Graph Matching, Matroid Intersection, and Beyond

Chien-Chung Huang

Graph Matching, Matroid Intersection, and Beyond

Chien-Chung Huang
cchuang@di.ens.fr

The present document consists of the following four components.

1. Introduction and summary of the dissertation (Chapter 1).
2. Five representative papers (Chapters 2-6).

Contents

Chapter 1

Introduction

Given a graph $G = (V, E)$, a subset of edges $M \subseteq E$ is *matching* if no two edges in M share a common endpoint. Graph matching problems can be generally described as finding a matching that satisfies some condition, such as optimality with respect to some objective function, stability [70], popularity [74], or fairness [91]. The most common goal is to ask for a *maximum weight* matching, where we assume that edges are assigned weights and the weight of a matching is the sum of the weights of its edges. Without a doubt, graph matching is one of the most fundamental combinatorial optimisations problems. Some remarkable applications include: shortest paths [117], the Chinese postman problem [114], planar max cut [82], and the travelling salesman problem [25]. Due to applications in online auctions, the importance of matchings has become even more prominent in recent years

A *matroid* M is a pair $(E, \mathcal{I})$, where E is the ground set and $\mathcal{I}$ is a family of subsets of E. Furthermore, the following conditions are satisfied. (1) $\emptyset \in \mathcal{I}$; (2) if $I \subseteq J$ and $J \in \mathcal{I}$, then $I \in \mathcal{I}$; (3) given $I, J \in \mathcal{I}$, and $|J| > |I|$, then there exists an element $e \in J \backslash I$ so that $I \cup \{e\} \in \mathcal{I}$. The sets in $\mathcal{I}$ are called *independent sets*. Given two matroids, $M_1 = (E, \mathcal{I}_1)$ and $M_2 = (E, \mathcal{I}_2)$, a *matroid intersection* problem asks for a common independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ that satisfies some condition. As in the case of matching, the most common objective is to ask for maximum weight common independent set, where the elements in E are assigned weights. Matroids, along with matroid intersection, find numerous applications in combinatorial optimisation, the most well-known being bipartite matching, packing of spanning trees, and finding arborescences in a directed graph [158]. Other applications exist in electric circuit theory [133, 146], rigidity theory [146], and network coding [34].

Besides giving applications in a large number of areas, matching and matroid intersection play a central role in theory, especially in algorithmic design, polyhedral combinatorics, linear programming, and graph theory. In a certain sense, the boundary of $\mathbf{P}$ is to a large extent delineated by these problems: those problems that can be solved in polynomial time are often generalisations of these two, or share similar structures.

1.1 History

When talking about the matching and matroids, the name of Jack Edmonds is unavoidable. He not only solved weighted matching and weighted matroid intersection problems, but also pioneered an entire field. First consider the problem of maximum cardinality matching (i.e., when all weights are uniform). Berge [12] showed that a matching M is of maximum cardinality if and only if there is no augmenting path.[1] Therefore to find a maximum cardinality matching, it suffices to repeat augmenting the matching $O(|V|)$ times. Finding an augmenting path in a bipartite graph is easy to do in polynomial time; however, in a general graph, finding such a path is highly non-trivial. It was Edmonds in 1965 who solved this problem [42], by proposing his famous "blossom" algorithm. It is noteworthy that in this article, he formally introduced the notion of polynomial time algorithms and in a sense anticipated the class of NP-complete problems. In the same year [41], he solved the maximum weight matching problem. An important consequence of his primal-dual algorithm is a linear description of the convex hull of all matchings, namely, the matching polytope. Remarkably, the integrality of his matching polytope does not follow from standard total-unimodularity and is generally regarded as a breakthrough in polyhedral combinatorics.[2]

Matroids were first introduced by Whitney [**?**] (and much less well-known, independently by Nakasawa [**?**]). They capture many common structures across different branches of mathematics (algebra, geometry, and graph theory). Their importance in algorithm design is demonstrated by Edmonds [43, 44, 45]: he showed how a greedy algorithm can be used to find an optimal independent set, proved a mini-max theorem on matroid intersection, gave an efficient algorithm to compute a maximum weight common independent set, and showed that the intersection of the independent set polytopes of two matroids gives exactly the convex hull of the common independent sets. The fact that one can use two matroids to capture a large class of problems and use his algorithm to solve them is really outstanding; the frontier of **P** is impressively pushed back by him.

1.2 After matching and matroid intersection

We now discuss some more developments after matching and matroid intersection problems were resolved by Edmonds. These developments relate to problems that we have investigated or have the plan of studying in the future.

It is known that both graph matching (in a general graph) and matroid intersection are different generalisations of the bipartite graph matching. Since they are both solvable in polynomial time, it is natural to ask for a common generalisation of graph matching and matroid intersection. Such a generalisation is formulated by Lawler [119], as the *matroid parity* problem,[3] where, given disjoint pairs of elements, one finds the largest set of pairs whose union is an independent set in the matroid. Such a problem is known to be unsolvable using a polynomial number of oracle calls (the oracle can answer the query whether a certain set is independent in the matroid or not) [101**?**]. Remarkably,

[1] Indeed Petersen [143] made such an observation as early as in 1891.

[2] After Edmonds, there are also non-constructive proofs showing his linear system describes the matching polytope, see for example [8, 11].

[3] Other common generalisations are also proposed, namely, the *matchoid* [**?**] and the *matroid matching* [**?**]. It turns out that all three formulations are equivalent.

for the special case of a linear matroid, Lovász [**?**] gave a polynomial time algorithm and also gave a mini-max formula. Several well-known applications of his algorithm include finding a maximum size forest in a 3-uniform hyper-graph [**?**], finding the minimum vertex feedback set when the given graph has degree at most 3 [168], and finding the maximum number of disjoint $\mathcal{S}$-paths [**?**].

For long years, the complexity of *weighted* linear matroid parity problem (when the paired elements are given weights) has remained unsolved. An exciting new result was achieved by Iwata and Kobayashi [**?**], who gave a polynomial time algorithm for this problem.

Another subject related to matroids is that of *submodular* functions. Such a function captures the property of "diminishing marginal return" and can be regarded as a discrete version of a convex function. The rank function of a matroid is submodular and in fact many problems in combinatorial optimisation are just special cases of maximising/minimising a submodular function (with or without additional constraints). Some well-known examples include (directed) max-cut, k-coverage, and matroid intersection [158]. Due to its generality in theory and wide application (the maximisation problem sees many applications in machine learning in recent days, examples including document summarisation, sensor placement, and image collection summarisation [1]; the minimisation problem sees applications in discrete optimisation, game theory, queueing theory and information theory [129]), submodular function optimisation received a large amount of attention in the last two decades; notably, Iwata, Fleischer and Fujishige [**?**] and Schrijver [156] have received Fulkerson prize in 2003 for their strongly polynomial time algorithms for minimising a submodular function.

1.3 Contribution of the author

The author works on both exact and approximation algorithms. His main research interests include: stable matching, algorithmic game theory, machine scheduling, and more generally, combinatorial optimisation. His interest on the main topics of this proposal, namely matching, matroid, and submodular function, was developed gradually over the years after the completion of his Ph.D in 2008. In the following subsections, five papers representing his contribution on these topics are summarised.

1.3.1 New algorithms for maximum weight matching (SODA 2012, MOR 2017)

Designing faster algorithms to solve the maximum weight matching problem is one of the most studied subjects in the area of algorithm design. Let $n = |V|$, $m = |E|$, and assume that integral edge weight function $w : E \to \{1, 2, \cdots, W\}$ is given. We summarise the current fastest algorithms.

1. Gabow's algorithm [61], which takes $O(n(m + n \log n))$ time.
2. Duan, Pettie, and Su's algorithm [37], which takes $O(m\sqrt{n} \log nW)$ time.
3. Huang and Kavitha's algorithm, which takes $O(Wn^{\omega})$ or $O(W\sqrt{n}m \log_n \frac{n^2}{m})$ time, where $\omega \approx 2.37$ is the exponent of the matrix multiplication time.

Thus, when W is large, say in the order of $\Omega(n^{\text{polylog}n})$, Gabow's strongly polynomial time algorithm is the choice; when W is not overly small, Duan, Pettie, and Su's algorithm is the best. Our pseudo-polynomial time algorithm is superior to the previous two when the graph is dense enough and W is really small (say in the order of $o(\log n)$.)

Before explaining our approach, let us summarise the previous techniques. The most common approach to solving the maximum weight matching problem is the primal-dual schema—often called the *Hungarian method* [115] for the special case of bipartite graphs. For general graphs, this approach was initiated by Edmonds [41] and various later algorithms, can be regarded as refinements of Edmonds' algorithm. The idea is to build up feasible primal and dual solutions simultaneously and show that in the end, both solutions satisfy complementary slackness conditions and hence by the linear programming duality theorem, the primal solution is a maximum weight matching. A second approach, which can be regarded as the "primal approach," is to maintain a feasible matching and augment it successively to increase its weight until no further augmentation is possible. The work of Cunningham and Marsh [30] (and also Derigs [33]) can be regarded as representative of this approach.

A third approach, though still primal-dual in nature, can be called "weight-decomposition." It was originally proposed by Kao et al. [102] for the special case of bipartite graphs. Roughly speaking, they show that the problem can be reduced to solving a sequence of W maximum cardinality matchings in subgraphs of the original graph. Their algorithm can be described as follows: in the i-th iteration, only edges of weights higher than $W - i$ are considered and, in particular, only edges whose weight minus the vertex potentials (i.e., the dual solution) equals one are retained. Then a maximum cardinality matching is computed in the subgraph and this matching is subsequently used to update the vertex potentials. The final matching and the vertex potentials can be shown to satisfy the complementary slackness conditions. It should be emphasised why this approach leads to some computational efficiency: the unweighted maximum cardinality algorithm is *slightly* more efficient than its weighted counter-part. Therefore, when W is small enough, this approach leads to faster algorithms. Indeed, it can be imagined that in the future, if there is further improvement on the maximum cardinality algorithm, our algorithm for the weighted problem becomes faster as well.

In [90], we showed that the same approach can be generalised to the case of non-bipartite graphs. The main challenge lies in handling the blossoms and the more complicated dual program. A by-product of our algorithm is a new proof showing that the matching polytope is *totally dual integral*. More generally, we prove the following decomposition theorem.

In a graph $G = (V, E)$ with edge weights in $\{1, \ldots, W\}$, we show that the maximum weight of a matching is exactly $\sum_{i=1}^{W} |M_i|$, where for each $i \in \{1, \ldots, W\}$, M_i is a maximum weight matching in the subgraph G_i of G with edge set $E'_i = \{e \in E : w(e) \geq W - (i-1)\}$ and the weight function $w_i : E'_i \to \{1, \ldots, i\}$ defined as $w_i(e) = w(e) - (W - i)$. In particular, we show the following equation:

$$|M_i| = \sum_{u \in V} y_u^i + \sum_{B \in \Omega} z_B^i \left(\frac{|B|-1}{2} \right) - \sum_{u \in V} y_u^{i-1} - \sum_{B \in \Omega} z_B^{i-1} \left(\frac{|B|-1}{2} \right),$$

where $\langle y_u^i, z_B^i \rangle_{(u \in V,\, B \in \Omega)}$ is an integral optimal solution to the dual program for the maximum

weight matching program in the graph G_i (note that y_u^0 and z_B^0 are just 0 for all $u \in V$ and $B \in \Omega$). Since the graph $G_W = G$, the above equation yields the decomposition theorem.

In the journal version of the paper, we also consider the maximum weight bipartite capacitated b-matching problem. Let $G = (A \cup B, E)$ be a bipartite graph with weight function $w : E \to \{1, \cdots, W\}$. Note that G need not be simple, i.e., multi-edges are allowed. Additionally, there is a *quota* $b : A \cup B \to Z_{>0}$ on the vertices and a *capacity* $c : E \to Z_{>0}$ on the edges. A function $M : E \to Z_{\geq 0}$ is a feasible solution if (1) $M(e) \leq c(e)$ for every $e \in E$ and (2) $\sum_{e \in \delta(v)} M(e) \leq b(v)$ for every $v \in A \cup B$. The goal is to find a feasible solution M so that $\sum_{e \in E} w(e)M(e)$ is maximised.

We show that this problem can also be decomposed into W unweighted (and capacitated) versions of the same problem. In particular, we can solve the problem in

1. $O(Wnm)$ time, in the case of simple b-matching (where $c \equiv 1$);
2. $O(W\sqrt{\beta}m)$ time, where $\beta = \sum_{v \in A \cup B} b(e)$;
3. $O(W(n_1 m + n_1^3))$, where $n_1 = \min\{|A|, |B|\}$;
4. $O(W(n_1 m + n_1^2\sqrt{m}))$;
5. $O(W(n_1 m + n_1^2\sqrt{\log C}))$, where $C = \max_{e \in E} c(e)$;
6. $O(Wn_1 m \log(2 + \frac{n_1^2}{m}))$.

As there are many parameters, it is difficult to give a detailed comparison of our algorithms to all the fastest algorithms in the literature. But *in general*, unless the largest capacity $C = \max_{e \in E} c(e)$ is really large, the recent algorithm of Lee and Sidford [122] is the fastest algorithm. Our algorithms are faster than theirs when the graph is very unbalanced, i.e., when $\min\{|A|, |B|\} = o(n)$ and W is $O(\text{polylog}(m, C))$.

1.3.2 Exact and approximation algorithms for maximum weight matroid intersection (SODA 2016, MP accepted)

As discussed above, both weighted and unweighted matroid intersection problems were solved by Edmonds. Recall that in the weighted version, an integral weight function $w : E \to \{1, 2, \cdots W\}$ is defined over the ground set E of the two matroids M_1 and M_2. Since Edmonds, a sequence of papers have proposed algorithms with faster running time and/or simpler analyses. It should be mentioned that in the literature, the complexity of the algorithms is measured by the number of oracle calls. For the unweighted case, the fastest algorithm was due to Cunningham [31], which takes $O(nr^{1.5})$ oracle calls, where $n = |E|$ and r is the smaller rank of M_1 and M_2. For the weighted case, various algorithms [14, 54, 69, 160] take $O(nr^2)$ calls or $O(n^2\sqrt{r}\log rW)$ calls. Note that the gap in the complexity between the unweighted and weighted version is more pronounced than in the matching case. It is thus tempting to apply the same weight-decomposition technique in the previous section.

Our paper [86] achieves exactly this. Here lies the challenge of this task: even though the matroid intersection problem can be written as a linear program, its dual variables are harder to

reason with and to control, unlike the case of graph matching, where the dual variables can be interpreted as "potential" on the vertices and the blossoms. The way we overcome this issue is to adopt the "weight-splitting" approach of Frank [54]. He showed that the dual variables can be replaced by a weight splitting $w = w_1 + w_2$ and the complementary slackness condition for optimality can also be replaced by weight-optimality in w_1 and w_2.

A critical step of our algorithm is the following: on each round, after applying the unweighted matroid intersection algorithm on a subset of elements in E, we update w_1 and w_2; more importantly, two new matroids M_1' and M_2' are also defined based on the weights w_1 and w_2. The newly defined matroids help to maintain the optimality condition in the subsequent rounds that is satisfied in this round.

In summary, our approach gives rise to the following algorithms:

1. with two general matroids, an algorithm taking $O(Wnr^{1.5})$ oracle calls;
2. with two linear matroids (given in the form of two r-by-n matrices), an algorithm taking $O(nr \log r_* + Wnr_*^{\omega-1})$ time, where $r_* \leq r$ is the maximum size of a common independent set;
3. with two graphic matroids, an algorithm taking $O(W\sqrt{r}n \log r)$ time.

These algorithms are faster than previously known algorithms when W is relatively small. For instance, given two general matroids, our algorithm is faster when $W = o(\min\{\sqrt{r}, \frac{n \log r}{r}\})$. Note that after the technical report of our paper was published, Lee-Sidford-Wong [123] used a novel cutting-plane method to develop a new algorithm taking only $O(n^2 \log nW)$ calls. Our algorithm remains faster when $r = o(\sqrt{n})$.

A recent trend in research is to design fast approximation algorithms for fundamental optimisation problems even if they are in **P**. Examples include maximum weight matching [35], shortest paths [166], and maximum flow [24, 108, 124, 159]. Using the algorithms of [21, 31, 68], our decomposition technique delivers a $(1-\varepsilon)$-approximate solution,

1. when given two general matroids, using $O(\varepsilon^{-1}nr^{1.5} \log r)$ oracle calls;
2. when given two linear matroids, using $O(nr \log r_* + \varepsilon^{-1}nr_*^{\omega-1} \log r_*)$ time;
3. when given two graphic matroids, using $O(\varepsilon^{-1}\sqrt{r}n \log^2 r)$ time.

Our approximation algorithms are significantly faster than most exact algorithms. Prior to our results, there is only a simple greedy 1/2-approximation algorithm [100, 111]. We note that for general matroids, after our paper was published, Chekuri and Quanrud [19] improved upon our results: they obtain a $(1-\varepsilon)$-approximate solution using $O(nr\varepsilon^{-2} \log^2 \varepsilon^{-1})$ oracle calls. For the special case of linear and graphic matroid, our approximation algorithms remain the fastest.

1.3.3 Fair matching (FSTTCS 2012, Algorithmica 2016)

We consider a problem in the context of *matching under preferences* (the well-known stable matching problem [70] falls into this category). Let $G = (A \cup B, E)$ be a bipartite graph, where each node

$u \in A \cup B$ has a list ranking its neighbors in an order of preference (ties are allowed). Depending on the applications, various optimality conditions can be imposed. We consider the problem of *fair matching*, defined as follows.

> A matching M is fair if M has maximum cardinality, subject to this, M matches the minimum number of vertices to rank r neighbors, and subject to that, M matches the minimum number of vertices to rank $(r-1)$ neighbors, and so on, where r is the worst rank used in the preference lists of vertices.

Fair matching can be solved in polynomial time by the following reduction: for an edge e with incident ranks i and j, let $w(e) = n^{i-1}+n^{j-1}$. It is easy to see that a maximum cardinality matching of minimum weight (under weight function w) is a fair matching in G. Such a matching can be computed via the maximum weight matching algorithm by resetting e's weight to $4n^r - n^{i-1} - n^{j-1}$, where r is the largest rank used in any preference list. However, this approach can be expensive even if we use the fastest maximum-weight bipartite matching algorithms [39, 56, 63, 138]. The running time will be $O(rmn)$ or $\tilde{O}(r^2 m\sqrt{n})$. Note that these complexities follow from the customary assumption that an arithmetic operation takes $O(r)$ time on weights of the order n^r.

We introduce a combinatorial algorithm to solve the problem of fair matching. This algorithm takes $\tilde{O}(r^* m\sqrt{n})$ or $\tilde{O}(r^* n^\omega)$ time, where r^* is the largest rank used in a fair matching, thus significantly faster than the aforementioned brute-force reduction to weighted matching. Our algorithm is similar *in spirit* to the weight-decomposition approach in the previous two papers: in each round, a subset of edges are considered and a maximum weight matching, under the additional constraint that certain vertices need to be matched, is computed. Such a matching is then used to define the dual variables, which in turn determine the set of edges used in the next round. In particular, we show how to solve the following problem, which can be of independent interest in other contexts, in polynomial time.

> *Generalized minimum weighted vertex cover problem.* Let $G_i = (A \cup B, E_i)$ be a bipartite graph with edge weights given by $w_i : E_i \to \{0, 1, \ldots, c\}$. Let $K_{i-1} \subseteq A \cup B$ satisfy the property that there is a matching in G that matches all $v \in K_{i-1}$. Find a cover $\{y_u^i\}_{u \in A \cup B}$ so that $\sum_{u \in A \cup B} y_u^i$ is minimized subject to (1) for each $e = (a, b)$ in E_i, we have $y_a^i + y_b^i \geq w_i(e)$, and (2) $y_u^i \geq 0$ if $u \notin K_{i-1}$.

1.3.4 Maximising a monotone submodular function with a knapsack constraint under the streaming model (APPROX 2017)

Let $f : 2^E \to \mathbb{R}_+$ be a *monotone non-negative submodular* function. Recall that a function is submodular if it satisfies the *diminishing marginal returns* property, i.e., for all subsets $S \subseteq T \subsetneq E$ and $e \in E \setminus T$, we have

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T).$$

A function is *monotone* if $f(S) \leq f(T)$ for any $S \subseteq T$.

There can be various types of constraints which define the feasible sets of S. The simplest, and probably the most well-known constraint is that of cardinality, namely, up to K elements in E are

allowed. This problem includes the *k-coverage* problem as a special case. By the well-known result of Feige [49], one cannot hope to get an approximation ratio better than $1 - \frac{1}{e} + \varepsilon$, for any $\varepsilon > 0$, unless P=NP. A generalisation of the cardinality constraint would be the knapsack constraint. In the latter, a size function $c : E \to \mathbb{N}$ is given, and the goal is to maximise $f(S)$, under the restriction that $\sum_{e \in S} c(e) \leq K$. For both cardinality and knapsack constraints, it is known that one can achieve the approximation ratio of $1 - \frac{1}{e}$ using a greedy algorithm [52, 164], matching the lower bound.

An interesting question is to consider the same problem in the single-pass streaming model. In this model, elements arrive one by one but our storage space is much smaller than the total input. An element, if discarded, due to the space limitation, is lost forever. A generalisation of this model is that of multiple passes. The storage space is still small compared to the total input size, but one is allowed to scan through the data a (usually very small) number of times. Observe that this setting (single pass or multiple passes) renders most of the techniques in the traditional setting useless, as they typically require random access to the input data.

