

[55] R. W. Irving and D. F. Manlove. The stable roommates problem with ties. *Journal of Algorithms*, 43:85–105, 2002.

[56] R. W. Irving, D. F. Manlove, and S. Scott. The Hospitals / Residents problem with ties. In M. M. Halldórsson, editor, *Proceedings of SWAT '00: the 7th Scandinavian Workshop on Algorithm Theory*, volume 1851 of *Lecture Notes in Computer Science*, pages 259–271. Springer, 2000.

[57] K. Iwama, D. F. Manlove, S. Miyazaki, and Y. Morita. Stable marriage with incomplete lists and ties. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Proceedings of ICALP '99: the 26th International Colloquium on Automata, Languages, and Programming*, volume 1644 of *Lecture Notes in Computer Science*, pages 443–452. Springer, 1999.

[58] W. S. Jewell. *Multi-commodity Network Solutions*. Operations Research Center, University of California, 1966.

[59] S. Kalyanaraman and C. Umans. The complexity of rationalizing matchings. In S. Hong, H. Nagamochi, and T. Fukunaga, editors, *Proceedings of ISAAC '08: the 19th International Symposium on Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 171–182. Springer, 2008.

[60] T. Kavitha. A size-popularity tradeoff in the stable marriage problem. *SIAM Journal on Computing*, 43:52–71, 2014.

[61] T. Kavitha, K. Mehlhorn, D. Michail, and K. Paluch. Strongly stable matchings in time $O(nm)$ and extension to the Hospitals-Residents problem. *ACM Transactions on Algorithms*, 3, article number 15, 2007.

[62] T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412:2679–2690, 2011.

[63] T. Kavitha and M. Nasre. Optimal popular matchings. *Discrete Applied Mathematics*, 157:3181–3186, 2009.

[64] S. Khot and O. Regev. Vertex cover might be hard to approximate to within $2-\varepsilon$. *Journal of Computer and System Sciences*, 74:335–349, 2008.

[65] S. Khuller, S. Mitchell, and V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127:255–267, 1994.

[66] T. Király and J. Pap. A note on kernels and Sperner's Lemma. *Discrete Applied Mathematics*, 157:3327–3331, 2009.

[67] T. Király and J. Pap. Stable multicommodity flows. *Algorithms*, 6:161–168, 2013.

[68] Z. Király. Linear time local approximation algorithm for maximum stable marriage. *Algorithms*, 6:471–484, 2013.

[69] B. Klaus and F. Klijn. Paths to stability for matching markets with couples. *Games and Economic Behavior*, 58:154–171, 2007.

[70] D. Knuth. *Mariages Stables.* Les Presses de L'Université de Montréal, 1976. English translation in *Stable Marriage and its Relation to Other Combinatorial Problems*, volume 10 of CRM Proceedings and Lecture Notes, American Mathematical Society, 1997.

[71] M. Mahdian. Random popular matchings. In J. Feigenbaum, J. C. Chuang, and D. M. Pennock, editors, *Proceedings of EC '06: the 7th ACM Conference on Electronic Commerce*, pages 238–242. ACM, 2006.

[72] D. F. Manlove. Stable marriage with ties and unacceptable partners. Technical Report TR-1999-29, University of Glasgow, Department of Computing Science, January 1999.

[73] D. F. Manlove. *Algorithmics of Matching Under Preferences.* World Scientific, 2013.

[74] D. F. Manlove, R. W. Irving, K. Iwama, S. Miyazaki, and Y. Morita. Hard variants of stable marriage. *Theoretical Computer Science*, 276:261–279, 2002.

[75] D. F. Manlove and G. O'Malley. Paired and altruistic kidney donation in the UK: algorithms and experimentation. *ACM Journal of Experimental Algorithmics*, 19, article number 2.6, 2014.

[76] D. F. Manlove and C. Sng. Popular matchings in the capacitated house allocation problem. In Y. Azar and T. Erlebach, editors, *Proceedings of ESA '06: the 14th Annual European Symposium on Algorithms*, volume 4168 of *Lecture Notes in Computer Science*, pages 492–503. Springer, 2006.

[77] R. M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In E. Laber, C. Bornstein, L. Nogueira, and L. Faria, editors, *Proceedings of LATIN '08: the 8th Latin-American Theoretical Informatics Symposium*, volume 4957 of *Lecture Notes in Computer Science*, pages 593–604. Springer Berlin Heidelberg, 2008.