The main result of this paper is:

1. a single-pass streaming algorithm with approximation ratio $4/11 - \varepsilon \approx 0.363 - \varepsilon$, using $O(K \cdot \text{poly}(\varepsilon^{-1})\text{polylog}(K))$ space.

2. a multiple-pass streaming algorithm with approximation ratio $2/5 - \varepsilon = 0.4 - \varepsilon$, using $O(K \cdot \text{poly}(\varepsilon^{-1})\text{polylog}(K))$ space.

We now summarise the technique. First consider the single-pass model. A straightforward greedy algorithm, where the incoming item is taken only if marginal ratio—defined as its marginal value divided by its size—passes the threshold, if there is still space. But this gives only $\frac{1}{3} - \varepsilon$ approximation. The difficulty in improving further is the following situation: A new arriving item that is relatively large in size passes the marginal-ratio threshold and is part of OPT, but its addition would cause the size of the current set to exceed the capacity K. In this case, we are forced to throw it away, but in doing so, we are unable to bound the approximation ratio of the current set against that of OPT properly.

We propose a branching procedure to overcome this issue. Roughly speaking, when the size of the current set exceeds a certain size (depending on the parameters), we create a secondary set. We add an item to the secondary set only if it passes the marginal-ratio threshold with respect to the original set, but its addition to the original set would violate the size constraint. In the end, whichever set achieves the higher value is returned. In a way, the secondary set serves as a "back-up" with enough space in case the original set does not have it, and this allows us to bound the ratio properly. Such a branching technique leads to a $(4/11 - \varepsilon)$-approximation.

The main bottleneck of the above single-pass algorithm lies in the situation where there is a large item in OPT whose size exceeds $K/2$. To overcome this case, we develop a $(2/3 - \varepsilon)$-approximation algorithm for constraint of a partition matroid of rank 2. Indeed, our technique is more general and can be of independent interest; so let us describe it briefly.

Let $E_{\text{r}} \subseteq E$ be a subset of the ground set E (called "red items"). Consider the problem defined below:

$$\text{maximize} \quad f(S) \quad \text{subject to} \quad c(S) \leq K, \quad |S \cap E_R| \leq 1. \tag{1.1}$$

We show that, given $\varepsilon \in (0, 1]$, an approximation v to $f(\mathsf{OPT})$ with $v \leq f(\mathsf{OPT}) \leq (1+\varepsilon)v$, and an approximation θ of $f(o_{\mathrm{r}})$ for the unique item o_{r} in $\mathsf{OPT} \cap E_{\mathrm{r}}$, we can choose $O(1)$ of the red items so that one of them $e \in E_{\mathrm{r}}$ satisfies that $f(\mathsf{OPT} - o_{\mathrm{r}} + e) \geq (\Gamma(\theta) - O(\varepsilon))v$, where Γ is a piecewise linear function lower-bounded by $2/3$.

Our multiple pass algorithm uses this technique by treating the large item as the red items. A pass is used to choose a good red item and in the next pass a modified version of our single-pass algorithm is used to further collect smaller items.

1.3.5 Mixed popular matchings (SODA 2017)

We consider a problem of matching under preferences. Let $G = (A \cup B, E)$ be a bipartite graph where A is called the set of men, B is called the set of women, and every vertex $u \in A \cup B$ has a preference list ranking its neighbors in a *strict order* of preference. Such a graph G is also referred to as an instance of the *stable marriage* problem [70] with strict and possibly incomplete preference lists. Our interest here, however, is not the stability of a matching. It is rather the *popularity*, a notion introduced by Gärdenfors [74]. A vertex $u \in A \cup B$ *prefers* matching M to matching M' if u is matched in M and unmatched in M' or it is matched in both and $M(u)$ ranks higher than $M'(u)$ in u's preference list. For any two matchings M and M' in G, let $\phi(M, M')$ be the number of vertices that prefer M to M'. We say M is *more popular than* M' if $\phi(M, M') > \phi(M', M)$.

Definition 1.3.1. *A matching M is* popular *if $\phi(M, M') \geq \phi(M', M)$ for every matching M' in G, i.e., $\Delta(M, M') \geq 0$ where $\Delta(M, M') = \phi(M, M') - \phi(M', M)$.*

By this definition, a popular matching is a sort of "global stable" matching. Popular matching has a strange history, much less glamorous, compared with its counterpart, the stable matching. After Gärdenfors' paper, the notion of popular matching seemed almost forgotten by the world. Its revive was thanks to a paper in SODA 2005 [3], where a polynomial algorithm is given for deciding the existence of a popular matching when only one side has preferences and can vote (e.g., agents and houses). After this paper, a plethora of papers have been published on various aspects of popular matching, for both one-sided and two-sided preferences, in theoretical computer science. Some notable results include: a polynomial time algorithm for computing a maximum cardinality popular matching [103] with the two-sided preferences (i.e., an instance of the stable marriage problem); the existence of "mixed popular matching" (definition below) for both one-sided or two-sided preferences [105]. We remark that at present, it is unknown whether there is a polynomial time algorithm to compute a maximum utility (weight) popular matching (except for the case all utilities are uniform—which is just maximum cardinality popular matching).

We now explain what a mixed popular matching is. A mixed matching is a probability distribution over matchings, i.e., a mixed matching $\Pi = \{(M_0, p_0), \ldots, (M_k, p_k)\}$, where $M_0, \ldots, M_k$ are matchings in G and $\sum_{i=0}^{k} p_i = 1$, $p_i \geq 0$ for all i, and the utility of Π is $\sum_{i=0}^{k} p_i \cdot w(M_i)$.

The function $\phi(M, M')$ defined earlier easily extends to $\phi(\Pi, M')$ as follows: $\phi(\Pi, M') = \sum_{i=0}^{k} p_i \cdot \phi(M_i, M')$, where $\Pi = \{(M_0, p_0), \ldots, (M_k, p_k)\}$. The definition of $\phi(M', \Pi))$ is analogous.

Definition 1.3.2. *A mixed matching Π is popular if $\phi(\Pi, M') \geq \phi(M', \Pi)$ for all matchings M' in G, i.e., $\Delta(\Pi, M') \geq 0$, where $\Delta(\Pi, M') = \phi(\Pi, M') - \phi(M', \Pi)$.*

Suppose $\Lambda = \{(N_0, q_0), \cdots, (N_h, q_h)\}$ is another mixed matching. Let $\Delta(\Pi, \Lambda) = \sum_{i=1}^{k} \sum_{j=1}^{h} p_i q_j \Delta(M_i, N_j)$. Then it follows easily from the above definitions that if Π is popular then $\Delta(\Pi, \Lambda) \geq 0$ for all *mixed* matchings Λ in G. Thus a popular mixed matching "beats" all integral and mixed matchings. We remark that it is *not true* that a mixed popular matching is a convex combination of integral popular matchings.

Mixed matchings are closely related to *fractional* matchings. A fractional matching $\vec{x}$ in G is a point in $\mathbb{R}^m_{\geq 0}$ that satisfies $\sum_{e \in \delta(v)} x_e \leq 1$, for every vertex v, where $\delta(v)$ is the set of edges incident on vertex v. In a bipartite graph, a mixed matching is equivalent to a fractional matching (Birkhoff-von Neumann theorem). The polytope $\mathcal{P}_G \subseteq \mathbb{R}^m$ of all popular mixed matchings in $G = (A \cup B, E)$ involves exponentially many constraints. However a compact extended formulation of this polytope was given in [105]. Thus a max-utility popular mixed matching can be computed in polynomial time.

However a potential drawback of generalizing from matchings to mixed matchings is that the optimal solution has become more complex to describe and more difficult to implement. A mixed matching can be interpreted as either a lottery over matchings or a time-sharing arrangement when the mixed matching is viewed as a fractional matching [150]: in the former case, we need access to several random bits to implement a lottery and the latter case involves sub-dividing jobs and assigning several fractional jobs to an applicant. Thus we may need to deal with an unstructured optimal solution. Our main result is the following.

Theorem 1.3.3. *Given an instance $G = (A \cup B, E)$ with strict preference lists and a utility function $w : E \to \mathbb{Q}$, G always has a max-utility popular mixed matching $\Pi = \{(M_0, \frac{1}{2}), (M_1, \frac{1}{2})\}$ and Π can be computed in polynomial time. Moreover, if G admits a perfect stable matching, i.e. a stable matching where no vertex is unmatched, then $\Pi = \{(M, 1)\}$, namely, Π is pure.*

Thus our result implies that to achieve max-utility, we need just a *single random bit* to implement the lottery or we can find a time-sharing arrangement that is simple and organized—every vertex has at most two partners and spends the same amount of time with each. Hence we can find a max-utility popular mixed matching that is highly structured. This theorem also implies a polynomial time algorithm for computing a max-utility popular half-integral matching.

We now explain our technique. What we proved is in fact stronger than Theorem 1.3.3. We show that the polytope $\mathcal{P}_G \subseteq \mathbb{R}^m$ of all popular mixed matchings is *half-integral*. We started by making an observation: the linear program describing the polytope of mixed popular matching, originally introduced in [105] for the one-sided preference, is *self-dual* when the input problem is of two-sided preferences. This essentially means that if we look at the dual program, after replacing all variables by their negative counter-parts (thus a minimisation problem becomes a maximisation

problem), the new program will be identical to the original primal program.[4] This self-duality is useful in our proof: due to complementary slackness condition, assuming that x is mixed popular, if $x_{ab} > 0$, then the inequality corresponding to man a and woman b must be tight.

Our proof borrows some ideas from Teo and Sethuraman [165], who introduced an interesting technique to prove the integrality of the stable matching polytope: for each man a (woman b), line up from 0 to 1 the "boxes" whose widths corresponding to the fractional values $x_{a,b'}$ by the *decreasing* order of women b' on his list (similarly for woman b, we line up boxes whose widths corresponding to $x_{a',b}$ by the *increasing* order of men a' on her list). It can be shown that picking any point in $[0, 1]$, if we match the man a (woman b) to the woman b' (resp. man a') for which $x_{a,b'}$ (resp. $x_{a',b}$) touches this point, the outcome is a stable matching. This fact implies that a fractional stable matching x is a convex combination of integral stable matchings.

We proved that such a technique can also be generalised to the context of popular matching. A new ingredient in our proof is to use the "witness" of a popular matching (a set of $|A \cup B|$ variables corresponding to the vertices, which serve as a succinct certificate) to help us establish the popularity of an integral matching.

[4] This is the only example we are aware of where the linear program for a natural problem has this surprising property.

Chapter 2

New Algorithms for Maximum Weight Matching and a Decomposition Theorem

This paper first appeared in SODA 2012 and its full version appears in Mathematics of Operations Research 2017. It is joint-work with Kavitha Telikepalli.

Abstract We revisit the classical maximum weight matching problem in general graphs with non-negative integral edge weights. We present an algorithm that operates by decomposing the problem into W unweighted versions of the problem, where W is the largest edge weight. Our algorithm has running time as good as the current fastest algorithms for the maximum weight matching problem when W is small. One of the highlights of our algorithm is that it also produces an integral optimal dual solution, thus our algorithm also returns an integral certificate corresponding to the maximum weight matching that was computed.

Our algorithm yields a new proof to the total dual integrality of Edmonds' matching polytope and it also gives rise to a decomposition theorem for the maximum weight of a matching in terms of the maximum size of a matching in certain subgraphs. We also consider the maximum weight capacitated b-matching problem in bipartite graphs with non-negative integral edge weights and show that it can also be decomposed into W unweighted versions of the problem, where W is the largest edge weight. Our second algorithm is competitive with known algorithms when W is small.

2.1 Introduction

The input here is a graph $G = (V, E)$ with edge weights given by the function $w : E \to \{1, 2, \cdots, W\}$. A *matching* M is a subset of E such that no two edges in M share an endpoint. The weight of a matching M is the sum of the weights of the edges in M. Our objective is to find a *maximum weight*

matching in G. We note that a maximum weight matching need not be a maximum cardinality matching; a maximum weight matching has many applications. For instance, suppose V is a set of players, E is the set of possible pairings of players, and the weight of each edge is the utility of pairing its endpoints together. We seek a set of disjoint pairings so that the sum of utilities is maximized, in other words, what we seek is a maximum weight matching in G.

A closely related problem is that of computing a maximum weight perfect matching, where the goal is to find a perfect matching, and subject to that, one with the largest weight. Although the maximum weight matching problem and the maximum weight perfect matching problem can be reduced to each other in polynomial time, the two problems are different and an algorithm designed for one problem may not achieve the same running time in the other problem. See [36] for a more detailed discussion on this issue. The algorithms presented in this paper have running time guarantees only for the maximum weight matching problem.

Matching problems lie at the core of graph theory, polyhedral combinatorics, and linear optimization. Due to their fundamental nature and vast application, in the past decades, intense investigations have been made for the problem. Edmonds' pioneering work back in the 60s especially highlights the intricate correlation between algorithm design and polyhedral characterization. We refer the interested reader to [36, 157] for a history of the various matching algorithms and their performance. For the polyhedral characterization aspect of matchings, see [26, 157].

The most common approach to solving the maximum weight matching problem is the primal-dual schema—often called the *Hungarian method* [115] in the special case of bipartite graphs. For general graphs, this approach was initiated by Edmonds [41] and various later algorithms, such as [59], can be regarded as refinements of Edmonds' algorithm. The idea is to build up feasible primal and dual solutions simultaneously and show that in the end, both solutions satisfy complementary slackness conditions and hence by the duality theorem, the primal solution is a maximum weight matching. Another approach in dealing with the maximum weight matching problem is to maintain a feasible matching and successively augment it to increase its weight, until no more augmentation is possible. The work of Cunningham and Marsh [30] (and also Derigs [33]) can be regarded as representative of this approach.

A different approach, though still primal-dual in nature, was proposed by Kao et al. [102]. In the special case of bipartite graphs, they showed that the problem can be decomposed into W maximum cardinality matchings. Using the fastest maximum cardinality matching algorithms as a subroutine, their algorithm was faster than other algorithms when W is small. Their algorithm can be roughly described as follows. In the i-th iteration, only edges of weights higher than $W - i$ are considered and in particular, only edges whose weight minus the vertex potentials (i.e., the dual solution) equals one are retained. Then a maximum cardinality matching is computed in the subgraph and this matching is subsequently used to update the vertex potentials. The final matching and the vertex potentials can be shown to satisfy the complementary slackness conditions.

In a preliminary version of the paper [87], we showed that the same approach can be generalized to the case of general graphs.[1] The difficulty mostly lies in how to manipulate the "blossoms" of

[1] In Kao et al.[102], a decomposition theorem of the maximum weight matching in bipartite graphs is given and their algorithm is then derived from it. However, in general graphs, the same theorem fails and the correctness of our algorithm rather follows from the complementary slackness conditions. See Section 2.2.2 for details.

Edmonds' algorithm and update the more complicated dual variables. Throughout the paper, let $n = |V|$ and $m = |E|$. We show that by using one of the fastest maximum cardinality matchings as a subroutine [76, 132], we can solve the maximum weight matching problem in $O(W\sqrt{n}m \log_n(n^2/m))$ time or in $O(Wn^\omega)$ time with high probability, where $\omega \approx 2.3728$ is the exponent of matrix multiplication [71].

Table 2.1 below has the running times of the various fastest maximum weight matching algorithms in general graphs. Subsequent to the preliminary version of this paper, Pettie [144] pointed out that the earlier algorithm of Gabow [60] can be shown to achieve the same running time with a much simpler proof; in fact, he showed that Gabow's algorithm also can be regarded as decomposing the problem into W maximum cardinality matching problems. Compared to the previous algorithms [38, 61, 65], Gabow's algorithm [60] and our algorithm (without using the algebraic algorithm [132] as a subroutine) are faster when $W = o(\log n)$, and if the graph is very dense, i.e. $m = \Theta(n^2)$, then the two latter algorithms are faster when $W = o(\log^2 n)$. Compared to the algebraic algorithm [32], Gabow's algorithm and ours are always faster by a poly-log factor.

Table 2.1: A summary of the current fastest maximum weight matching algorithms in $G = (V, E)$, where $|V| = n$ and $|E| = m$. The $\tilde{O}$ notation of Cygan, Gabow and Sankowski's algorithm hides some factors of $\log nW$.

Running Time	Algorithm
$O(n(m + n \log n))$	Gabow [61]
$O(m\sqrt{n \log n} \log nW)$, $O(m\sqrt{n} \log nW)$	Gabow and Tarjan [65], Duan, Pettie, and Su [38]
$\tilde{O}(Wn^\omega)$	Cygan, Gabow, and Sankowski [32]
$O(Wn^\omega)$, $O(W\sqrt{n}m \log_n n^2/m)$	Gabow, Pettie [60, 144], **this paper**

In the present work, we give a new algorithm for solving the maximum weight matching problem in general graphs. This algorithm is significantly different from our previous one and its proof and presentation are much simpler. Additionally, our algorithm produces an integral optimal dual solution. This in turn gives a new proof to the fact that the linear program proposed by Edmonds to describe the matching polytope is totally dual integral. It is well-known that if a matrix A is *totally unimodular* and b, c are integral vectors, then $\max\{c^T x : Ax \leq b, x \geq 0\}$ and $\min\{y^T b : y^T A \geq c^T, y \geq 0\}$ are attained by integral vectors x and y whenever the optima exist and are finite [155]. However total unimodularity is a very strong condition and in fact, the integrality of vectors x and y holds even under the weaker condition that the system of inequalities $Ax \leq b, x \geq 0$ is totally dual integral. Since TDI is a weaker sufficient condition for the polytopes $\{Ax \leq b, x \geq 0\}$ and $\{y^T A \geq c^T, y \geq 0\}$ to be integral (where b and c are integral vectors), it is interesting to know if a given system of inequalities is TDI.

Although the total dual integrality of the constraints describing Edmonds' matching polytope is well-known and there are several proofs for it [26, 30, 85], to the best of our knowledge, ours is the fastest algorithm to compute an integral optimal dual solution when W is small. Note that this dual solution is a witness or certificate to the optimality of our matching.

Our algorithm also gives rise to a decomposition theorem. In a graph $G = (V, E)$ with edge

weights in $\{1,\ldots,W\}$, we show that the maximum weight of a matching is exactly $\sum_{i=1}^{W}|M_i|$, where for each $i \in \{1,\ldots,W\}$, M_i is a maximum weight matching in a subgraph G_i of G with the edge set $E'_i = \{e \in E : w(e) \geq W-(i-1)\}$ and the weight function $w_i : E'_i \to \{1,\ldots,i\}$ defined as $w_i(e) = w(e) - (W-i)$. In particular, we show the following equation:

$$|M_i| = \sum_{u\in V} y_u^i + \sum_{B\in\Omega} z_B^i \left(\frac{|B|-1}{2}\right) - \sum_{u\in V} y_u^{i-1} - \sum_{B\in\Omega} z_B^{i-1}\left(\frac{|B|-1}{2}\right),$$

where $\langle y_u^i, z_B^i\rangle_{(u\in V,\, B\in\Omega)}$ is an integral optimal solution to the dual program for the maximum weight matching program in the graph G_i (note that y_u^0 and z_B^0 are just 0 for all $u \in V$ and $B \in \Omega$). Since the graph $G_W = G$, the above equation yields the decomposition theorem.

Maximum Weight Bipartite Capacitated b-matching.

Let $G = (A \cup B, E)$ be a bipartite graph with weight function $w : E \to \{1,\cdots,W\}$. Note that G need not be simple, i.e., multi-edges are allowed. Additionally, there is a *quota* $b : A \cup B \to Z_{>0}$ on the vertices and a *capacity* $c : E \to Z_{>0}$ on the edges. A function $M : E \to Z_{\geq 0}$ is a feasible solution if (1) $M(e) \leq c(e)$ for every $e \in E$ and (2) $\sum_{e\in\delta(v)} M(e) \leq b(v)$ for every $v \in A \cup B$. The goal is to find a feasible solution M so that $\sum_{e\in E} w(e)M(e)$ is maximized.

We show that this problem can also be decomposed into W unweighted (and capacitated) versions of the same problem. Using Orlin's new maximum flow algorithm [137] as a subroutine, we can solve the problem in $O(Wnm)$ time; in the case of simple b-matching (where $c \equiv 1$), we can use Gabow's algorithm [61] to solve the problem in $O(W\sqrt{\beta}m)$ time, where $\beta = \sum_{v\in A\cup B} b(e)$. Moreover, in the case that the graph G is very "unbalanced", i.e., the number of vertices on one side is much larger than the number on the other side, then we can use the algorithms of Ahuja et al. [5] as a subroutine, to solve the problem, in $O(W(n_1m + n_1^3))$, $O(W(n_1m + n_1^2\sqrt{m}))$, $O(W(n_1m + n_1^2\sqrt{\log C}))$, or $O(Wn_1m\log(2+\frac{n_1^2}{m}))$ time, where $n_1 = \min\{|A|,|B|\}$ and $C = \max_{e\in E} c(e)$. See Table 2.2 for a summary of the fastest algorithms for this problem. As there are many parameters, it is difficult to compare the running time of these algorithms without resorting to case analysis. But *in general*, unless the largest capacity $C = \max_{e\in E} c(e)$ is really large, the recent algorithm of Lee and Sidford [122] is the fastest one. Compared to theirs, our algorithms are faster only when the graph is very unbalanced, i.e., when $\min\{|A|,|B|\} = o(n)$ and W is $O(\text{poly-log}(m,C))$.

Our second algorithm differs from the first in that it does not seek to find the feasible dual solution in each iteration. A final adjustment step is performed to show that the produced dual solution and the primal matching satisfy complementary slackness conditions.

2.2 Maximum Weight Matching in General Graphs

In this section we present our algorithm for computing a maximum weight matching in $G = (V, E)$ with edge weights in $\{1,\ldots,W\}$. As mentioned in Section 2.1, our algorithm also computes an integral optimal dual solution along with the optimal matching M_W. We recall the linear program describing the matching polytope and its dual program below. Let Ω be the set of all odd sized

Table 2.2: A summary of the current fastest maximum weight bipartite capacitated b-matching. Here $\beta = \sum_{v \in A \cup B} b(e)$, $B = \max_{v \in A \cup B} B(e)$, $C = \max_{e \in E} c(e)$, $n_1 = \min(|A|, |B|)$, and $SP_+(n, m, W)$ is the time needed to solve a shortest path problem in a digraph with n vertices, m arcs, and nonnegative edge length function l and $\sum_{e \in E} l(e) \leq W$. Furthermore, the $\tilde{O}$ notation of Gabow and Sankowski's algorithm hides some factors of $\log \beta W$; the $\tilde{O}$ notation of Lee and Sidford's algorithm hides some factors of $\log m$.

Running Time	Algorithm	Note
$O(n \log \beta \cdot SP_+(n, m, W))$	Lawler [119]	
$O(nm \log n^2/m \log nW)$	Goldberg and Tarjan [77, 78]	
$O(m \log n \cdot SP_+(n, m, W))$	Orlin [135, 136]	
$O((\sqrt{\beta}m + \beta \log \beta) \log nW)$	Gabow and Tarjan [63]	
$O((nm + \beta \log \beta) \log nW)$	Gabow and Tarjan [63]	
$O(n_1 m + n_1^3 \log n_1 W)$	Ahuja, Orlin, Stein, and Tarjan [5]	
$O(n_1 m \log(2 + \frac{n_1^2}{m}) \log n_1 W)$	Ahuja, Orlin, Stein, and Tarjan [5]	
$\tilde{O}(\sqrt{n}m \cdot \log^{O(1)} C)$	Lee and Sidford [122]	
$\tilde{O}(W\beta^\omega)$	Gabow and Sankowski [62]	only for simple graphs
$O(Wnm)$	**this paper**	
$O(W\sqrt{\beta}m)$	**this paper**	only for simple b-matching ($c \equiv 1$)
$O(W(n_1 m + n_1^3))$	**this paper**	
$O(W(n_1 m + n_1^2\sqrt{m}))$	**this paper**	
$O(W(n_1 m + n_1^2\sqrt{\log C}))$	**this paper**	
$O(Wn_1 m \log(2 + \frac{n_1^2}{m}))$	**this paper**	

subsets of V of size at least 3, also referred to as *odd sets* of vertices. For any set $B \subseteq V$, let $E[B]$ be the set of edges (x, y) both of whose endpoints x and y belong to B. For any vertex v, let $E(v)$ be the set of edges incident on v.

We first briefly review some classical concepts on which our algorithm is built: Edmonds' blossom algorithm and Gallai-Edmonds decomposition. We highlight the main features of the blossom algorithm. More details can be found in [127, 157]. In this and the next section, $M(u)$ refers to the vertex that is matched to u under the matching M.

Petersen [143] observed as early as in 1891 that a matching M is of maximum cardinality if and only if there is no augmenting path with respect to M. It is not difficult to detect an augmenting path with respect to a given matching in bipartite graphs. But finding such a path in general graphs turns out to be more challenging. To overcome this difficulty, Edmonds introduced the idea of opening/closing blossoms.

Definition 2.2.1. *Let $G = (V, E)$ be the original graph. Let $G(V_1)$ be a* shrunken *graph of G, defined as follows.*

(i) $V_1 \subset V \cup \Omega$, i.e., each vertex $v \in V$ is either in V_1 or it belongs to an odd set in $V_1 \cap \Omega$,

(ii) Edges in $G(V_1)$ are induced by V_1. More precisely, (a, b) is an edge in $G(V_1)$ if and only if there exists an edge $(u, v) \in E$ so that (1) $u = a$ or $u \in a$ (where $a \in V_1 \cap \Omega$), and (2) $v = b$ or $v \in b$ (where $b \in V_1 \cap \Omega$).

$$\max \sum_{e\in E} w_e x_e$$

$$\sum_{e\in E(v)} x_e \le 1 \qquad \forall v \in V$$

$$\sum_{e\in E[B]} x_e \le \frac{|B|-1}{2} \qquad \forall B \in \Omega$$

$$x_e \ge 0 \qquad \forall e \in E.$$

$$\min \sum_{v\in V} y_v + \sum_{B\in\Omega} z_B \frac{|B|-1}{2}$$

$$y_u + y_v + \sum_{B:e\in E[B]} z_B \ge w_e \qquad \forall e = (u,v) \in E$$

$$y_v \ge 0 \qquad \forall v \in V$$

$$z_B \ge 0 \qquad \forall B \in \Omega$$

Suppose $G(V_1)$ is a shrunken graph of G and M_1 is a matching in it. A set of vertices $B = (a_0, a_1, \ldots, a_{2t})$ is a blossom *if (1) there exists a circuit traversing the vertices in B, i.e., $(a_i, a_{(i+1) \bmod 2t+1}) \in E_1$ for $0 \le i \le 2t$, and (2) $M(a_{2i-1}) = a_{2i}$, for $1 \le i \le t$. The first vertex a_0 is called the* base *of the blossom B. (Note that a_0 can be matched to some vertex in $V_1 \setminus B$, or it can be left unmatched.)*

***Closing a blossom**. Suppose $B = (a_0, a_1, \ldots, a_{2t})$ is a blossom. Then closing the blossom B means we form a new shrunken graph $G(V_2)$, where $V_2 = (V_1 \setminus \{a_i\}_{i=0}^{2t}) \cup \{B\}$, and a new matching M_2 in $G(V_2)$ as follows:*

$$M_2(a) = M_1(a) \text{ if } a \notin B \cup \{M_1(a_0)\};\ M_2(B) = M_1(a_0).$$

Notice that once the blossom B is closed in $G_1(V_1)$ to form a new shrunken graph $G(V_2)$, there can be another blossom B' in $G(V_2)$. It can happen that B in $G(V_2)$ (now a vertex in V_2) forms part of B'. In this case, B is said to be *embedded* in B'. A blossom not embedded in any other blossom is an *outermost* blossom. (Note that by definition, an outermost blossom must be a vertex in the last shrunken graph).

Definition 2.2.2. ***Opening a blossom**. Let $G(V_1)$ be an shrunken graph of G derived from G by a sequence of closing blossoms. Let M_1 a matching in $G(V_1)$. Let B be an outermost blossom in V_1 and assume that $B = (a_0, a_1, \cdots, a_{2t})$, where a_is are the nodes corresponding to B when B is closed (note that a_i can be a vertex or a blossom). Then opening the blossom B means that we form a new shrunken graph $G(V_2)$, where $V_2 = (V_1 \setminus \{B\}) \cup \{a_i\}_{i=0}^{2t}$ and create a new matching M_2 in $G(V_2)$ as follows:*

- *$M_2(a) = M_1(a)$ if $a \notin B \cup \{M_1(B)\}$.*
- *If B is unmatched in M_1, then a_0 is unmatched in M_2 as well and $M_2(a_{2i-1}) = a_{2i}, \forall 1 \le i \le t$.*
- *If $M_1(B) = a' \in V_1$, then choose $a_k \in B$ so that there is an edge in E connecting a vertex in a' and a vertex in a_k; furthermore, let*

$$M_2(a_{(k+2i-1) \bmod (2t+1)}) = a_{(k+2i) \bmod (2t+1)},\ \forall 1 \le i \le t;\ M_2(a_k) = a'.$$

We now describe how Edmonds' blossom algorithm works. In each round, it seeks to find an augmenting path by building a Hungarian forest. A Hungarian forest is a disjoint set of trees,

whose roots are unmatched nodes; every vertex in such a tree is connected to the root by an alternating path using the edges in the tree. In the process of building the Hungarian forest, a blossom may be detected. A detected blossom is then closed and the building of the Hungarian forest restarts with respect to the updated graph. After the closing of blossoms, if an augmenting path is found, then the matching is augmented along it. Furthermore, all blossoms are reopened (thus restoring the graph completely) and this round is terminated.

In the last round of the blossom algorithm, no augmenting path will be detected even after the closing of some blossoms. Let $\tilde{G} = (\tilde{V}, \tilde{E})$ denote the final (updated) graph, $\tilde{M}$ the current matching in it, and $\tilde{T}$ the final Hungarian forest that was constructed. Some of the vertices in $\tilde{G}$ can indeed be outermost blossoms. To avoid confusion, we refer to the vertices $\tilde{V}$ in $\tilde{G}$ as *nodes*. Note that $\tilde{M}$ is a maximum cardinality matching in $\tilde{G}$.

By Tutte-Berge formula [12, 167], it can be shown that if we reopen all blossoms, then we have a maximum cardinality matching in the original graph. At the end of the blossom algorithm, we are left with a Hungarian forest and a set of matched edges in $\tilde{M}$ (and the latter cannot be reached by any alternative path starting from a unmatched node). Together they encode the Gallai-Edmonds decomposition [42, 73]:

- in $\tilde{T}$, we have a set of pairwise disjoint trees, whose roots are left unmatched in $\tilde{M}$. Each tree is composed of a set of alternating paths starting from the root. A node is *odd* (similarly, *even*) if there is an alternating path of odd (resp., even) length starting from the root to this node.
- we have also the remaining matched edges in $\tilde{M}$ (that are not part of $\tilde{T}$). The endpoint nodes of these matched edges are *unreachable*.

Furthermore,

- All blossoms are (or are contained in) even nodes.
- There is no edge in the original graph between an even node and an even/unreachable node in $\tilde{V}$ (but notice that if an even node is a blossom, then G has some edges between its members).
- No edge between an odd node and an odd/unreachable node is present in M.

For convenience of presentation, in the following, when we say the Hungarian forest and use the notation $\tilde{T}$, we mean both the disjoint trees and those remaining matched edges not included in them.

We now show a simple proof that Edmonds' algorithm produces a maximum cardinality matching. Note that this proof is different from most found in the textbooks; it highlights the new ingredient in our approach.

Proposition 2.2.3. *Let $\tilde{T}$ be the Hungarian forest at the end of Edmonds' blossom algorithm, with the node set $\tilde{V} \subset V \cup \Omega$ and $\tilde{M}$ the corresponding matching. Then the matching M obtained by opening all blossoms in $\tilde{V}$ is a maximum cardinality matching in G.*

Proof. We show the optimality of M by constructing a dual solution $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ such that this dual solution and M together satisfy complementary slackness conditions. For each vertex v, the value y_v is defined as follows:

- if v is an odd node in $\tilde{T}$, then $y_v = 1$;
- if v is an even node in $\tilde{T}$ or is contained in an even node, then $y_v = 0$;
- let $U \subseteq V$ be the set of remaining vertices – these are the unreachable vertices in $\tilde{T}$. Choose any one of them $u^* \in U$; set $y_{u^*} = 1$ and $y_u = 0$ for the remaining vertices in $u \in U \setminus \{u^*\}$.

For each odd set $B \in \Omega$, the value z_B is defined as follows:

- if $B \in \Omega$ is an outermost blossom in $\tilde{G}$, then $z_B = 1$.
- if $|U \setminus \{u^*\}| \geq 2$, then set $z_{U \setminus \{u^*\}} = 1$; for the remaining odd sets $B \in \Omega$, let $z_B = 0$.

We now verify that $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ is a dual feasible solution in the original graph G: the non-negativity conditions obviously hold. We need to check that all edges are properly covered.

- Edges incident on an odd vertex v are clearly covered since $y_v = 1$.
- It follows from Gallai-Edmonds decomposition that there is no edge between an even node and an even/unreachable node. If an edge $e = (u, v)$ has both endpoints in the same even node B in $\tilde{T}$, then $z_B = 1$.
- If an edge $e = (u, v)$ is incident on an unreachable node u, then the other endpoint v is either an odd or an unreachable node. In the former case, $y_v = 1$; in the latter case, either $v = u^*$ (then $y_{u^*} = 1$) or $e \in E[U \setminus \{u^*\}]$ (then $z_{U \setminus \{u^*\}} = 1$).

We now show that M and the above dual solution $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ satisfy complementary slackness conditions:

- all vertices $v \in V$ with $y_v > 0$ are actually matched in M.
- all odd sets B with $z_B > 0$ have exactly $(|B|-1)/2$ edges of $E[B]$ matched in M; this also includes the odd set $B = U \setminus \{u^*\}$.
- every edge in M is between an odd node and an even node, or between 2 unreachable nodes, or within a blossom; thus all edges $e = (u, v) \in M$ are tight, i.e., $y_u + y_v + \sum_{B : e \in E[B]} z_B = 1$.

Hence by complementary slackness, $\langle y_u, z_B \rangle_{(u \in V,\, B \in \Omega)}$ is dual optimal and M is primal optimal, i.e., M is a maximum cardinality matching in G. □

A more well-known optimal dual (e.g., see [26, Exercise 5.30]) is the following: all dual variables remain the same as defined in the proof above except for the following two changes: (1) all unreachable nodes $v \in U$ have $y_u = 1/2$, and (2) the odd set $U \setminus \{u^*\}$ has $z_{U \setminus \{u^*\}} = 0$. In fact, this dual was used by the earlier version of our algorithm and also by the algorithm of Gabow. We choose not to use it because the final dual solution will only be half-integral here.

Observe that we use the variable $z_{U\setminus\{u^*\}}$ to cover all edges in $E[U \setminus \{u^*\}]$. Roughly speaking, in our algorithm, we will regard the odd set $U \setminus \{u^*\}$ as a "pseudo-blossom". This pseudo-blossom is *shrunk* or contracted into a single node and gets matched to u^* in M – we will show in the next section that this contraction lasts for exactly one iteration: this pseudo-blossom will be reopened during the next iteration. The details are described in the next section.

2.2.1 The Algorithm

Our algorithm runs for W iterations: in the i-th iteration, the algorithm computes a maximum weight matching M_i in a graph $G_i = (V, E'_i)$, where $E'_i = E_W \cup \cdots \cup E_{W+1-i}$ and $E_t = \{e | w(e) = t\}$ for $W + 1 - i \leq t \leq W$. The edge weights in G_i are given by the function $w_i : E'_i \to \{1, \ldots, i\}$ defined as $w_i(e) = w(e) - (W - i)$ for $e \in E'_i$. For simplicity of presentation, in case $e \notin E'_i$, we write $w_i(e) = 0$.

The optimality of the matching M_i in G_i will be established via the dual variables y^i_u, for each $u \in V$, and z^i_B, for each $B \in \Omega$. In the i-th iteration we compute integral values for these dual variables and ensure that the dual feasibility conditions ((2.1) and (2.5)) and complementary slackness conditions ((2.2), (2.3), and (2.4)) are satisfied by these dual values and the matching M_i. This will establish the optimality of M_i and these dual values in G_i.

We will ensure that conditions (2.1)-(2.5) hold for all $i \in \{1, \ldots, W\}$ in our algorithm. In the last iteration, i.e., when $i = W$, the set E'_W of edges in G_W is the same as the original edge set E and its weight function w_W coincides with the original weight function w. This will guarantee that M_W is a maximum weight matching in the original graph G.

$$y^i_u + y^i_v + \sum_{B:\, e \in E[B]} z^i_B \geq w_i(e) \qquad \forall \text{edges } \ e = (u, v) \text{ in } E'_i \tag{2.1}$$

$$y^i_u + y^i_v + \sum_{B:\, e \in E[B]} z^i_B = w_i(e) \qquad \forall \text{edges } \ e = (u, v) \in M_i \tag{2.2}$$

$$y^i_u > 0 \Rightarrow \ u \text{ is matched in } M_i \qquad \forall\ u \in V \tag{2.3}$$

$$z^i_B > 0 \Rightarrow \ \frac{|B|-1}{2} \text{ edges of } E[B] \text{ are in } M_i \qquad \forall\ B \in \Omega \tag{2.4}$$

$$y^i_u \geq 0 \ \text{ and } \ z^i_B \geq 0 \qquad \forall\ u \in V \text{ and } B \in \Omega. \tag{2.5}$$

The graph G_i and its function w_i are introduced mainly to facilitate the proof. In implementation, our algorithm does not work with $G_i = (V, E'_i)$. The actual *working graph* is an *unweighted* graph $H_i = (V_i, F_i)$. The vertex set $V_i \subset V \cup \Omega$ is decided by the previous iterations; the edge set F_i consists of edges $e \in E'_i$ that satisfy $w_i(e) - y^{i-1}_u - y^{i-1}_v - \sum_{B:e \in E(B)} z^{i-1}_B = 1$.

We now describe the i-th iteration: the graph is $H_i = (V_i, F_i)$ and the starting matching is $\tilde{M}_{i-1}$ (inherited from the previous iteration) and it will be the case that $\tilde{M}_{i-1} \subseteq F_i$. Edmonds' blossom algorithm is called to augment $\tilde{M}_{i-1}$ into $\tilde{M}_i$. Let $\tilde{V}_i$ be the resulting node set after the execution of Edmonds' algorithm.[2] Recall that $\tilde{V}_i$ can be partitioned into $\mathcal{O}_i \,\dot\cup\, \mathcal{U}_i \,\dot\cup\, \mathcal{E}_i$, where $\mathcal{O}_i$

[2]Here in the description and also in Step 2(c) of the algorithm, we use Edmonds' algorithm for ease of presentation. In actual implementation, we can use any other maximum cardinality algorithm to find a maximum cardinality $\tilde{M}_i$

is the set of odd nodes, $\mathcal{U}_i$ is the set of unreachable nodes, and $\mathcal{E}_i$ is the set of even nodes.

We then create a pseudo-blossom B_i^* to replace a subset of unreachable nodes, as discussed in the previous section, and a unreachable node u_i^* not in the subset is chosen to be B_i^*'s partner in $\tilde{M}_i$. (Note that the pseudo-blossom B_i^* is also closed as a regular blossom, i.e., the set of unreachable nodes in B_i^* are now replaced by a single node B_i^* and its incident edges are induced by the nodes contained in B_i^* and the nodes not in B_i^*.) We define the next round of dual variables $\langle y_u^i, z_B^i \rangle_{(u\in V,\, B\in\Omega)}$ based on $\langle y_u^{i-1}, z_B^{i-1} \rangle_{(u\in V,\, B\in\Omega)}$ and the decomposition $\mathcal{O}_i \,\dot\cup\, \mathcal{U}_i \,\dot\cup\, \mathcal{E}_i$.

We will guarantee that in the i-th iteration, in H_i, u_{i-1}^* has no other incident edge except (u_{i-1}^*, B_{i-1}^*) (see Inequality (2.7) in Lemma 2.2.4 and the discussion immediately before that lemma). This guarantees that the edge (u_{i-1}^*, B_{i-1}^*) is never part of an augmenting path and the edge is never shrunk into other blossoms in the i-th iteration. Moreover, at the end of i-th iteration, either both B_{i-1}^* and u_{i-1}^* belong to $\mathcal{U}_i$ or the former is in $\mathcal{O}_i$ while the latter is in $\mathcal{E}_i$. In both cases, we will reopen the pseudo-blossom B_{i-1}^*; thus there will be at most one pseudo-blossom at the end of the i-th iteration. Note that opening a pseudo-blossom simply means to match the node originally matched with u_{i-1}^* and restore the other originally matched edges contained in B_{i-1}^*. We now give the algorithm formally below. Recall that for any $e \in E$, $w_i(e) = w(e) - (W - i)$.

1. *Initialization*: $y_v^0 = 0\ \forall v \in V$; $z_B^0 = 0\ \forall B \in \Omega$; $\tilde{M}_0 = \emptyset$; $V_1 = V$.

2. For $i = 1$ to W do

   (a) Let $H_i = (V_i, F_i)$, where $F_i = \{e = (u, v) \in E_i' : w_i(e) - y_u^{i-1} - y_v^{i-1} - \sum_{B: e\in E[B]} z_B^{i-1} = 1\}$.

   (b) Let $\tilde{M}_{i-1}$ be the initial matching in H_i.

   (c) Call Edmonds' blossom algorithm to augment $\tilde{M}_{i-1}$ so as to find a maximum cardinality matching $\tilde{M}_i$ in H_i.

   (d) Let $\tilde{V}_i$ be the resultant node set and let $\tilde{V}_i = \mathcal{O}_i \,\dot\cup\, \mathcal{U}_i \,\dot\cup\, \mathcal{E}_i$, where $\mathcal{O}_i$ (similarly, $\mathcal{U}_i$ or $\mathcal{E}_i$) is the set of odd (resp., unreachable or even) nodes.

   (e) If $i = 1$ then choose an arbitrary node $u_1^* \in \mathcal{U}_1$. If $|\mathcal{U}_1 \setminus \{u_1^*\}| \geq 2$, then replace $\mathcal{U}_1 \setminus \{u_1^*\}$ by a pseudo-blossom B_1^* and let $(B_1^*,\, u_1^*) \in \tilde{M}_1$.

   Else (i.e., $i > 1$) then choose an arbitrary node $u_i^* \in \mathcal{U}_i \setminus \{B_{i-1}^*, u_{i-1}^*\}$.