[78] E. McDermid and R. W. Irving. Popular matchings: structure and algorithms. *Journal of Combinatorial Optimization*, 22:339–358, 2011.

[79] E. McDermid and D. F. Manlove. Keeping partners together: algorithmic results for the Hospitals / Residents problem with couples. *Journal of Combinatorial Optimization*, 19:279–303, 2010.

[80] J. Mestre. Weighted popular matchings. *ACM Transactions on Algorithms*, 10, article number 2, 2014.

[81] G. O'Malley. *Algorithmic Aspects of Stable Matching Problems.* PhD thesis, University of Glasgow, Department of Computing Science, 2007.

[82] M. Ostrovsky. Stability in supply chain networks. *American Economic Review*, 98:897–923, 2008.

[83] C. H. Papadimitriou. On the complexity of the parity argument and other inefficient proofs of existence. *Journal of Computer and System Sciences*, 48:498–532, 1994.

[84] N. Perach, J. Polak, and U. G. Rothblum. A stable matching model with an entrance criterion applied to the assignment of students to dormitories at the Technion. *International Journal of Game Theory*, 36:519–535, 2008.

[85] E. Ronn. NP-complete stable matching problems. *Journal of Algorithms*, 11:285–304, 1990.

[86] A. E. Roth. The evolution of the labor market for medical interns and residents: a case study in game theory. *Journal of Political Economy*, 92:991–1016, 1984.

[87] A. E. Roth. The national residency matching program as a labor market. *Journal of the American Medical Association*, 275:1054–1056, 1996.

[88] A. E. Roth. Deferred acceptance algorithms: history, theory, practice, and open questions. *International Journal of Game Theory*, 36:537–569, 2008.

[89] A. E. Roth and E. Peranson. The redesign of the matching market for American physicians: Some engineering aspects of economic design. *American Economic Review*, 89:748–780, 1999.

[90] A. E. Roth, T. Sönmez, and M. U. Ünver. Pairwise kidney exchange. *Journal of Economic Theory*, 125:151–188, 2005.

[91] A. E. Roth and M. A. O. Sotomayor. *Two-Sided Matching: A Study in Game-Theoretic Modeling and Analysis*, volume 18 of *Econometric Society Monographs*. Cambridge University Press, 1990.

[92] A. E. Roth and J. H. Vande Vate. Random paths to stability in two-sided matching. *Econometrica*, 58:1475–1480, 1990.

[93] U. G. Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54:57–67, 1992.

[94] S. Scott. *A study of stable marriage problems with ties.* PhD thesis, University of Glasgow, Department of Computing Science, 2005.

[95] D. B. Shmoys and É. Tardos. Scheduling unrelated machines with costs. In V. Ramachandran, editor, *Proceedings of SODA '93: the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–454, 1993.

[96] M. Skutella. Approximating the single source unsplittable min-cost flow problem. *Mathematical Programming*, 91:493–514, 2002.

[97] J. Tan. A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12:154–178, 1991.

[98] É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, pages 250–256, 1986.

[99] C.-P. Teo and J. Sethuraman. LP based approach to optimal stable matchings. In M. E. Saks, editor, *Proceedings of SODA '97: the 8th ACM-SIAM Symposium on Discrete Algorithms*, pages 710–719. ACM-SIAM, 1997.

[100] C.-P. Teo and J. Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23:874–891, 1998.

[101] I. Wegener. *Complexity theory: exploring the limits of efficient algorithms*. Springer Science & Business Media, 2005.

[102] D. B. West. *Introduction to Graph Theory*, volume 2. Prentice-Hall Upper Saddle River, 2001.

[103] K. J. Williams. A reexamination of the NRMP matching algorithm. National Resident Matching Program. *Academic Medicine*, 70:470–6; discussion 490–4, June 1995.

[104] K. J. Williams, V. P. Werth, and J. A. Wolff. An analysis of the resident match. *New England Journal of Medicine*, 304:1165–1166, May 1981.

[105] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[106] D. Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3:103–128, 2007.

[107] National Resident Matching Program, Results and Data: 2015 Main Residency Match®. National Resident Matching Program, Washington, DC. 2015.

[108] National Resident Matching Program. About the NRMP. Web document available at `http://www.nrmp.org/match-process/match-algorithm/`. Accessed 25 August 2015.

[109] Organ Donation and Transplantation Directorate, NHS Blood and Transplant website. `http://www.organdonation.nhs.uk`. Accessed 25 August 2015.

[110] Paired Donation Network website. `http://www.paireddonationnetwork.org`. Accessed 25 August 2015.