   - if $|\mathcal{U}_i \setminus \{B_{i-1}^*, u_{i-1}^*, u_i^*\}| \geq 2$, then replace this set by a pseudo-blossom B_i^*; in case $|\mathcal{U}_i \setminus \{B_{i-1}^*, u_{i-1}^*, u_i^*\}| = 1$ and $\mathcal{U}_i \setminus \{B_{i-1}^*, u_{i-1}^*, u_i^*\}$ is a blossom B, then let $B_i^* = B$.
   - let $(B_i^*,\, u_i^*) \in \tilde{M}_i$.

   [we now update the dual values: if no explicit update happens for vertex u, then $y_u^i = y_u^{i-1}$; similarly $z_B^i = z_B^{i-1}$ for any odd set B unless otherwise stated.]

   (f) For each $v \in V$: if $v \in \mathcal{O}_i$ or $v \in B$ where $B \in \mathcal{O}_i$, then set $y_v^i = y_v^{i-1} + 1$.

   (g) For each outermost blossom or pseudo-blossom $B \in \mathcal{O}_i$: set $z_B^i = z_B^{i-1} - 1$.

in H_i, as long as $\tilde{M}_i$ is an augmentation of $\tilde{M}_{i-1}$ (that is, a node matched in $\tilde{M}_{i-1}$ must be matched in $\tilde{M}_i$ as well). Then by building the Hungarian forest according to $\tilde{M}_i$, we can shrink the blossoms and the resultant matching and nodes sets will be $\tilde{M}_i$ and $\tilde{V}_i$.

(h) If u_i^* is a vertex $v \in V$, then $y_v^i = y_v^{i-1} + 1$.
Else (u_i^* is a blossom $B \in \Omega$) set $y_v^i = y_v^{i-1} + 1$ for all $v \in B$ and $z_B^i = z_B^{i-1} - 1$.

(i) If $B_{i-1}^* \in \mathcal{U}_i$, then set $y_v^i = y_v^{i-1} + 1$ for each $v \in B_{i-1}^*$ and set $z_{B_{i-1}^*}^i = z_{B_{i-1}^*}^{i-1} - 1$.

(j) For each outermost blossom $B \in \mathcal{E}_i$ and for $B = B_i^*$: set $z_B^i = z_B^{i-1} + 1$.

(k) If u_{i-1}^* is a blossom and is in $\mathcal{U}_i$, then set $z_{u_{i-1}^*}^i = z_{u_{i-1}^*}^{i-1} + 1$.

(l) Suppose that $B \in \Omega \setminus \{u_i^*\}$ is a pseudo-blossom or an outermost blossom. If $z_B^i = 0$, then open the blossom B and recursively so any other blossom B' in B with $z_{B'}^i = 0$.

(m) Let V_{i+1} be the current node set and let $\tilde{M}_i$ be the corresponding matching.

3. Return the matching M_W by opening all blossoms in $\tilde{M}_W$.

Step 2 is the heart of the above algorithm and this step essentially consists of 2 parts: steps (a)-(e) compute the matching $\tilde{M}_i$ while Steps (f)-(k) set the dual values. The matching $\tilde{M}_i$ gets further updated in step (l) by opening out all outermost blossoms whose z-value becomes 0 and now if there are new "outermost" blossoms (these blossoms were earlier embedded but due to the outermost blossom getting opened, these embedded ones become outermost) with z-value 0, these get opened and so on. The last step defines the node set V_{i+1} to be used in the next iteration.

The steps that update the dual values here are analogous to how dual values are set in Proposition 2.2.3. Here we make $y_u^i = y_u^{i-1} + 1$ for every vertex u in $\mathcal{O}_i$ and this includes vertices that belong to blossoms in $\mathcal{O}_i$. For outermost blossoms B in $\mathcal{O}_i$, we decrease z_B^i by 1. The y-value for vertices in $\mathcal{E}_i$ is unchanged and if B is an outermost blossom in $\mathcal{E}_i$, then we increase z_B^i by 1.

The node u_i^* is treated as an *odd* node while B_i^* is treated as an even node. Thus we make $y_u^i = y_u^{i-1} + 1$ for every vertex u in u_i^* and decrease the z-value of u_i^* by 1 (if u_i^* is a blossom) while the y-value for vertices in B_i^* is unchanged and we increase the z-value of B_i^* by 1. To compensate for this, in the $(i+1)$-st iteration, u_i^* will get treated as an *even* node while B_i^* will get treated as an odd node – this is seen here in steps (j) and (k) where the dual variable values for u_{i-1}^* and B_{i-1}^* are updated: u_{i-1}^* is treated as an even node while B_{i-1}^* is treated as an odd node.

Note that after we shrink the pseudo-blossom B_i^* and match it to u_i^*, the dual variables are set in such a way that in the next iteration, u_i^* has only one incident edge (u_i^*, B_i^*) in H_{i+1}. This can be seen by Inequality (2.7).

Lemma 2.2.4. *For each $1 \le i \le W$, conditions (2.1)-(2.5) stated earlier hold, where M_i is the matching derived from $\tilde{M}_i$ by opening all blossoms; conditions (2.6) and (2.7) stated below hold as well.*

$$\text{There is at most one pseudo-blossom } B_i^* \text{ in } V_{i+1} \text{ and if such a } B_i^* \text{ exists, then } z_{B_i^*}^i = 1. \tag{2.6}$$

$$\text{Given } e = (u, v) \in E_i',\ u = u_i^* \text{ (if } u_i^* \in V) \text{ or } u \in u_i^* \text{ (if } u_i^* \in \Omega), \text{ and } v \notin B_i^* \cup u_i^*, \text{ then } y_u^i + y_v^i + \textstyle\sum_{B: e \in E[B]} z_B^i > w_i(e). \tag{2.7}$$

Proof. We show by induction on i that conditions (2.1)-(2.7) hold for all $1 \leq i \leq W$. The base case is $i = 1$. The matching M_1 is the maximum cardinality matching obtained by running Edmonds' algorithm in the graph H_1 and the setting of dual variables in the first iteration of our algorithm corresponds exactly to the setting of dual variables in Proposition 2.2.3. Thus by the same arguments as in the proof of Proposition 2.2.3, conditions (2.1)-(2.5) hold.

It is also easy to see that condition (2.6) holds as there is at most one pseudo-blossom $B_1^* \in \mathcal{U}_1$ at the end of the first iteration. We now show condition (2.7). Since at the end of Edmonds' algorithm, all blossoms are in even nodes, it follows that $u_1^* \in \mathcal{U}_1$ is a vertex in V and we have $y^1_{u_1^*} = 1$. Let $e = (u_1^*, v) \in E'_1$ where $v \notin B_1^*$. Then v has to be in $\mathcal{O}_1$ (as there are no edges in $\mathcal{U}_1 \times \mathcal{E}_1$ in F_1 and $E'_1 = F_1$). Thus we have $y^1_v = 1$ and so $y^1_{u_1^*} + y^1_v + \sum_{B: e \in E[B]} z^i_B \geq 2 > w_1(e)$ since $w_1(e) = 1$.

We now assume conditions (2.1)-(2.7) hold for $i = k-1$ and show that conditions (2.1)-(2.7) corresponding to $i = k$ hold as well.

Condition (2.1). We know by conditions (2.1) and (2.5) for $i = k-1$ that for any edge $e = (u, v)$ in E'_k, we have $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B \geq w_{k-1}(e)$. We now need to show the following inequality:

$$y^k_u + y^k_v + \sum_{B:\, e \in E[B]} z^k_B \quad \geq \quad w_{k-1}(e) + 1 = w_k(e). \tag{2.8}$$

Consider first the case when $e \in E[B]$, where $B \in \tilde{V}_k \cap \Omega$. If $B \in \mathcal{O}_k$, then $y^k_u = y^{k-1}_u + 1$, $y^k_v = y^{k-1}_v + 1$, while $z^k_B = z^{k-1}_B - 1$. Thus Inequality (2.8) clearly holds. If $B \in \mathcal{E}_k$, then there is some blossom $B' \supseteq B$ in $\tilde{V}_k$ and we set $z^k_{B'} = z^{k-1}_{B'} + 1$, thereby ensuring Inequality (2.8). If $B \in \mathcal{U}_k$, then we have the following cases: (i) $B = u_k^*$, (ii) $B = B_k^*$ or $B \subset B_k^*$, (iii) $B = u_{k-1}^*$, (iv) $B = B_{k-1}^*$. Note that cases (i) and (iv) are analogous to $B \in \mathcal{O}_k$ and cases (ii) and (iii) are analogous to $B \in \mathcal{E}_k$. Thus Inequality (2.8) holds when $e \in E[B]$.

We now consider the case when u and v are not inside the same blossom in $\tilde{V}_k$. If one of u, v is in $\mathcal{O}_k$, then $y^k_u + y^k_v \geq y^{k-1}_u + y^{k-1}_v + 1$. Thus Inequality (2.8) holds. If both u and v are in $\mathcal{E}_k$ and the edge (u, v) is in E'_k, then it has to be the case that $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B > w_{k-1}(e)$ (as there are no edges of F_k in $\mathcal{E}_k \times \mathcal{E}_k$ other than edges inside blossoms). Since $y^k_u \geq y^{k-1}_u$, $y^k_v \geq y^{k-1}_v$, $z^k_B = z^{k-1}_B = 0$ for every odd set B containing e, and $w_{k-1}(e) + 1 = w_k(e)$, Inequality (2.8) again holds. The only case left is when one of u, v is in $\mathcal{U}_k$ and the other is in $\mathcal{E}_k \cup \mathcal{U}_k$.

Since F_k has no edges in $\mathcal{E}_k \times \mathcal{U}_k$, this means that $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B > w_{k-1}(e)$ and we again have $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B \geq w_{k-1}(e) + 1 = w_k(e)$. So Inequality (2.8) holds. Finally, if both u and v are in $\mathcal{U}_k$ and by listing all subcases here, it is easy to see that Inequality (2.8) holds. We show one particular subcase, which is when $u = u_{k-1}^*$ (or $u \in u_{k-1}^*$). Here we use condition (2.7) for $i = k-1$ which states that if $v \notin B_{k-1}^*$, then $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B > w_{k-1}(e)$. Thus Inequality (2.8) holds when $v \notin B_{k-1}^*$ and when $v \in B_{k-1}^*$, Inequality (2.8) holds since $y^k_v = y^{k-1}_v + 1$.

Condition (2.2). Let $e = (u, v)$ be an edge in M_k. It follows from the definition of the edge set F_k that $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B = w_{k-1}(e)$. By going through the same case analysis as in the proof of Inequality (2.8), we can show that $y^k_u + y^k_v + \sum_{B:\, e \in E[B]} z^k_B = w_k(e)$. For instance, when $u, v \in V_k$ (i.e., u and v do not belong to any odd set $B \in V_k$) and say $u \in \mathcal{O}_k$, then since an odd

node can only be matched to an even node, it follows that $v \in \mathcal{E}_k$ and thus $y_u^k = y_u^{k-1} + 1$ while $y_v^k = y_v^{k-1}$ and $z_B^k = z_B^{k-1} = 0$ for any odd set B containing (u, v). Thus we have

$$y_u^k + y_v^k + \sum_{B:\, e \in E[B]} z_B^k = (y_u^{k-1} + 1) + y_v^{k-1} + \sum_{B:\, e \in E[B]} z_B^{k-1} \;=\; w_{k-1}(e) + 1 \;=\; w_k(e).$$

Condition (2.3). We are given that $y_u^k > 0$ and there are two cases here: $y_u^{k-1} > 0$ or $y_u^{k-1} = 0$. Consider the former case – here it follows from condition (2.3) for $i = k-1$ that u was matched in M_{k-1} and we now show that u continues to be matched in M_k. We first claim that all edges of $\tilde{M}_{k-1}$ are present in the edge set F_k, i.e., step 2(b) is correct. This follows from condition (2.2) for $i = k-1$: for each edge $e = (u, v) \in M_{k-1}$, we have

$$w_k(e) - y_u^{k-1} - y_v^{k-1} - \sum_{B:e \in E[B]} z_B^{k-1} = (w_{k-1}(e) + 1) - y_u^{k-1} - y_v^{k-1} - \sum_{B:e \in E[B]} z_B^{i-1} = 1.$$

Thus edge $e \in F_k$. Now condition (2.3) follows from the fact that $\tilde{M}_k$ is augmented from $\tilde{M}_{k-1}$ and the fact that (pseudo-)blossom opening guarantees that vertices matched in M_{k-1} remain matched in M_k. Now consider the case $y_u^{k-1} = 0$. For y_u^k to become positive, it must be the case that $u \in \mathcal{O}_k \cup B_{k-1}^* \cup u_k^*$ (or $u = u_k^*$). In all three cases, u is matched in M_k.

Condition (2.4). This is similar to the above proof that condition (2.3) holds. We are given that $z_B^k > 0$ and if z_B^{k-1} is also positive, then it means that the odd set $B \in V_k$ and B remains a blossom throughout the k-th iteration. By the definition of opening a blossom, it follows that when M_k is derived from $\tilde{M}_k$, there are $(|B|-1)/2$ edges of $E[B]$ in M_k. If $z_B^{k-1} = 0$, then it means that the (pseudo-)blossom B was newly formed in the k-th iteration. So $B \in \tilde{V}_k \cap \mathcal{E}_k$ or $B = B_k^*$ or $B = u_{k-1}^*$ and it is easy to see that in all three cases, $(|B|-1)/2$ edges in $E[B]$ are present in M_k.

Condition (2.5). The non-negativity of y_u^k for all vertices u is clear because $y_u^k \geq y_u^{k-1}$ for each $u \in V$ and by condition (2.5) for $i = k-1$, we have $y_u^{k-1} \geq 0$ for all $u \in V$. Regarding z_B^k, we claim that at the end of the $(k-1)$-st iteration, all outermost blossoms $B \in V_k$, along with $B = B_{k-1}^*$, have $z_B^k > 0$, with the possible exception of u_{k-1}^*. This is because of Step 2(l) where we recursively opened up all outermost blossoms and pseudo-blossoms B that satisfy $z_B^{k-1} = 0$. Thus at the beginning of the k-th iteration, all outermost blossoms (other than possibly u_{k-1}^*) and pseudo-blossoms B satisfy $z_B^k > 0$. Hence in spite of decreasing by 1 the z^k-value of some outermost blossoms and pseudo-blossom B_{k-1}^*, at the end of the k-th iteration we maintain the condition that $z_B^k \geq 0$ for all $B \in \Omega$.

Condition (2.6). By condition (2.7) for $i = k-1$, in the edge set E'_{k-1}, the only edge incident on the node u_{k-1}^* is (u_{k-1}^*, B_{k-1}^*) in H_i. Suppose there is some other edge $e = (u, v)$, where $u = u_{k-1}^*$ (or $u \in u_{k-1}^*$) and $v \notin B_{k-1}^* \cup u_{k-1}^*$, in the edge set F_k. Then it has to be the case that

$$y_u^{k-1} + y_v^{k-1} + \sum_{B:\, e \in E[B]} z_B^{k-1} \;=\; w_{k-1}(e).$$

Since $e \in E'_k \setminus E'_{k-1}$, we have $w_{k-1}(e) = 0$ while we know that $y_u^{k-1} \geq 1$ since $u = u_{k-1}^*$ (or $u \in u_{k-1}^*$). This contradicts the above equation as y_v^{k-1} and z_B^{k-1} are non-negative for all $v \in V$

and $B \in \Omega$. Thus the only edge incident on u^*_{k-1} in the graph H_k is the edge $(u^*_{k-1}, B^*_{k-1}) \in \tilde{M}_{k-1}$. Hence in the k-th iteration of the algorithm, neither u^*_{k-1} nor B^*_{k-1} belongs to any augmenting path and so these nodes also cannot become a part of a newly formed blossom. Thus they remain matched to each other during the k-th iteration.

If B^*_{k-1} is a pseudo-blossom, then by condition (2.6) for $i = k-1$, we have $z^{k-1}_{B^*_{k-1}} = 1$. So $z^k_{B^*_{k-1}}$ becomes 0 because either $B^*_{k-1} \in \mathcal{O}_k$ or $B^*_{k-1} \in \mathcal{U}_k$; hence we open up B^*_{k-1} at the end of the k-th iteration. By condition (2.7) for $i = k-1$, we also know that there is no other pseudo-blossom at the end of the $(k-1)$-st iteration. Thus there is at most one pseudo-blossom B^*_k at the end of the k-th iteration. Finally, if B^*_k is a pseudo-blossom, then $z^{k-1}_{B^*_k} = 0$, so $z^k_{B^*_k} = 1$.

Condition (2.7). Suppose $e = (u, v)$ is an edge in E'_k with one endpoint $u \in u^*_k$ and the other endpoint $v \notin B^*_k \cup u^*_k$ (for simplicity of exposition, we assume u^*_k is a blossom here; the same proof holds when u^*_k is a vertex also). We consider two cases here: (i) $v \in \mathcal{E}_k \cup u^*_{k-1}$ and (ii) $v \in \mathcal{O}_k \cup B^*_{k-1}$.

We know that F_k has no edges in $\mathcal{U}_k \times \mathcal{E}_k$, also we have seen in the proof of condition (2.6) that the only edge incident on u^*_{k-1} in the graph H_k is the edge (u^*_{k-1}, B^*_{k-1}). Thus in case (i), i.e. when $v \in \mathcal{E}_k \cup u^*_{k-1}$, there is no edge (u, v) in F_k; hence we have $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B > w_{k-1}(e)$. Since $y^k_u = y^{k-1}_u + 1$, $y^k_v \geq y^{k-1}_v$, $\sum_{B:\, e \in E[B]} z^k_B = \sum_{B:\, e \in E[B]} z^{k-1}_B = 0$ (as there is no blossom $B \supseteq \{u, v\}$ either in V_k or in V_{k+1}) and $w_k(e) = w_{k-1}(e) + 1$, we have $y^k_u + y^k_v + \sum_{B:\, e \in E[B]} z^k_B > w_k(e)$.

We now consider case (ii). In this case $v \in \mathcal{O}_k \cup B^*_{k-1}$, so $y^k_v = y^{k-1}_v + 1 \geq 1$. We know that $y^k_u = y^{k-1}_u + 1 \geq 1$. If $e \notin E'_{k-1}$, then $w_k(e) = 1$. Thus we have $y^k_u + y^k_v + \sum_{B:\, e \in E[B]} z^k_B \geq 2 > w_k(e)$.

So suppose $e \in E'_{k-1}$. We have $y^k_u + y^k_v + \sum_{B:\, e \in E[B]} z^k_B = (y^{k-1}_u + 1) + (y^{k-1}_v + 1) + \sum_{B:\, e \in E[B]} z^{k-1}_B$. Since $y^{k-1}_u + y^{k-1}_v + \sum_{B:\, e \in E[B]} z^{k-1}_B \geq w_{k-1}(e)$ for all edges in E'_{k-1}, we have $y^k_u + y^k_v + \sum_{B:\, e \in E[B]} z^k_B \geq w_{k-1}(e) + 2 \geq w_k(e) + 1$. This finishes the proof that conditions (2.1)-(2.7) hold for $i = k$ as well. □

So for each i, M_i is a matching in G_i and $\langle y^i_u, z^i_B \rangle_{(u \in V,\, B \in \Omega)}$ is a setting of dual variables such that conditions (2.1)-(2.5) are satisfied. This immediately proves that M_i is an optimal solution for the primal program of the i-th iteration, in other words, M_i is a maximum weight matching in the graph G_i. Similarly, $\langle y^i_u, z^i_B \rangle_{(u \in V,\, B \in \Omega)}$ is an (integral) optimal solution for the dual program of the i-th iteration. Hence we can conclude Theorem 2.2.5.

Theorem 2.2.5. *The matching M_W returned by the algorithm is a maximum weight matching. Furthermore, the variables $\langle y^W_u, z^W_B \rangle_{(u \in V,\, B \in \Omega)}$ is an integral optimal solution for the dual program. Thus the linear program describing the matching polytope is totally dual integral.*

Since the maximum weight matching problem in a graph $G = (V, E)$ with edge weights in $\{1, \ldots, W\}$ can be solved as W maximum cardinality matching problems, we can draw the following computational conclusion. Recall that m and n are the number of edges and number of vertices in G, respectively.

Theorem 2.2.6. *The maximum weight matching problem in G can be solved in $O(W\sqrt{n}m \log_n(n^2/m))$ time, or in $O(Wn^\omega)$ time with high probability, using the algorithms of [76, 132] as a subroutine, where $\omega \approx 2.3728$ is the exponent of matrix multiplication.*

Proof. The bottleneck in each iteration is in step 2(c), where the maximum cardinality matching in

H_i gets computed, and we need to spend $O(\sqrt{n}m\log_n(n^2/m))$ or $O(n^\omega)$ time, using the algorithms of [76, 132]. Each of the other parts of Step 2 can be done in $O(m)$ time.

In case we do not use Edmonds' blossom algorithm but use Mucha and Sankowski's algebraic algorithm in step 2(c), then to ensure condition (2.3), where we claim that the new maximum cardinality matching in H_i is augmented from $\tilde{M}_{i-1}$, we can do the following: first find *any* maximum cardinality matching in H_i and let its cardinality be t. Create $|V_i|-2t$ dummy vertices and connect each of them to all nodes in V_i that are left unmatched by $\tilde{M}_{i-1}$. It is easy to see that there is now a perfect matching and we can find it by running the maximum cardinality matching algorithm again. Moreover, the perfect matching so found must guarantee that only the nodes in V_i left unmatched by $\tilde{M}_{i-1}$ can be matched to the dummy vertices and the rest of the matching is the desired maximum cardinality matching $\tilde{M}_i$ in H_i. Then in step 2(d) we can build the Hungarian forest according to $\tilde{M}_i$ to define the resulting vertex set $\tilde{V}_i = \mathcal{O}_i \dot{\cup} \mathcal{U}_i \dot{\cup} \mathcal{E}_i$ in $O(m)$ time. □

2.2.2 Consequences of the above algorithm: a decomposition theorem

Our algorithm gives rise to the following decomposition theorem in a graph $G = (V, E)$ with edge weights in $\{1, \ldots, W\}$. Define graphs $G_1, \ldots, G_W$ as follows: $G_i = (V, E'_i)$ where $E'_i = \{e \in E : w(e) \geq W - (i-1)\}$ with edge weight function $w_i(e) = w(e) - (W - i)$ for each $e \in E'_i$.

Theorem 2.2.7. *There exist matchings $M_1, \ldots, M_W$ and dual solutions $\langle y^i_u, z^i_B\rangle_{(u\in V,\, B\in\Omega)}$ for $i = 1, \ldots, W$ such that the following properties hold:*

1. *For $1 \leq i \leq W$, M_i is a maximum weight matching in G_i and $\langle y^i_u, z^i_B\rangle_{(u\in V,\, B\in\Omega)}$ is an optimal dual solution.*
2. *For $1 \leq i \leq W$, $\langle y^i_u, z^i_B\rangle_{(u\in V,\, B\in\Omega)}$ is an integral solution; furthermore, the set of odd sets B with $z^i_B > 0$ forms a laminar family.*
3. *For $1 \leq i \leq W$, $|M_i| = \sum_{u\in V} y^i_u + \sum_{B\in\Omega} z^i_B \left(\frac{|B|-1}{2}\right) - \sum_{u\in V} y^{i-1}_u - \sum_{B\in\Omega} z^{i-1}_B \left(\frac{|B|-1}{2}\right)$, where $y^0_u = 0\ \forall\, u \in V$ and $z^0_B = 0\ \forall\, B \in \Omega$.*
4. *The maximum weight of a matching in G is equal to $\sum_{i=1}^{W} |M_i|$.*

Proof. For $1 \leq i \leq W$, we will show that the matching M_i and the dual solution $\langle y^i_u, z^i_B\rangle_{(u\in V,\, B\in\Omega)}$ computed in the i-th iteration of our algorithm satisfy all the 4 parts of the above theorem. Part 1 is a corollary of Lemma 2.2.4.

Part 2 follows in a straightforward manner from our algorithm – it is easy to see that y^i_u and z^i_B are integral for all vertices u and odd sets B. The fact that the sets B with $z^i_B > 0$ form a laminar family follows from how the node set $\tilde{V}_i$ is formed and how the z-values are assigned.

We now show part 3 of the above theorem. It follows from conditions (2.1)-(2.5) that

$$\sum_{e\in M_i} w_i(e) \;=\; \sum_{u\in V} y^i_u \;+\; \sum_{B\in\Omega} z^i_B \left(\frac{|B|-1}{2}\right).$$

We know that for each edge e, $w_i(e) = w_{i-1}(e) + 1$. Thus $\sum_{e\in M_i} w_i(e) = \sum_{e\in M_i} w_{i-1}(e) + |M_i|$. Since every edge used in M_i belongs to the edge set F_i, we have $w_{i-1}(e) = y_u^{i-1} + y_v^{i-1} + \sum_{B:e\in E[B]} z_B^{i-1}$ for edge $e = (u,v) \in M_i$. Also, the vertices that are not matched in M_i are unmatched in M_{i-1} as well. Thus $y_u^{i-1} = 0$ for all vertices unmatched in M_i (by condition (2.3)). Hence we have

$$\sum_{u\in V} y_u^i \;+\; \sum_{B\in\Omega} z_B^i \left(\frac{|B|-1}{2}\right) \;=\; |M_i| + \sum_{u\in V} y_u^{i-1} \;+\; \sum_{e\in M_i} \sum_{B:e\in E[B]} z_B^{i-1}.$$

What is left to show is that $\sum_{e\in M_i} \sum_{B:e\in E[B]} z_B^{i-1} = \sum_{B\in\Omega} z_B^{i-1} \left(\frac{|B|-1}{2}\right)$. By rearranging the terms in the sum, we have $\sum_{e\in M_i} \sum_{B:e\in E[B]} z_B^{i-1} = \sum_{B\in\Omega} z_B^{i-1} \cdot |E[B] \cap M_i|$. Thus what is left to show is that for each $B \in \Omega$ with $z_B^{i-1} > 0$, exactly $(|B|-1)/2$ edges of $E[B]$ are present in M_i.

Consider any such B. Since $z_B^{i-1} > 0$, at the beginning of the i-th iteration of our algorithm, the odd set B is shrunk in the node set $\tilde{V}_i$. At the end of the i-th iteration, either $z_B^i > 0$ in which case $(|B|-1)/2$ edges of B are present in M_i (by condition (2.4)) or z_B^i becomes 0 in which case by the process of opening a blossom, $(|B|-1)/2$ edges of $E[B]$ are present in M_i. This completes the proof of part 3.

Part 4 follows from adding the equations of part 3 for all $i \in \{1,\ldots,W\}$. Since the right hand side consists of a cascading sum and $y_u^0 = 0$ for all u and $z_B^0 = 0$ for all B, this results in

$$\sum_{i=1}^{M} |M_i| \;=\; \sum_{u\in V} y_u^W + \sum_{B\in\Omega} z_B^W \left(\frac{|B|-1}{2}\right).$$

We also know that the above right side is the optimum value for the dual program in the W-th iteration. So this equals the value of the optimal primal solution, which is the maximum weight of a matching in G_W. Since $G_W = G$, the maximum weight of a matching in G equals $\sum_{i=1}^{W} |M_i|$. □

It may be tempting to try to generalize the following decomposition theorem of Kao et al. [102] to the context of general graphs.

Theorem 2.2.8 (from [102]). *Let G be bipartite. Let G' be G_i with the weight function w_i for some $i \in \{1,\ldots,W\}$. Let $\langle y_u^i \rangle_{u\in V}$ be any minimum weight cover in G_i and G'' be the subgraph of G with the edge set $\{e = (a,b) : w(e) - y_a^i - y_b^i > 0\}$ with weight of edge e given by $\tilde{w}(e) = w(e) - y_a^i - y_b^i$. Then the maximum weight of a matching in G is equal to the sum of the maximum weights of a matching in G' and in G''.*

Unfortunately, the above theorem with $\tilde{w}(e)$ updated to $w(e) - y_u^i - y_u^i - \sum_{B:e\in E[B]} z_B^i$ for $e = (u,v)$ need not hold in non-bipartite graphs. Consider the example in Figure 2.1.

The maximum weight of a matching in G is 5. Let $G' = G_1$, so the weight function is $w_1(e) = w(e) - 2$; thus we have $w_1(e) = 1$ for $e \in \{(a,b),(b,c),(c,a)\}$ while $w_1(e) \leq 0$ for $e \in \{(a,x),(b,y),(c,z)\}$. The maximum weight of a matching in G' is 1 and there is a unique optimal dual solution: $z_{\{a,b,c\}}^1 = 1$; all other z-values and y-values are 0. In G'', the weight of edge $e = (u,v)$ is $\tilde{w}(e) = w(e) - y_u^1 - y_v^1 - \sum_{B:e\in E[B]} z_B^1$ (see Figure 2.2). The maximum weight of a matching in G'' is 5. However the maximum weight of a matching in G is *not* $1 + 5 = 6$.

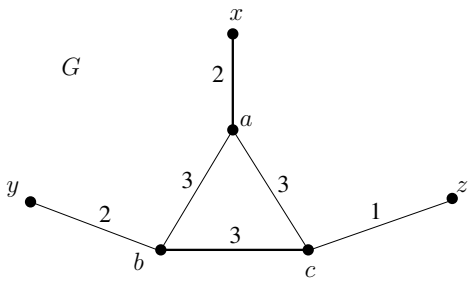

Figure 2.1: The numbers on edges denote their weights and the bold edges (x, a) and (b, c) are the ones in the maximum weight matching. The weight of a maximum weight matching in G is 5.

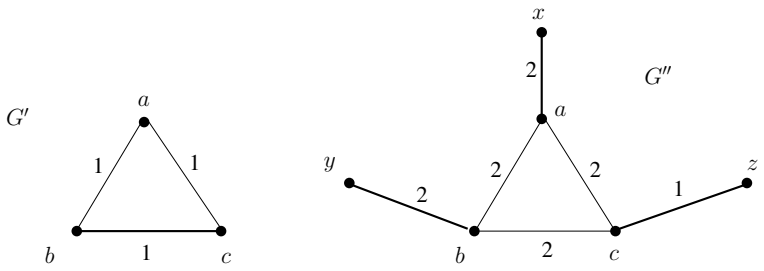

Figure 2.2: In the graph G', the edges with non-positive weight have not been shown above. The weight of a maximum weight matching in G' is 1 and that in G'' is 5.

2.3 Maximum Weight Bipartite Capacitated b-matching

The input here is a bipartite graph $G = (A \cup B, E)$ where as before, there is a weight function $w : E \to \{1, \ldots, W\}$. Associated with each vertex is a quota given by $b : A \cup B \to Z^+$ and the goal is to compute a *b-matching* of maximum weight. A b-matching is a subset $M \subseteq E$ such that for any vertex u, at most $b(u)$ edges incident on u are present in M. When $b(u) = 1$ for each vertex u, the resulting b-matching is the standard matching studied in the previous section.

In fact, here we consider an even more generalized concept called "capacitated b-matchings": several copies of an edge e can be present in M. That is, there is an edge capacity function $c : E \to Z^+$ and up to $c(e)$ copies of edge e are allowed in M. Recall that $E(v)$ is the set of edges incident on vertex v. For any vertex v, we assume that $b(v) \leq \sum_{e \in E(v)} c(e)$.

Associated with every capacitated b-matching M is an m-tuple $(M(e_1), \ldots, M(e_m))$ where $0 \leq M(e_i) \leq c(e_i)$ for every edge e_i. For simplicity, we refer to capacitated b-matchings as b-matchings here. When $w(e) = 1$ for all edges e, a maximum weight b-matching M maximizes $\sum_{e \in E} M(e)$, so we will call such a matching a *maximum cardinality* b-matching.

We will solve the maximum weight b-matching problem in bipartite graphs by reducing it to several instances of the maximum cardinality b-matching problem. We will use the following terminology here.

- An edge e is *saturated* in M if $M(e) = c(e)$, otherwise, e is *unsaturated*. Similarly, a vertex v is *saturated* if $\sum_{e \in E(v)} M(e) = b(v)$, otherwise v is *unsaturated*.

- An edge e with $M(e) > 0$ is a *positive* edge.

Definition 2.3.1. *An* alternating *path with respect to a b-matching M is a path $p = \langle e_0, e_1, \ldots, e_k \rangle$ in G such that $e_0, e_2, \ldots$ are unsaturated edges while $e_1, e_3, \ldots$ are positive edges. That is, unsaturated edges and positive edges alternate in p.*

An alternating path p of *odd* length with respect to a b-matching M such that both the endpoints of p are unsaturated in M is also called an "augmenting path" with respect to M. It is easy to see that a maximum cardinality b-matching admits no augmenting path with respect to it. We present below a generalization of the coarse version of Dulmage-Mendelsohn decomposition of bipartite graphs [40]. We first need the following definition.

Definition 2.3.2. *Let M be a maximum cardinality b-matching in $G = (A \cup B, E)$.*

- *Let $\mathcal{E}_A \subseteq A$ (similarly, $\mathcal{O}_B \subseteq B$) be the set of vertices reachable from an unsaturated vertex in A via an alternating path of even (resp., odd) length with respect to M.*
- *Let $\mathcal{E}_B \subseteq B$ (similarly, $\mathcal{O}_A \subseteq A$) be the set of vertices reachable from an unsaturated vertex in B via an alternating path of even (resp., odd) length with respect to M.*
- *Let $\mathcal{U}_A = A \setminus (\mathcal{E}_A \cup \mathcal{O}_A)$ and let $\mathcal{U}_B = B \setminus (\mathcal{E}_B \cup \mathcal{O}_B)$.*

Proposition 2.3.3. *Let M be a maximum cardinality b-matching in $G = (A \cup B, E)$ and $\mathcal{E}_A, \mathcal{E}_B, \mathcal{O}_A, \mathcal{O}_B, \mathcal{U}_A, \mathcal{U}_B$ be the sets of vertices as defined in Definition 2.3.2. Furthermore, let M' be an arbitrary maximum cardinality b-matching in G. Then the following holds.*

(1) the three sets $\mathcal{O}_A$, $\mathcal{E}_A$, and $\mathcal{U}_A$ are pairwise disjoint; so are $\mathcal{O}_B$, $\mathcal{E}_B$, and $\mathcal{U}_B$.

(2) all vertices in $\mathcal{O}_A \cup \mathcal{O}_B \cup \mathcal{U}_A \cup \mathcal{U}_B$ are saturated in M'.

(3) there is no positive edge in $(\mathcal{U}_A \cup \mathcal{O}_A) \times \mathcal{O}_B$ or in $\mathcal{O}_A \times \mathcal{U}_B$ in M'.

(4) every edge in $\mathcal{E}_A \times (\mathcal{E}_B \cup \mathcal{U}_B)$ is saturated in M'; so is every edge in $\mathcal{U}_A \times \mathcal{E}_B$ in M'.

Proof. To show (1), we first show that $\mathcal{O}_A$, $\mathcal{E}_A$, and $\mathcal{U}_A$ are disjoint from one another. It follows from the definition of $\mathcal{U}_A$ that $\mathcal{U}_A \cap (\mathcal{E}_A \cup \mathcal{O}_A) = \emptyset$. So what is left to show is that $\mathcal{E}_A \cap \mathcal{O}_A = \emptyset$. Suppose $v \in \mathcal{E}_A \cap \mathcal{O}_A$ – then gluing p_1 and p_2 till their first common vertex yields an augmenting path p with respect to M, where p_1 is the even length alternating path between some unsaturated vertex $a \in A$ and v (such a path p_1 exists because $v \in \mathcal{E}_A$) and p_2 is the odd length alternating path between some unsaturated vertex $b \in B$ and v (such a path p_2 exists because $v \in \mathcal{O}_A$). The augmenting path p contradicts that M is a maximum cardinality b-matching. A similar argument shows that $\mathcal{O}_B$, $\mathcal{E}_B$, and $\mathcal{U}_B$ are disjoint from one another. This finishes the proof of (1).

In the following, we first prove (2)-(4) assuming that $M' = M$; we will later remove this assumption.

It follows from the definition of $\mathcal{U}_A$ and $\mathcal{U}_B$ that every vertex u in $\mathcal{U}_A \cup \mathcal{U}_B$ has to be saturated in M (otherwise there is a length zero alternating path from an unsaturated vertex to u, contradicting

that $u \in \mathcal{U}_A \cup \mathcal{U}_B$). If a vertex $a \in \mathcal{O}_A$ is unsaturated, then there is an alternating path of odd length from some unsaturated vertex in B to the unsaturated vertex a, i.e., M admits an augmenting path, a contradiction. Similarly every vertex in $\mathcal{O}_B$ is saturated. This finishes the proof of (2) when $M' = M$.

Suppose there is an edge $e \in \mathcal{O}_A \times \mathcal{O}_B$ such that $M(e) > 0$. Let $e = (u, v)$. Then we can again show an augmenting path with respect to M between an unsaturated vertex $a \in A$ and an unsaturated vertex $b \in B$ using the paths p_1, p_2, and the edge (u, v), where p_1 is the (odd length) a-u alternating path and p_2 is the (odd length) b-v alternating path. Similarly, if $e = (u, v)$ in $\mathcal{U}_A \times \mathcal{O}_B$ satisfies $M(e) > 0$, then there is an alternating path from some unsaturated vertex in A to u; this contradicts the fact that $u \in \mathcal{U}_A$. Similarly, there is no positive edge in $\mathcal{O}_A \times \mathcal{U}_B$. This finishes the proof of (3) when $M' = M$.

Suppose there is an edge $e \in \mathcal{E}_A \times \mathcal{E}_B$ such that $M(e) < c(e)$. Let $e = (u, v)$. Then we can again show an augmenting path between some unsaturated vertex $a \in A$ and some unsaturated vertex $b \in B$ using p_1, p_2, and (u, v), where p_1 is the (even length) a-u alternating path and p_2 is the (even length) b-v alternating path. Similarly, if $e = (u, v)$ in $\mathcal{U}_A \times \mathcal{E}_B$ satisfies $M(e) < c(e)$, then there is an alternating path between some unsaturated $b \in B$ and u; this contradicts the fact that $u \in \mathcal{U}_A$. Similarly, there is no unsaturated edge in $\mathcal{E}_A \times \mathcal{U}_B$. This finishes the proof of (4) when $M' = M$.

We now remove the assumption that $M' = M$. Let M' be an arbitrary maximum cardinality b-matching in G. Let us define a bipartite directed flow f as follows: $f(e) = |M'(e) - M(e)|$. The direction of the edge $e = (u, v)$ under f is from u to v if $M'(e) > M(e)$ and from v to u if $M'(e) < M(e)$. It is clear that M' is the sum of M and the flow f. Let us summarize what we know about f using what we have proved so far.

(i) Under f, there is no edge going from $\mathcal{E}_A$ to $\mathcal{U}_B \cup \mathcal{E}_B$ and no edge from $\mathcal{U}_A$ to $\mathcal{E}_B$,

(ii) under f, there is no edge going from $\mathcal{O}_B$ to $\mathcal{U}_A \cup \mathcal{O}_A$, and no edge from $\mathcal{U}_B$ to $\mathcal{O}_A$,

(iii) for all vertices in $\mathcal{O}_A \cup \mathcal{U}_A \cup \mathcal{O}_B \cup \mathcal{U}_B$, the amounts of incoming flow and outgoing flow are equivalent.

It is well known, e.g., [4] that f can be decomposed into a set of cycle flows C_i and a set of path flows P_j by a greedy algorithm. By (i) and (ii), the cycle C_i consists entirely of vertices in $\mathcal{E}_A \cup \mathcal{O}_A$ or $\mathcal{E}_B \cup \mathcal{O}_B$. Furthermore, by (iii), the starting and ending vertices of P_j can only be in $\mathcal{E}_A \cup \mathcal{E}_B$. By (i) and (ii), P_j cannot start from $\mathcal{E}_A$ and end in $\mathcal{E}_B$. Moreover, we cannot have P_j start from $\mathcal{E}_B$ and end in $\mathcal{E}_A$, as it would imply an augmenting path in M' from $\mathcal{E}_A$ to $\mathcal{E}_B$. We can thus conclude that the cycle C_i and P_j consist entirely of vertices in $\mathcal{E}_A \cup \mathcal{O}_A$ or in $\mathcal{E}_B \cup \mathcal{O}_B$ and the proof follows.

□

Note that if a maximum cardinality b-matching M is given, the decomposition $A = \mathcal{O}_A \cup \mathcal{E}_A \cup \mathcal{U}_A$ and $B = \mathcal{O}_B \cup \mathcal{E}_B \cup \mathcal{U}_B$ with respect to M can be determined in $O(m+n)$ time easily. We are now ready to describe our algorithm to compute a maximum weight b-matching in $G = (A \cup B, E)$. The linear program corresponding to the maximum weight b-matching problem and the dual LP are given below.

$$\max \sum_{e \in E} w(e) x_e$$

$$\sum_{e \in E(v)} x_e \leq b(v) \qquad \forall v \in A \cup B.$$

$$0 \leq x_e \leq c(e) \qquad \forall e \in E.$$

$$\min \sum_{v \in V} b_v y_v + \sum_{e \in E} c(e) z_e$$

$$y_a + y_b + z_e \geq w(e) \qquad \forall e = (a,b) \in E.$$

$$y_v \geq 0 \qquad \forall v \in A \cup B.$$

$$z_e \geq 0 \qquad \forall e \in E.$$

Our algorithm is similar in spirit to the algorithm from Section 2.2.1 but it also has some subtle differences from that one. The algorithm here also runs for W iterations, where $W = \max_{e \in E} w(e)$. In the first iteration, we consider only edges of weight W: this is the graph H_1. We compute a maximum cardinality b-matching M_1 here and assign vertex potentials y_u^1; however not all edges get covered by these vertex potentials, i.e., there exist edges $e = (a,b)$ in H_1 such that $y_a^1 = y_b^1 = 0$. More precisely, all the uncovered edges are in $\mathcal{E}_A^1 \times (\mathcal{E}_B^1 \cup \mathcal{U}_B^1)$; we also know from Proposition 2.3.3(4) that each edge in $\mathcal{E}_A^1 \times (\mathcal{E}_B^1 \cup \mathcal{U}_B^1)$ is *saturated* in M_1.

In fact, this will be an important invariant that we will maintain: every uncovered or "not fully paid for" edge will be saturated. In general, in iteration i we have an *unsatisfied set* $\Psi_i \subseteq E$, which consists of those edges $e = (a,b)$ such that $y_a^{i-1} + y_b^{i-1} < w_{i-1}(e)$ and it will be the case that e is saturated by the previous maximum cardinality b-matching M_{i-1}.

In the i-th iteration, the algorithm works with the unweighted graph $H_i = (A \cup B, F_i)$, where $F_i = \{e = (a,b) \in E : w_i(e) - y_a^{i-1} - y_b^{i-1} = 1\}$ and $w_i(e) = w(e) - (W - i)$. Furthermore, the quotas of the vertices are updated according to the unsatisfied set Ψ_{i-1}, namely, $b_i(v) = b(v) - \sum_{e \in \Psi_{i-1} \cap E(v)} c(e)$, where $E(v)$ is the set of edges incident on vertex v. Our task is to obtain a maximum cardinality b-matching in H_i, where vertex quotas are described by the function b_i; we obtain such a matching M_i' by augmenting $M_{i-1} \setminus \Psi_{i-1}$ in the graph H_i. The matching M_i will be M_i' along with all edges in Ψ_{i-1} (these edges are all saturated). We will show that the final matching M_W is a maximum weight b-matching in G. We present our algorithm below.

1. *Initialization*: Set $M_0 = \emptyset$, $\Psi_0 = \emptyset$, and $y_v^0 = 0 \ \forall v \in V$.
2. For $i = 1$ to W do
   (a) For each $e \in E$: let $w_i(e) = w(e) - (W - i)$.
   (b) Construct $H_i = (A \cup B, F_i)$, where $F_i = \{e = (a,b) : w_i(e) - y_a^{i-1} - y_b^{i-1} = 1\}$.
   (c) For each $v \in A \cup B$ do: set $b_i(v) = b(v) - \sum_{e \in \Psi_{i-1} \cap E(v)} c(e)$.
   (d) Find a maximum cardinality b-matching M_i' in H_i (where the quotas of vertices are given by b_i) by augmenting $M_{i-1} \setminus \Psi_{i-1}$ in H_i.
   (e) Using M_i', partition $A = \mathcal{O}_A^i \cup \mathcal{E}_A^i \cup \mathcal{U}_A^i$ and $B = \mathcal{O}_B^i \cup \mathcal{E}_B^i \cup \mathcal{U}_B^i$.
      - for each $v \in \mathcal{O}_A^i \cup \mathcal{O}_B^i \cup \mathcal{U}_A^i$ do: set $y_v^i = y_v^{i-1} + 1$.
      - for each $v \in \mathcal{E}_A^i \cup \mathcal{E}_B^i \cup \mathcal{U}_B^i$ do: set $y_v^i = y_v^{i-1}$.
   (f) Let M_i be $M_i' \cup \Psi_{i-1}$, where $M_i(e) = c(e)$ for each $e \in \Psi_{i-1}$.

(g) Let $S_i = \{e = (a,b) \in F_i$ where $a \in \mathcal{E}_A^i$ and $b \in \mathcal{E}_B^i \cup \mathcal{U}_B^i\}$;
let $C_i = \{e = (a,b) \in \Psi_{i-1}$ such that $w_i(e) - y_a^i - y_b^i = 0\}$.

(h) $\Psi_i = (\Psi_{i-1} \setminus C_i) \cup S_i$.

3. Return M_W.

It follows from the definition of F_i and from Step 2(e) (where vertex potentials are updated) that those edges of F_i with at least one endpoint in $\mathcal{O}_A^i \cup \mathcal{O}_B^i \cup \mathcal{U}_A^i$ are covered or paid for while those with both endpoints in $\mathcal{E}_A^i \times (\mathcal{E}_B^i \cup \mathcal{U}_B^i)$ are uncovered or not fully paid for. The edges of F_i in $\mathcal{E}_A^i \times (\mathcal{E}_B^i \cup \mathcal{U}_B^i)$ form the set S_i and these are added to the set Ψ_i in Step 2(h).

Regarding the edges that are already in the set Ψ_{i-1}, it could be the case that some of them get covered now (due to Step 2(e) where vertex potentials are updated) – for this to happen, it is necessary that both the endpoints of such an edge are in $\mathcal{O}_A^i \cup \mathcal{O}_B^i \cup \mathcal{U}_A^i$. These newly covered edges form the set C_i and these are no longer present in the unsatisfied set Ψ_i.

Lemma 2.3.4. *For each $1 \le i \le W$, the above algorithm maintains conditions (2.9)-(2.13) listed below.*

$$y_a^i + y_b^i \ge w_i(e) \qquad \forall \text{ edges } e = (a,b) \in E \setminus \Psi_i \tag{2.9}$$

$$y_a^i + y_b^i < w_i(e) \text{ and } M_i(e) = c(e) \qquad \forall\, e \in \Psi_i \tag{2.10}$$

$$y_u^i > 0 \Rightarrow u \text{ is saturated in } M_i \qquad \forall\, u \in A \cup B \tag{2.11}$$

$$y_a^i + y_b^i = w_i(e) \qquad \forall \text{edges } e = (a,b) \in F_i \setminus S_i \text{ such that } M_i'(e) \succ \tag{2.12}$$

$$y_u^i \ge 0 \qquad \forall\, u \in A \cup B. \tag{2.13}$$

Proof. We show by induction on i that conditions (2.9)-(2.13) hold for all $1 \le i \le W$. The base case corresponds to $i = 1$. The set $\Psi_1 = \mathcal{E}_A^1 \times (\mathcal{E}_B^1 \cup \mathcal{U}_B^1)$ and it is easy to see that conditions (2.9)-(2.13) for $i = 1$ follow from Proposition 2.3.3 and from how y_u^1-values are set in Step 2(e).

We now assume that conditions (2.9)-(2.13) hold when $i = k-1$ and show that the conditions corresponding to $i = k$ hold as well. To show condition (2.9) for $i = k$, assume that $e \in E \setminus \Psi_k$. If $e \in \Psi_{k-1} \setminus \Psi_k$, then e has to be in C_k, in other words, $w_k(e) = y_a^k + y_b^k$. So we can henceforth assume that $e \notin \Psi_{k-1}$. So $y_a^{k-1} + y_b^{k-1} \ge w_{k-1}(e)$ by induction hypothesis. We consider two cases here: (i) $e \in F_k$ and (ii) $e \notin F_k$.

Since F_k consists of edges $e = (a,b)$ such that $w_k(e) - y_a^{k-1} - y_b^{k-1} = 1$ and $w_k(e) = w_{k-1}(e) + 1$, it is easy to see that F_k is exactly the set of those edges $e = (a,b)$ such that $y_a^{k-1} + y_b^{k-1} = w_{k-1}(e)$. If $e \notin F_k$, then we have $y_a^{k-1} + y_b^{k-1} > w_{k-1}(e)$, that is, $y_a^{k-1} + y_b^{k-1} \ge w_{k-1}(e) + 1$. So we have

$$y_a^k + y_b^k \ge y_a^{k-1} + y_b^{k-1} \ge w_{k-1}(e) + 1 = w_k(e).$$

So suppose $e \in F_k$. In this case, at least one of $\{a, b\}$ has to be in $\mathcal{O}_A^k \cup \mathcal{O}_B^k \cup \mathcal{U}_A^k$, otherwise $e \in S_k$ and thus $e \in \Psi_k$. Since a or b (or both) is in $\mathcal{O}_A^k \cup \mathcal{O}_B^k \cup \mathcal{U}_A^k$, we have

$$y_a^k + y_b^k \;\ge\; y_a^{k-1} + y_b^{k-1} + 1 \;=\; w_{k-1}(e) + 1 \;=\; w_k(e).$$

Thus condition (2.9) holds for $i = k$.

We now show condition (2.10) for $i = k$. The set $\Psi_k = (\Psi_{k-1} \setminus C_k) \cup S_k$. It is easy to see that every $e = (a, b) \in S_k$ satisfies $y_a^k + y_b^k < w_k(e)$ and $M_k(e) = c(e)$: this is because every such edge belongs to $F_k \cap (\mathcal{E}_A^k \times (\mathcal{E}_B^k \cup \mathcal{U}_B^k))$, thus $w_k(e) = w_{k-1}(e) + 1 = y_a^{k-1} + y_b^{k-1} + 1 = y_a^k + y_b^k + 1$ and by Proposition 2.3.3.4, $M_k(e) = c(e)$.

We now consider the case when $e \in \Psi_{k-1} \setminus C_k$. By induction hypothesis, every edge $e = (a, b)$ in Ψ_{k-1} satisfies $y_a^{k-1} + y_b^{k-1} < w_{k-1}(e)$ and $M_{k-1}(e) = c(e)$. In the k-th iteration we have

$$y_a^k + y_b^k \;\leq\; y_a^{k-1} + y_b^{k-1} + 2 \;\leq\; w_{k-1}(e) + 1 \;=\; w_k(e).$$

We are given that $e \notin C_k$, so $y_a^k + y_b^k \neq w_k(e)$. Thus $y_a^k + y_b^k < w_k(e)$. Since e is in Ψ_{k-1}, it follows that $M_k(e) = c(e)$.

We now show condition (2.11) for $i = k$. We are given that $y_u^k > 0$. If $y_u^{k-1} > 0$, then it follows from induction hypothesis that u is saturated in M_{k-1}; since M'_k is obtained by augmenting $M_{k-1} \setminus \Psi_{k-1}$ and $M_k = M'_k \cup \Psi_{k-1}$, it follows that $\sum_{e \in E(u)} M_{k-1}(u) = \sum_{e \in E(u)} M_k(u)$, thus u is saturated in M_k. If $y_u^{k-1} = 0$, then it must be the case that $y_u^k \in \mathcal{O}_A^k \cup \mathcal{O}_B^k \cup \mathcal{U}_A$ and it follows from Proposition 2.3.3.2 that u is saturated in M_k.

We now show condition (2.12) for $i = k$. Let $e = (a, b)$ be an edge such that $M'_k(e) > 0$. We know that M'_k is a maximum cardinality b-matching in H_k: since the edge set of H_k is F_k, this implies that $y_a^{k-1} + y_b^{k-1} = w_{k-1}(e)$. We also know that $e \notin S_k$. It follows from Proposition 2.3.3.3 that $e \in (\mathcal{O}_A^k \times \mathcal{E}_B^k) \cup (\mathcal{E}_A^k \times \mathcal{O}_B^k) \cup (\mathcal{U}_A^k \times \mathcal{E}_B^k)$. So we have $y_a^k + y_b^k = y_a^{k-1} + y_b^{k-1} + 1$. So $y_a^k + y_b^k = y_a^{k-1} + y_b^{k-1} + 1 = w_{k-1}(e) + 1 = w_k(e)$.

It is easy to see that condition (2.13) holds. Initially $y_u^0 = 0$ for all vertices u and Step 2(e) (where certain y_u^i values increase) is the only step where the y_u^i values get updated. Thus $y_u^k \geq 0$ for all u.

□

Theorem 2.3.5. *The matching M_W is a maximum weight b-matching in G.*

Proof. We prove the optimality of M_W by showing a dual feasible solution $\langle y_u^*, z_e^* \rangle_{(u \in A \cup B,\, e \in E)}$ such that this dual solution and M_W satisfy complementary slackness conditions. Let $y_u^* = y_u^W$ for all $u \in A \cup B$ and define $z_e^* = 0$ if $e \notin \Psi_W$, else $z_e^* = w(e) - y_a^* - y_b^*$ where $e = (a, b)$.

Observe that $w_W(e) = w(e)$ for all edges e. The dual feasibility of $\langle y_u^*, z_e^* \rangle_{(u \in A \cup B,\, e \in E)}$ follows from conditions (2.9), (2.10), (2.13), and the definition of z_e^* values. Primal complementary slackness follows from conditions (2.10) and (2.12) along with the definition of z_e^* values for $e \in \Psi_W$. Dual complementary slackness follow from conditions (2.10) and (2.11) along with the observation that if $z_e^* > 0$ then $e \in \Psi_W$. This proves that M_W is primal optimal and $\langle y_u^*, z_e^* \rangle_{(u \in A \cup B,\, e \in E)}$ is dual optimal.

□

Thus the maximum weight bipartite capacitated b-matching problem can be decomposed into W unweighted versions of the same problem, where $W = \max_{e \in E} w(e)$. The following computational result is immediate.

Theorem 2.3.6. *The maximum weight capacitated b-matching problem in $G = (A \cup B, E)$ can be solved in*

1. $O(Wnm)$ *time using Orlin's maximum flow algorithm [137], or*
2. $O(W\sqrt{\beta}m)$ *time, using Gabow's algorithm [61], where* $\beta = \sum_{v \in A \cup B} b(e)$, *in the case of simple b-matching (where* $c \equiv 1$*), or*
3. $O(W(n_1 m + n_1^3))$ *time, using Ahuja et al.'s algorithm [5], where* $n_1 = \min\{|A|, |B|\}$, *or*
4. $O(W(n_1 m + n_1^2\sqrt{m}))$ *time, using Ahuja et al.'s algorithm [5], or*
5. $O(W(n_1 m + n_1^2\sqrt{\log C}))$ *time, using Ahuja et al.'s algorithm [5], or*
6. $O(Wn_1 m \log(2 + \frac{n_1^2}{m}))$ *time, using Ahuja et al.'s algorithm [5].*

2.4 Conclusions and Open Problems

We considered the maximum weight matching problem in $G = (V, E)$ with integral edge weights. We solved this problem via the maximum cardinality matching algorithm – the running time of our algorithm is W times the running time of a maximum cardinality matching algorithm, where W is the largest edge weight. This running time is as good as the current fastest algorithms for the maximum weight matching problem. Our algorithm also computed an optimal dual solution that is integral, thereby showing an integral certificate to the optimality of the computed matching.

We then extended this approach to the maximum weight capacitated b-matching problem in bipartite graphs, where edge weights are in $\{1, 2, \ldots, W\}$. We showed that this problem can also be decomposed into W unweighted and capacitated versions of the same problem. An open problem is to extend this approach to the maximum weight b-matching problem in general graphs.

Chapter 3

Exact and Approximation Algorithms for Weighted Matroid Intersection

This paper first appeared in SODA 2016 and its full version will appear in Mathematical Programming. It is joint-work with Naonori Kakimura and Naoyuki Kamiyama.

Abstract In this paper, we propose new exact and approximation algorithms for the weighted matroid intersection problem. Our exact algorithm is faster than previous algorithms when the largest weight is relatively small. Our approximation algorithm delivers a $(1-\epsilon)$-approximate solution with a running time significantly faster than most known exact algorithms.

The core of our algorithms is a decomposition technique: we decompose an instance of the weighted matroid intersection problem into a set of instances of the unweighted matroid intersection problem. The computational advantage of this approach is that we can make use of fast unweighted matroid intersection algorithms as a black box for designing algorithms. More precisely, we show that we can solve the weighted matroid intersection problem via solving W instances of the unweighted matroid intersection problem, where W is the largest given weight, assuming that all given weights are integral. Furthermore, we can find a $(1-\epsilon)$-approximate solution via solving $O(\epsilon^{-1}\log r)$ instances of the unweighted matroid intersection problem, where r is the smaller rank of the two given matroids. Our algorithms make use of the weight-splitting approach of Frank [54] and the geometric scaling scheme of Duan and Pettie [35].

Our algorithms are simple and flexible: they can be adapted to special cases of the weighted matroid intersection problem, using specialized unweighted matroid intersection algorithms. In addition, we give a further application of our decomposition technique: we solve efficiently the rank-maximal matroid intersection problem, a problem motivated by matching problems under preferences.

3.1 Introduction

In the classical *weighted matroid intersection problem*, we are given two matroids $\mathbf{M}_1 = (S, \mathcal{I}_1), \mathbf{M}_2 = (S, \mathcal{I}_2)$ and a weight function $w\colon S \to \mathbb{Z}_{\geq 0}$, where $\mathbb{Z}_{\geq 0}$ is the set of non-negative integers. Then, the goal is to find a maximum-weight common independent set I of $\mathbf{M}_1$ and $\mathbf{M}_2$, i.e., $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ with $\sum_{e \in I} w(e)$ being maximized. This problem was introduced by Edmonds [43, 45] and solved by Edmonds [43, 45] and others [6, 94, 117, 118] in 1970s. This problem is a common generalisation of various combinatorial optimization problems such as bipartite matchings, packing spanning trees, and arborescences in a directed graph. In addition, it has many applications, e.g., in electric circuit theory [133, 146], rigidity theory [146], and network coding [34]. The fact that two matroids capture the underlying common structures behind a large class of polynomially solvable problems has been impressive and motivated substantial follow-up research (see, e.g., [57, 158]). Techniques and theorems developed surrounding this problem have become canon in contemporary combinatorial optimization literature.

Since 1970s, quite a few algorithms have been proposed for matroid intersection problems, e.g., [14, 31, 54, 58, 160], with better running time and/or simpler proofs. See Table 3.1 for a summary. Throughout the paper, n is the size of the ground set S, r is the smallest rank of the two given matroids, and W is the largest given weight. The oracle to check the independence of a given set has the running time of τ.

Table 3.1: Matroid intersection algorithms for general matroids. See also [43, 45, 142]. The complexity is measured only by the number of independence oracle calls. In case the original algorithms (Fujishige–Zhang and Gabow–Xu) use (co-)circuit oracles, each call of such oracles is replaced by n independence calls in the table.

Algorithm	Weight	Time complexity
Aigner–Dowling [6]	Unweighted	$O(\tau n r^2)$
Cunningham [31], Gabow–Xu [69]	Unweighted	$O(\tau n r^{1.5})$
Lawler [117, 118], Iri–Tomizawa [94]	Weighted	$O(\tau n r^2)$
Frank [54]	Weighted	$O(\tau n^2 r)$
Brezovec–Cornuéjols–Glover [14]	Weighted	$O(\tau n r^2)$
Fujishige–Zhang [58], Shigeno–Iwata [160], Gabow–Xu [69]	Weighted	$O(\tau n^2 \sqrt{r} \log rW)$
Lee-Sidford-Wong [123]	Weighted	$O(\tau n^2 \log nW)$
Chekuri–Quanrud [19] ($(1-\epsilon)$-approximation)	Weighted	$O(\tau n r \epsilon^{-2} \log^2 \epsilon^{-1})$
This paper	Weighted	$O(\tau W n r^{1.5})$
This paper ($(1-\epsilon)$-approximation)	Weighted	$O(\tau \epsilon^{-1} n r^{1.5} \log r)$

3.1.1 Our Contribution

We propose both exact and approximation algorithms for the weighted matroid intersection problem. Our exact algorithm is faster than known algorithms when the largest given weight W is relatively small. Our approximation algorithm delivers a $(1-\epsilon)$-approximate solution for every

Table 3.2: Matroid intersection algorithms for graphic matroids.

Algorithm	Weight	Time complexity
Gabow–Stallman [67]	Unweighted	$O(\sqrt{r}n)$ if $n = \Omega(r^{3/2}\log r)$
	Unweighted	$O(rn^{2/3}\log^{1/3} r)$ if $n = \Omega(r\log r)$ & $n = O(r^{3/2}\log r)$
	Unweighted	$O(r^{4/3}n^{1/3}\log^{2/3} r)$ if $n = O(r\log r)$
Gabow–Xu [68]	Unweighted	$O(\sqrt{r}n\log r)$
Gabow–Xu [68]	Weighted	$O(\sqrt{r}n\log^2 r\log(rW))$
This paper	Weighted	$O(W\sqrt{r}n\log r)$
$(1-\epsilon)$ **approximation**	Weighted	$O(\epsilon^{-1}\sqrt{r}n\log^2 r)$

Table 3.3: Linear matroid intersection algorithms. Here the $\tilde{O}$ notation hides a polynomial of $\log n$ in the complexity.

Algorithm	Weight	Time complexity
Cunningham [31]	Unweighted	$O(nr^2\log r)$
Gabow–Xu [69]	Unweighted	$O(nr^{\frac{5-\omega}{4-\omega}}\log r)$
Harvey [84]	Unweighted	$O(nr^{\omega-1})$
Cheung, et al. [21]	Unweighted	$O(nr\log r_* + nr_*^{\omega-1})$
Gabow–Xu [69]	Weighted	$O(nr^{\frac{7-\omega}{5-\omega}}\log^{\frac{\omega-1}{5-\omega}} r\log nW)$
Harvey [83]	Weighted	$\tilde{O}(W^{1+\epsilon}nr^{\omega-1})$
This paper	Weighted	$O(nr\log r_* + Wnr_*^{\omega-1})$
$(1-\epsilon$ approximation)	Weighted	$O(nr\log r_* + \epsilon^{-1}nr_*^{\omega-1}\log r_*)$

fixed $\epsilon > 0$ in times substantially faster than known exact algorithms in most cases. Our algorithms and their analysis are surprisingly simple. Moreover, these algorithms can be specialized for particular classes of matroids.

The core of our algorithms is a decomposition technique. We show that a given instance of the weighted matroid intersection problem can be decomposed into a set of unweighted versions of the same problem. To be precise, we can solve the weighted problem exactly by solving W unweighted ones. Furthermore, we can solve the weighted problem $(1-\epsilon)$-approximately by solving $O(\epsilon^{-1}\log r)$ unweighted ones.

Our decomposition technique not only establishes a hitherto unclear connection between the weighted and unweighted problems, but also leads to computational advantages: the known unweighted matroid intersection algorithms are significantly faster than their weighted counterparts. Thus, we can make use of the former to design faster algorithms. It may be expected that in the future, there will be even more efficient unweighted matroid intersection algorithms, and that would imply our algorithms will become faster as well.

We summarize the complexity of our exact algorithms below. For comparison of our algorithms with previous results, see Tables 3.1–3.3.

General matroids. Given two general matroids, using the unweighted matroid intersection al-

gorithm of Cunningham [31], we can solve the weighted matroid intersection problem in $O(\tau W n r^{1.5})$ time. This algorithm is faster than all known algorithms when $W = o(\min\{\sqrt{r}, \frac{n \log r}{r}\})$ and $r = O(\sqrt{n})$. A slightly different analysis shows that the same algorithm has the complexity[1] of $O(\tau(\sum_{e \in S} w(e)) r^{1.5})$.

Graphic matroids. Given two graphic matroids, using the unweighted graphic matroid intersection algorithm proposed by Gabow and Xu [68], we can solve the weighted matroid intersection problem in $O(W\sqrt{r}n \log r)$ time. This is faster than the current fastest algorithm when $W = o(\log^2 r)$. If the graph is relatively dense, that is, $n = \Omega(r^{1.5} \log r)$, then we can use the algorithm of Gabow and Stallman [67] to solve the problem in $O(W\sqrt{r}n)$ time.

Linear matroids. Given two linear matroids (in the form of two r-by-n matrices), using the unweighted linear matroid intersection algorithm of [21], we can solve the weighted matroid intersection problem in $O(nr \log r_* + W n r_*^{\omega-1})$ time, where ω is the exponent of the matrix multiplication time and $r_* \leq r$ is the maximum size of a common independent set. This is faster than all known algorithms when $W = o(r^{\frac{\omega^2-7\omega+12}{5-\omega}})$ (if $\omega \approx 2.37$ [27, 72, 171], it is when $W = o(r^{0.41})$).

In the graphic matroid intersection problem, two graphs $G_1 = (V_1, E)$ and $G_2 = (V_2, E)$ with the same edge set E are given. An intersection of the two matroids means a subset of edges $E' \subseteq E$ so that E' induces a forest in both G_1 and G_2. In the linear matroid intersection problem, two r-by-n matrices M_1 and M_2 are given. An intersection of the two matroids means a subset of columns so that they are linearly independent in both M_1 and M_2. The graphic and linear matroid intersection problems arise in various branches in engineering. For example, the intersection of graphic matroids has applications in determining the order of complexity of an electrical network [93] and the unique solvability of open networks [145]; the intersection of linear matroids has applications in the analysis of systems of linear differential equations [133, 134].

A recent trend in research is to design fast approximation algorithms for fundamental optimization problems, even if they are in **P**. Examples include maximum weight matching [35], shortest paths [166], and maximum flow [24, 108, 124, 159]. Using the algorithms of [21, 31, 68], our decomposition technique delivers a $(1-\epsilon)$-approximate solution in

1. $O(\tau \epsilon^{-1} n r^{1.5} \log r)$ time with two general matroids,
2. $O(\epsilon^{-1}\sqrt{r}n \log^2 r)$ time with two graphic matroids,
3. $O(nr \log r_* + \epsilon^{-1} n r_*^{\omega-1} \log r_*)$ time with two linear matroids.

Our approximation algorithms are significantly faster than most exact algorithms. Prior to our results, there is only a simple greedy 1/2-approximation algorithm [100, 111] dated in 1970s. It should be noted that, by scaling weights to small integers, i.e., rounding W to $O(\frac{r}{\epsilon})$ (cf. Lemma 3.4.1), exact algorithms deliver a $(1-\epsilon)$-approximate solution (this is used in [22] for the linear matroid parity). Ours improve on such simple scaling significantly. We note that for general matroids, very recently, Chekuri and Quanrud [19] improved on our results: they obtain a $(1-\epsilon)$-approximate solution in $O(\tau n r \epsilon^{-2} \log^2 \epsilon^{-1})$ time.

[1] This complexity is superior to the previous one only when the given weights are very "unbalanced."

3.1.2 Our Technique

The idea of reducing a weighted optimization problem into unweighted ones has been successfully applied in the context of maximum-weight matching in bipartite graphs [102] and in general graphs [90, 144]. Roughly speaking, these matching algorithms proceed iteratively: in each round, in a subgraph, a maximum-cardinality matching and its optimal dual are computed; the latter is then used to update the edge weights to construct the next subgraph. The optimality of the final solution is shown via the complementary slackness condition.

The difficulty of extending such approaches for matching to the weighted matroid intersection problem lies in the fact that, in the latter problem, the dual variables are harder to reason with and to control. Instead we make use of Frank's weight-splitting approach [54, 55]. Originally Edmonds [45] gave a linear system describing the common independent sets and proved such a system to be *totally dual integral*. This fact suggests that there exists a weight-splitting $w = w_1 + w_2$ so that there exists a common independent set F so that F is of maximum weight for w_1 in M_1 and for w_2 in M_2 and in fact Frank designed an algorithm [54] using this fact. The technical innovation of Frank lies in avoiding the dual variables entirely when obtaining the weight splitting.

Our main insight is that the split weights w_1 and w_2 can also be used to re-define two new matroids for subsequent operations. This is analogous to using the dual optimal solution to update the edge weights in the maximum-weight matching [90, 102, 144].

Our approximation algorithms use the above basic ideas and a scaling technique of [35] for approximating maximum-weight matching. In particular, as in [35], the amount of adjustments done to the weights w_1 and w_2 decreases geometrically in each phase.

3.1.3 Application: rank-maximal matroid intersection

We consider a variation of the weighted matroid intersection problem, called the *rank-maximal matroid intersection problem*. Suppose that instead of a weight function w, a rank function $\lambda\colon S \to \{1, 2, \ldots, R\}$ is given, where R is some positive integer. The goal is to find a common independent set so that it has the maximum number of elements e with rank 1, and subject to that, it has the maximum number of elements e with rank 2 and so on. The problem is a generalization of the rank-maximal matching problem, introduced by [99] in the context of matching problems with preference lists.

As done in [99], we can reduce this problem to the weighted matroid intersection problem by assigning huge weights, say $\Omega(n^{R-i})$, to elements of rank i. However, such an approach would be inefficient in time and space. We show how to modify our exact algorithm to decompose the problem into R unweighted matroid intersection problems. In particular, we solve the rank-maximal matroid intersection problem using $O(Rnr^{1.5})$ independence oracle calls. Moreover, if the given two matroids are graphic or linear, the running times are reduced to $O(R\sqrt{r}n \log r)$ and $O(Rnr^{\omega-1})$, respectively.

3.1.4 Outline

The rest of the paper is organized as follows. In Section 3.2, we give definitions and basic properties of matroids. Our exact and approximation algorithms are presented in Sections 3.3 and 3.4, respectively. Implementation details about finding a maximum-cardinality common independent set are described in Section 3.5. The result of the rank-maximal matroid intersection problem is in Section 4.2.3. The relation of our results with previous work is discussed in Section 3.7.

3.2 Preliminaries

3.2.1 Matroids

A *matroid* is a pair $\mathbf{M} = (S, \mathcal{I})$ of a finite set S and a family $\mathcal{I}$ of subsets of S satisfying the following three conditions.

(I0) $\mathcal{I} \neq \emptyset$.

(I1) If $I \subseteq J$ and $J \in \mathcal{I}$, then $I \in \mathcal{I}$.

(I2) If $I, J \in \mathcal{I}$ and $|I| < |J|$, then there is $e \in J \setminus I$ such that $I + e \in \mathcal{I}$.[2]

A set in $\mathcal{I}$ is said to be *independent*, and a maximal independent set is called a *base*. In addition, a minimal non-independent subset C of S is called a *circuit*. A circuit of size one is a *loop*. Throughout the article, we assume that the given matroids have no loops.

Let $\mathbf{M} = (S, \mathcal{I})$ be a matroid and X a subset of S. The *restriction* of $\mathbf{M}$ to X is defined by $\mathbf{M}|X = (X, \mathcal{I}|X)$ with $\mathcal{I}|X = \{I \in \mathcal{I} \mid I \subseteq X\}$. The *contraction* of $\mathbf{M}$ with respect to X is defined as $\mathbf{M}/X = (S \setminus X, \mathcal{I}/X)$ with $\mathcal{I}/X = \{I \subseteq S \setminus X \mid I \cup B \in \mathcal{I}$ for some base B of $\mathbf{M}|X\}$. The *direct sum* of matroids $\mathbf{M}_1 = (S_1, \mathcal{I}_1)$ and $\mathbf{M}_2 = (S_2, \mathcal{I}_2)$, denoted by $\mathbf{M}_1 \oplus \mathbf{M}_2$, is defined to be $(S_1 \cup S_2, \mathcal{I}')$, where $\mathcal{I}' = \{I_1 \cup I_2 \mid I_1 \in \mathcal{I}_1, I_2 \in \mathcal{I}_2\}$.

Given a matroid $\mathbf{M} = (S, \mathcal{I})$ and a weight function $w\colon S \to \mathbb{Z}_{\geq 0}$, a set $I \in \mathcal{I}$ is said to be *w-maximum*, if its weight $\sum_{e \in I} w(e)$ is maximum among all independent sets in $\mathcal{I}$. A base is called a *w-maximum base*, if its weight is maximum among all bases. Using the family of w-maximum bases of $\mathbf{M} = (S, \mathcal{I})$, one can define a new matroid $\mathbf{M}^w = (S, \mathcal{I}^w)$, where

$$\mathcal{I}^w = \{I \mid I \subseteq B \text{ for some } w\text{-maximum base } B \text{ of } \mathbf{M}\}.$$

It is well known that $\mathbf{M}^w$ is a matroid (see e.g., [44]).

The following lemma states some important properties of such a derived matroid $\mathbf{M}^w$. As these properties are well-known (e.g. see [170]), we omit the proof.

Lemma 3.2.1. *Assume that we are given a matroid $\mathbf{M} = (S, \mathcal{I})$ and a weight function $w\colon S \to \{0, 1, \ldots, W\}$. We define $Z(t) = \{e \in S \mid w(e) \geq t\}$ for each integer $t \geq 0$.*

(i) $\mathbf{M}^w = \bigoplus_{t=0}^{W} (\mathbf{M}|Z(t))/Z(t+1)$.

(ii) *A set $I \in \mathcal{I}$ is w-maximum if and only if $I \cap Z(t)$ is a base of $\mathbf{M}|Z(t)$ for every $t = 1, 2, \ldots, W$.*

[2] We use the shorthand $I + e$ and $I - e$ to stand for $I \cup \{e\}$ and $I \setminus \{e\}$, respectively.

(iii) *Suppose that a set $I \in \mathcal{I}$ satisfies the condition that $I \cap Z(t)$ is a base in $\mathbf{M}|Z(t)$ for every integer t with $(\min_{e \in S} w(e)) + 1 \leq t \leq W$, and $I + e_0$, where $e_0 \in S \backslash I$, contains a circuit C' of $\mathbf{M}^w$. Then, every element in C' has weight equal to $w(e_0)$. Furthermore, there exists a circuit $C \supseteq C'$ in $I + e$ with respect to $\mathbf{M}$, and each element in $C \setminus C'$ has weight greater than $w(e_0)$.*

3.2.2 Matroid Intersection

Suppose that we are given a pair of matroids $\mathbf{M}_1 = (S, \mathcal{I}_1)$ and $\mathbf{M}_2 = (S, \mathcal{I}_2)$ on the same ground set S. A subset I of S is called a *common independent set* if I is in $\mathcal{I}_1 \cap \mathcal{I}_2$. The goal of the *matroid intersection problem* is to find a maximum-cardinality common independent set. Given $\mathbf{M}_1$ and $\mathbf{M}_2$, the *auxiliary graph* is a directed graph $G_{\mathbf{M}_1,\mathbf{M}_2}(I) = (S, E_1 \cup E_2)$, where

$$E_1 = \{ef \mid I + e \notin \mathcal{I}_1,\ I + e - f \in \mathcal{I}_1\},$$
$$E_2 = \{fe \mid I + e \notin \mathcal{I}_2,\ I + e - f \in \mathcal{I}_2\}.$$

In the auxiliary graph $G_{\mathbf{M}_1,\mathbf{M}_2}(I)$, we also define

$$X_1 = \{e \in S \setminus I \mid I + e \in \mathcal{I}_1\},$$
$$X_2 = \{e \in S \setminus I \mid I + e \in \mathcal{I}_2\}.$$

In the auxiliary graph, a directed path from X_2 to X_1 is an *augmenting path*. Let P be a shortest augmenting path. Define $I \triangle P = (I \setminus P) \cup (P \setminus I)$. It is known (e.g. [170]) that $I \triangle P$ is another common independent set, whose size is one larger than I. If there is no augmenting path in the auxiliary graph, then I is already a maximum-cardinality common independent set. Thus, we can find a maximum-cardinality common independent set in a polynomial number of oracle calls; starting with a common independent set $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ (I can be $\emptyset$), we repeatedly augment the current common independent set I to a larger one by finding a shortest augmenting path in $G_{\mathbf{M}_1,\mathbf{M}_2}(I)$. The algorithm constructs an auxiliary graph in each iteration, which takes $O(nr)$ independence oracle calls. Since the number of augmentations is at most r, it runs in $O(nr^2\tau)$ time.

Cunningham [31] improves the running time to $O(nr^{1.5}\tau)$ by finding a maximal number of disjoint augmenting paths in each iteration. For graphic matroids, we can obtain augmentation-type algorithms running in $O(\sqrt{r}n \log r)$ time [68], and $O(\sqrt{r}n)$ time if $n = \Omega(r^{1.5} \log r)$ [67].

Given two matroids $\mathbf{M}_\ell = (S, \mathcal{I}_\ell)$ ($\ell = 1, 2$) and a weight function $w\colon S \to \mathbb{Z}_{\geq 0}$, the *weighted matroid intersection problem* is to find a common independent set with maximum weight. A pair of functions $w_\ell\colon S \to \mathbb{Z}_{\geq 0}$ for $\ell = 1, 2$ is a *weight-splitting of* w if $w(e) = w_1(e) + w_2(e)$ for every $e \in S$. Frank gave two different proofs [54, 55] to the following min-max theorem. Note that our result (Theorem 4.2.9) gives an alternative proof of Theorem 3.2.2, as our algorithm does not rely on Theorem 3.2.2.

Theorem 3.2.2. *Let $\mathbf{M}_1 = (S, \mathcal{I}_1)$ and $\mathbf{M}_2 = (S, \mathcal{I}_2)$ be two matroids and $w\colon S \to \mathbb{Z}_{\geq 0}$ a weight*

function. Then the maximum weight of a common independent set is equal to

$$\min_{w_1,w_2:\textit{weight-splitting}} b_1(w_1) + b_2(w_2),$$

where $b_\ell(w_\ell)$ denotes the weight of the w_ℓ-maximum independent set of $\mathbf{M}_\ell$ for $\ell = 1, 2$.

3.3 Exact Algorithm

In this section, we present an exact algorithm for the weighted matroid intersection problem. Let $W = \max_{e \in S} w(e)$.

Our algorithm runs in W rounds. For ease of presentation, our algorithm starts from Round W and down to Round 1. In Round i, the subset $S' \subseteq S$ of elements e with $w(e) \geq i$ is the ground set of the two matroids.

We maintain a pair of weight functions w_1 and w_2 as a weight splitting of the original weight w. We define a new pair of matroids $\mathbf{M}'_1$ and $\mathbf{M}'_2$ as the restrictions of $\mathbf{M}_1^{w_1}$ and $\mathbf{M}_2^{w_2}$ to S'. In each round, the algorithm finds a maximum-cardinality common independent set I between $\mathbf{M}'_1$ and $\mathbf{M}'_2$ using I', where I' is the common independent set found in the previous round. As we will show in Section 3.5, the augmentation-type algorithm described in Section 3.2.2 can be used to obtain I with the additional property called *near-optimality* (see Definition 3.3.1). At the end of the round, we update w_1, w_2 based on the auxiliary graph $G_{\mathbf{M}'_1,\mathbf{M}'_2}(I)$. Below we first present the algorithm and then elaborate on the details.

Algorithm 1: Exact algorithm

Input: two matroids $\mathbf{M}_1 = (S, \mathcal{I}_1)$ and $\mathbf{M}_2 = (S, \mathcal{I}_2)$, a weight function $w\colon S \to \mathbb{Z}_{\geq 0}$, and $W = \max_{e \in S} w(e)$.

Output: $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ where I is a maximum-weight common independent set of $\mathbf{M}_1$ and $\mathbf{M}_2$.

Step 1. Set $i := W$, $w_1 := 0$, $w_2 := w$, and $I' := \emptyset$.

Step 2. While $i > 0$ do the following steps.

(2-1) Set $S' := \{e \in S \mid w_2(e) \geq i\}$.

(2-2) Set $\mathbf{M}'_\ell = (S', \mathcal{I}'_\ell)$ to be $\mathbf{M}_\ell^{w_\ell}|S'$ for $\ell = 1, 2$.

(2-3) Unweighted_Matroid_Intersection (I')

Construct I so that

(i) I is a maximum-cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, and

(ii) I is (w_1, w_2)-near-optimal in S'.

(2-4) Update_Weight

(2-4-1) Let $T \subseteq S'$ be the set of elements reachable from X_2 in $G_{\mathbf{M}'_1,\mathbf{M}'_2}(I)$.

(2-4-2) For each $e \in T$, let $w_1(e) := w_1(e) + 1$, $w_2(e) := w_2(e) - 1$.

(2-5) Set $i := i - 1$ and $I' := I$.

Step 3. Return I.

Note that in Step (2-3), Unweighted_Matroid_Intersection takes I', which is the common independent set computed in the previous round, to construct I. The implementation details (depending on the type of given matroids) will be given in Section 3.5. Roughly speaking, we will show that if I' is already (w_1, w_2)-near-optimal in S', then we can compute I, based on I', so that I becomes the maximum cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$ while remaining (w_1, w_2)-near-optimal.

3.3.1 Analysis

The final goal of our algorithm is to find a common independent set that is w_1-maximum in $\mathbf{M}_1$ and w_2-maximum in $\mathbf{M}_2$, which would imply that I is w-maximum if $w = w_1 + w_2$. For each integer t, let

$$Z_1(t) = \{e \in S \mid w_1(e) \geq t\},$$
$$Z_2(t) = \{e \in S \mid w_2(e) \geq t\}.$$

Lemma 3.2.1(ii) implies that I being w_1-maximum in $\mathbf{M}_1$ and w_2-maximum in $\mathbf{M}_2$ is equivalent to

1. $I \cap Z_1(t)$ is a base of $\mathbf{M}_1|Z_1(t)$ for every integer $t \geq 1$, and
2. $I \cap Z_2(t)$ is a base of $\mathbf{M}_2|Z_2(t)$ for every integer $t \geq 1$.

Such a common independent set I of $\mathbf{M}_1, \mathbf{M}_2$ is called (w_1, w_2)-*optimal*.

We relax the above condition as follows. We here define $Z'_\ell(t) = Z_\ell(t) \cap S'$ for each subset $S' \subseteq S$ and $\ell = 1, 2$.

Definition 3.3.1. *A common independent set I of $\mathbf{M}_1$ and $\mathbf{M}_2$ is* (w_1, w_2)-near-optimal in a subset $S' \subseteq S$ *if*

1. *$I \cap Z'_1(t)$ is a base of $\mathbf{M}_1|Z'_1(t)$ for every integer $t \geq 1$, and*
2. *$I \cap Z'_2(t)$ is a base of $\mathbf{M}_2|Z'_2(t)$ for every integer $t \geq \alpha + 1$, where $\alpha = \min_{e \in S'} w_2(e)$.*

Note that if $\alpha = 0$ and $S' = S$, then a (w_1, w_2)-near-optimal common independent set in S' is (w_1, w_2)-optimal.

In what follows, we will prove that, during the execution of our algorithm, the current set I is always (w_1, w_2)-near-optimal in S'. To prove this, we analyze the two procedures Unweighted_Matroid_Intersection and Update_Weight used in Steps (2-3) and (2-4).

In Unweighted_Matroid_Intersection of Step (2-3), if we only want a maximum-cardinality common independent set I of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, the step is trivial. The difficulty is how to guarantee that I is also (w_1, w_2)-near-optimal in S' *without* resorting to weighted matroid intersection. The details are deferred to Section 3.5. We use a lemma to summarize the outcome of Step (2-3). Recall that we denote $\mathbf{M}'_\ell = \mathbf{M}_\ell^{w_\ell}|S'$ for $\ell = 1, 2$.

Lemma 3.3.2. *Suppose that I' is (w_1, w_2)-near-optimal in a subset S'. Then we can construct another common independent set I, using known unweighted matroid intersection algorithms, that is simultaneously* (i) *a maximum-cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, and* (ii) *(w_1, w_2)-near-optimal in S'.*

We next prove that, if the maximum-cardinality common independent set I of $\mathbf{M}'_1$ and $\mathbf{M}'_2$ is (w_1, w_2)-near-optimal in S', then we can modify w_1 and w_2 at Step (2-4) so that I is still (w_1, w_2)-near-optimal in S'.

Lemma 3.3.3. *Suppose that all weights of w_1 and w_2 are nonnegative integers, and there are some integers p_1 and p_2 such that $w_1(e) \leq p_1$ and $w_2(e) \geq p_2$ for every $e \in S'$. In addition, suppose that I is* (i) *a maximum-cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, and* (ii) *(w_1, w_2)-near-optimal in S'. Then, after the procedure* Update_Weight*, we have*

(1) *$I \cap Z'_1(t)$ is a base of $\mathbf{M}_1|Z'_1(t)$ for every integer t with $1 \leq t \leq p_1 + 1$, and*

(2) *$I \cap Z'_2(t)$ is a base of $\mathbf{M}_2|Z'_2(t)$ for every integer $t \geq p_2$.*

It should be noted that Lemma 3.3.3 implies that after Step (2-4), I is still (w_1, w_2)-near-optimal in S', since then $\max_{e \in S'} w_1(e) \leq p_1 + 1$ and $\min_{e \in S'} w_2(e) \geq p_2 - 1$.

Proof. We only prove (1), since (2) follows symmetrically. To avoid confusion, let $\tilde{Z}'_1(t)$ denote the set $Z'_1(t)$ *after* the weights w_1 and w_2 are updated. Observe that, for every integer t with $1 \leq t \leq p_1 + 1$,

$$\tilde{Z}'_1(t) = Z'_1(t) \cup ((Z'_1(t-1) \setminus Z'_1(t)) \cap T),$$

where we note that $Z'_1(p_1 + 1) = \emptyset$ and $Z'_1(0) = S'$.

As $I \cap Z'_1(t)$ is a base of $\mathbf{M}_1|Z'_1(t)$, we argue that given an element $e \in ((Z'_1(t-1) \setminus Z'_1(t)) \cap T) \setminus I$:

$(*)$ $I + e \notin \mathcal{I}_1|S'$, and

$(**)$ the circuit of $I + e$ in $\mathbf{M}_1|S'$ is contained in $\tilde{Z}'_1(t)$.

This will establish that $I \cap \tilde{Z}'_1(t)$ is a base of $\mathbf{M}_1|\tilde{Z}'_1(t)$ for every $t = 1, 2, \ldots, p_1 + 1$.

To see $(*)$, observe that in $G_{\mathbf{M}'_1, \mathbf{M}'_2}(I)$, e is not part of X_1. Otherwise, there would be an augmenting path, contradicting to the assumption that I is a maximum-cardinality common independent set in $\mathbf{M}'_1$ and $\mathbf{M}'_2$. Thus, $I + e$ contains a circuit C' in $\mathbf{M}'_1$. Furthermore, by Lemma 3.2.1(iii) applied to $\mathbf{M}_1|S'$ (as the assumption is that $I \cap Z'_1(t)$ is a base of $\mathbf{M}_1|Z'_1(t)$ for every $t = 1, 2, \ldots, p_1$), $I + e$ also has a circuit $C \supseteq C'$ in $\mathbf{M}_1|S'$. Thus, $(*)$ is proved.

To see $(**)$, consider an element e' in $C' - e$. Then, e' is contained in $Z'_1(t-1) \setminus Z'_1(t)$ by Lemma 3.2.1(iii). Since $e' \in C'$, in $G_{\mathbf{M}'_1, \mathbf{M}'_2}(I)$, there is an arc from e to e'. Thus, e' is part of T. This implies that C' is a subset of $(Z'_1(t-1) \setminus Z'_1(t)) \cap T$, which in turn, by Lemma 3.2.1(iii), implies that the circuit $C \supseteq C'$ in $I + e$ with respect to $\mathbf{M}_1|S'$ is a subset of $Z'_1(t) \cup ((Z'_1(t-1) \setminus Z'_1(t)) \cap T) = \tilde{Z}'_1(t)$. The proof of $(**)$ follows.

□

By induction on i with Lemmas 3.3.2 and 3.3.3, we show that the current set I is always (w_1, w_2)-near-optimal.

Lemma 3.3.4. *In Round i with $1 \leq i \leq W$, the following holds.*

(1) $w = w_1 + w_2$,

(2) *after Step* (2-4), $I \cap Z'_1(t)$ *is a base of* $\mathbf{M}_1|Z'_1(t)$ *for every integer* t *with* $1 \leq t \leq W - i + 1$, *and*

(3) *after Step* (2-4), $I \cap Z'_2(t)$ *is a base of* $\mathbf{M}_2|Z'_2(t)$ *for every integer* t *with* $i \leq t \leq W$.

Proof. (1) can be easily seen. We prove (2) and (3) by induction on i.

For the base case of $i = W$, since $Z'_1(1) = \emptyset$ and $Z'_2(W + 1) = \emptyset$ hold, $I' = \emptyset$ is (w_1, w_2)-near-optimal in S', and thus Lemma 3.3.2 implies that we can obtain a maximum-cardinality common independent set I of $\mathbf{M}'_1$ and $\mathbf{M}'_2$ satisfying the condition that $I \cap Z'_1(1)$ is a base of $\mathbf{M}_1|Z'_1(1)$ and $I \cap Z'_2(W + 1)$ is a base of $\mathbf{M}_2|Z'_2(W + 1)$. Now applying Lemma 3.3.3 (with $p_1 = 0$ and $p_2 = W$), we have that $I \cap Z'_1(1)$ is a base of $\mathbf{M}_1|Z'_1(1)$ and $I \cap Z'_2(W)$ is a base of $\mathbf{M}_2|Z'_2(W)$.

For the induction step $i < W$, let I' be the common independent set obtained in Round $i + 1$. By induction hypothesis, $I' \cap Z'_1(t)$ is a base of $\mathbf{M}_1|Z'_1(t)$ for every integer t with $1 \leq t \leq W - i$ and $I' \cap Z'_2(t)$ is a base of $\mathbf{M}_2|Z'_2(t)$ for every integer t with $i + 1 \leq t \leq W$. Notice that when Round i begins, only elements e with $w_1(e) = 0$ and $w_2(e) = i$ are added to S'. Hence the two conditions remain true after Step (2-1).

By these facts, as $w_2(e) \geq i$ for $e \in S'$, Step (2-3) can be correctly applied by Lemma 3.3.2, and we obtain the new independent set I satisfying the two conditions stated in Step (2-3). The proof now follows by applying Lemma 3.3.3 (with $p_1 = W - i$ and $p_2 = i$).

□

Theorem 3.3.5. *The common independent set I returned by Algorithm 1 is a maximum-weight common independent set of $\mathbf{M}_1$ and $\mathbf{M}_2$.*

Proof. By Lemma 3.3.4, after the last round when $i = 1$, as $S' = S$, $I \cap Z_1(t)$ is a base of $\mathbf{M}_1|Z_1(t)$ for every $t = 1, 2, \ldots, W$, and $I \cap Z_2(t)$ is a base of $\mathbf{M}_2|Z_2(t)$ for every $t = 1, 2, \ldots, W$. Thus, it follows from Lemma 3.2.1(ii) that I is w_ℓ-maximum in $\mathbf{M}_\ell$ for every $\ell = 1, 2$. Then, for every common independent set J, we have

$$w(J) = w_1(J) + w_2(J) \leq w_1(I) + w_2(I) = w(I).$$

Thus, I is a maximum-weight common independent set. This completes the proof.

□

The algorithm clearly runs in $O(W(T_{\mathrm{u}} + T_{\mathrm{d}}))$ time, where T_{u} and T_{d} are the running times for executing Unweighted_Matroid_Intersection and Update_Weight, respectively. Note that T_{u} and T_{d} depend on the representation of the given matroids. Their complexities are discussed in Section 3.5.

3.4 Approximation Algorithm

In this section, we will design a $(1 - \epsilon)$-approximation algorithm for the weighted matroid intersection. Let W be the maximum weight. First of all, we show that we can round weights to small

integers, and bound W from above.

Lemma 3.4.1. *We can reduce a given instance of the weighted matroid intersection problem to one with integral weights whose maximum weight is at most $2r_*/\epsilon$, where $r_* \le r$ is the maximum size of a common independent set.*

Proof. Set $\eta = \epsilon W/2r_*$, and define $w'(e) = \lfloor w(e)/\eta \rfloor$ for each $e \in S$. Then, a $(1-\epsilon/2)$-approximate solution I' for the weight w' is a $(1-\epsilon)$-approximate solution for the weight w. Indeed, since $w(e) - \eta \le \eta w'(e) \le w(e)$ for every $e \in S$, we have

$$\begin{aligned}
w(I') &\ge \eta w'(I') \\
&\ge \eta(1-\epsilon/2)w'(I'_{\mathrm{opt}}) && (I'_{\mathrm{opt}} \text{ is an optimal solution for } w') \\
&\ge \eta(1-\epsilon/2)w'(I_{\mathrm{opt}}) && (I_{\mathrm{opt}} \text{ is an optimal solution for } w) \\
&\ge (1-\epsilon/2)(w(I_{\mathrm{opt}}) - \eta|I_{\mathrm{opt}}|) \\
&\ge (1-\epsilon/2)(w(I_{\mathrm{opt}}) - \eta r_*) \\
&= (1-\epsilon/2)(w(I_{\mathrm{opt}}) - \epsilon W/2) \\
&\ge (1-\epsilon)w(I_{\mathrm{opt}}),
\end{aligned}$$

where the last inequality follows because we assume that the given matroids have no loop, so the element e with $w(e) = W$ is a common independent set, thus, $w(I_{\mathrm{opt}}) \ge W$.

□

During the algorithm, the weight w is split so that $w \approx w_1 + w_2$; furthermore, we will guarantee that all weights of w_1 and w_2 are nonnegative multiples of some integer $\delta > 0$, where δ may change in different phases of the algorithm. At the end, we find a common independent set that is w_1-maximum in $\mathbf{M}_1$ and w_2-maximum in $\mathbf{M}_2$, which would imply that I is a $(1-\epsilon)$-approximate solution if $w \le w_1 + w_2 \le (1+\epsilon)w$.

For simplicity, we assume that the bound W and ϵ are both powers of 2. Then, our algorithm runs in $1 + \log_2 \epsilon W$ phases. In every phase, we apply a number (roughly $O(\epsilon^{-1})$) of Unweighted_Matroid_Intersection and Update_Weight operations. Note that $\log_2 \epsilon W = O(\log r)$ by Lemma 3.4.1.

Let $\delta_0 = \epsilon W$. For each integer i with $1 \le i \le \log_2 \epsilon W$, define $\delta_i = \delta_0/2^i$. The term δ_i will be the amount of change in the weights w_1 and w_2 during Phase i every time Update_Weight is invoked. For each $e \in S$ and each integer i with $0 \le i \le \log_2 \epsilon W$, define $w^i(e)$ to be the truncated weight of element e in Phase i, i.e., $w^i(e) = \lfloor w(e)/\delta_i \rfloor \delta_i$. Notice that $w^{i+1}(e) = w^i(e)$ or $w^{i+1}(e) = w^i(e) + \delta_{i+1}$. The algorithm, presented below, returns a $(\frac{1}{1+4\epsilon})$-approximate solution.

Algorithm 2: Approximation algorithm

Input: two matroids $\mathbf{M}_1 = (S, \mathcal{I}_1)$ and $\mathbf{M}_2 = (S, \mathcal{I}_2)$, a weight function $w\colon S \to \mathbb{Z}_{\ge 0}$, and $W = \max_{e \in S} w(e)$.

Output: $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ where $w(I) \ge \frac{w(I^{opt})}{1+4\epsilon}$.

Step 1. Set $i := 0$, $w_1 := 0$, $w_2 := w^0$, $I' := \emptyset$, and $h := W$.

Step 2. Applying Algorithm 1:

While $i \leq \log_2 \epsilon W$, do the following steps.

(2-0) Set $L := \frac{W}{2^{i+1}}$ if $i < \log_2 \epsilon W$, and $L := 1$ if $i = \log_2 \epsilon W$.

(2-1) While $h \geq L$, do the following steps.

(2-1-1) Set $S' := \{e \in S \mid w_2(e) \geq h\}$.

(2-1-2) Set $\mathbf{M}'_\ell = (S', \mathcal{I}'_\ell)$ to be $\mathbf{M}_\ell^{w_\ell}|S'$ for each $\ell = 1, 2$.

(2-1-3) Unweighted_Matroid_Intersection

Construct I using I' so that

(i) I is a maximum-cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, and

(ii) I is (w_1, w_2)-near-optimal in S'.

(2-1-4) Update_Weight

(i) Let $T \subseteq S'$ be the set of elements reachable from X_2 in $G_{\mathbf{M}'_1, \mathbf{M}'_2}(I)$.

(ii) For each $e \in T$, let $w_1(e) := w_1(e) + \delta_i$, $w_2(e) := w_2(e) - \delta_i$.

(2-1-5) Set $h := h - \delta_i$ and $I' := I$.

(2-2) Weight Adjustment:

If $i < \log_2 \epsilon W$, do the following.

(2-2-1) $\forall e \in I'$, let $w_2(e) = w_2(e) + \delta_{i+1}$.

(2-2-2) $\forall e \in S \setminus I'$ where $w^{i+1}(e) = w^i(e) + \delta_{i+1}$, let $w_2(e) = w_2(e) + \delta_{i+1}$.

(2-2-3) Set $h := h + \delta_{i+1}$.

(2-3) Set $i := i + 1$.

Step 3. Return I.

The outer loop Step 2 corresponds to a phase. We use a counter h to keep track of the progress of the algorithm. Initially $h = W$. In Phase i, the weights are always kept as nonnegative multiples of δ_i. In Step (2-1), the two matroids $\mathbf{M}'_1$ and $\mathbf{M}'_2$ are defined on the common ground set $S' = \{e \in S \mid w_2(e) \geq h\}$, and the two procedures Unweighted_Matroid_Intersection and Update_Weight are invoked as was done in the exact algorithm (Algorithm 1) in Section 3.3. The counter h is decreased by the amount of δ_i each time after Update_Weight is invoked in Step (2-1).

Each time h is halved, we make ready to move to the next phase, except in the last phase: in Phase $\log_2 \epsilon W$, we stop when h goes down to 1. The reason that we adjust the w_2-weights at Step (2-2) is that we want to ensure that in the beginning of the next phase, the weights w_1 and w_2 still approximate the next weight w^{i+1} (see Lemma 3.4.5). In particular, we increase the w_2-weights of all elements in the current common independent set I'. This is to make sure that I' is still w_2-maximum in the beginning of the next phase (with respect to the newly-defined set S' in Step (2-1)).

3.4.1 Analysis

We first observe the number of iterations in the algorithm.

Lemma 3.4.2. (1) *During Phase i with $0 \leq i \leq \log_2 \epsilon W$, w_1 and w_2 are nonnegative multiples of δ_i, except in Step* (2-2).

(2) *Step* (2-1) *is executed at most $\frac{\epsilon^{-1}}{2}$ times in Phase i with $0 \leq i < \log_2 \epsilon W$. In the last phase, Step* (2-1) *is executed $\epsilon^{-1} + 1$ times.*

(3) *The total number of iterations in Step* (2-1) *is $O(\epsilon^{-1} \log r)$.*

Proof. (1) can be easily verified. For (2), observe that in Phase 0, Step (2-1) is executed

$$\frac{W - W/2}{\delta_0} = \frac{\epsilon^{-1}}{2}$$

times. For Phase $i \geq 1$, in the beginning of that phase, $h = \frac{W}{2^i} - \delta_i$. Hence, if $i < \log_2 \epsilon W$, Step (2-1) is executed

$$\frac{(W/2^i - \delta_i) - W/2^{i+1}}{\delta_i} \leq \frac{\epsilon^{-1}}{2}$$

times, and if $i = \log_2 \epsilon W$, Step (2-1) is executed

$$\frac{W/2^i - \delta_i}{\delta_i} \leq \epsilon^{-1}$$

times. (3) now immediately follows from (2).

□

We say an element $e \in S$ *joins* in Phase j if in Phase j, element e becomes a part of the ground set S' in Step (2-1-1) the first time.

Lemma 3.4.3. *Suppose that an element $e \in S$ joins in Phase j for some integer j with $j < \log_2 \epsilon W$. Then the following holds.*

(1) $w^j(e) \geq \frac{W}{2^{j+1}} = \frac{\delta_j}{2\epsilon}$.

(2) *In every phase $i \geq j$, $w^i(e) \leq w_1(e) + w_2(e) \leq w^i(e) + 2\delta_j$.*

(3) *If $e \in S$ joins in the last phase $j = \log_2 \epsilon W$, then $w_1(e) + w_2(e) = w^j(e)$.*

Proof. Notice that immediately before e joins in Phase j, we have $w_1(e) + w_2(e) = w^j(e)$. This follows from the observation that unless e is part of I when Step (2-2-1) is executed, the weight splitting $w_1(e)$ and $w_2(e)$ is *exact* with respect $w^{j'}(e)$ for $j' \leq j$. (3) follows easily from this observation. In the case that $j < \log_2 \epsilon W$, we have that $w^j(e) \geq w_2(e) \geq \frac{W}{2^{j+1}}$. Thus (1) is proved.

(2) follows from the fact that the difference between the sum of $w_1(e)$ and $w_2(e)$ and the truncated weight $w^{j'}(e)$ grows larger only when Step (2-2-1) is executed in Phase $j' \geq j$ and e is part of the common independent set I in that step. Hence it holds that

$$\begin{aligned} w^i(e) \leq w_1(e) + w_2(e) &\leq w^i(e) + \sum_{s=j}^{i} \delta_s \\ &\leq w^i(e) + 2\delta_j. \end{aligned}$$

This completes the proof.

□

Since all weights of w_1, w_2 are nonnegative multiples of δ_i and we modify w_1 and w_2 by δ_i at Update_Weight, we have the following lemma, which can be obtained similarly to Lemma 3.3.3 by dividing all the values by δ_i.

Lemma 3.4.4. *Suppose that all weights of w_1 and w_2 are nonnegative multiples of δ, and there are some integers p_1 and p_2 such that $w_1(e) \leq p_1$ and $w_2(e) \geq p_2$ for every $e \in S'$. In addition, suppose that I is* (i) *a maximum-cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, and* (ii) *(w_1, w_2)-near-optimal in S'. Then after the procedure* Update_Weight*, we have*

(1) *$I \cap Z'_1(t)$ is a base of $\mathbf{M}_1|Z'_1(t)$ for every integer t with $1 \leq t \leq p_1 + \delta$, and*

(2) *$I \cap Z'_2(t)$ is a base of $\mathbf{M}_2|Z'_2(t)$ for every integer $t \geq p_2$.*

Note that the lemma implies that the current independent set I is still (w_1, w_2)-near-optimal in S' after Step (2-1-4).

We finally see that Weight_Adjustment maintains I' (w_1, w_2)-near-optimal in S'.

Lemma 3.4.5. *In Phase i, after Step* (2-1) *terminates, we have the following.*

(1) *$I' \cap Z'_1(t)$ is a base of $\mathbf{M}_1|Z'_1(t)$ for every integer $t \geq 1$.*

(2) *$I' \cap Z'_2(t)$ is a base of $\mathbf{M}_2|Z'_2(t)$ for every integer $t \geq h + \delta_i$.*

Proof. We first prove the following claim.

> **Claim 1.** *In each phase, if* (1) *and* (2) *hold before the first iteration of Step* (2-1) *starts, we have* (1) *and* (2) *after the final iteration of Step* (2-1) *terminates.*
>
> *Proof.* We prove the claim by induction on the number of times Step (2-1) is invoked. For the base case, we have (1) and (2) in the beginning by the assumption.
> Suppose that we have (1) and (2) for the previous set I' at the beginning of the current iteration in Step (2-1). At Step (2-1-1), some elements may be added into S'. However, all such elements have $w_1(e) = 0$ and $w_2(e) = h$. Thus, I' still satisfies (1) and (2), and thus it is (w_1, w_2)-near-optimal in S' since $w_2(e) \geq h$ for every $e \in S'$. By Lemma 3.3.2, Step (2-1-3) can be correctly implemented, and we obtain a maximum-cardinality common independent set I of $\mathbf{M}'_1$ and $\mathbf{M}'_2$ that is (w_1, w_2)-near-optimal in S'. After Step (2-1-4), by Lemma 3.4.4 (by setting $\delta = \delta_i$, $p_1 = \max_{e \in S'} w_1(e)$, and $p_2 = h$), we have that I satisfies (1) and $I \cap Z'_2(t)$ is a base of $\mathbf{M}_2|Z'_2(t)$ for any integer $t \geq h$. Since h is decreased by δ_i in Step (2-1-5), we have (1) and (2) at the end of the current iteration. This proves the claim.
>
> □

We prove the lemma by induction on the number of phases. For the base case, as in the beginning of the algorithm, $h = W$ and $I' = \emptyset$, the set I' is (w_1, w_2)-near-optimal in S'. This means that we have (1) and (2) for I', and hence Claim 1 implies that we have (1) and (2) after the iterations of Step (2-1) terminates in Phase 0.

For the induction step, suppose that currently the algorithm is in Phase i, and that (1) and (2) are satisfied after Step (2-1) are done. We argue that after the weight adjustment done in Step (2-2), I' still satisfies (1) and (2).

To avoid confusion, let $\tilde{Z}_\ell(t)$ $(\ell = 1, 2)$ denote the sets *after* w_2-weights are modified in Steps (2-2-1) and (2-2-2), and let $\tilde{h}$ be the value of h after Step (2-2-3), i.e., $\tilde{h} = h + \delta_{i+1}$.

By Lemma 3.4.2(1), all w_1 and w_2 weights are multiples of δ_i in Phase i before Step (2-2). Therefore, after Step (2-1), the fact that I' satisfies (2) implies

($\star$) $I' \cap Z'_2(t)$ is a base of $\mathbf{M}_2|Z'_2(t)$ for every integer $t \geq h + \delta_{i+1}$.

To see this, note that I' satisfying (2) only guarantees this property for $t \geq h + \delta_i$. We can subtract δ_{i+1} further because there is no element with w_2-weight of the form $a\delta_i + \delta_{i+1}$ for some integer $a \geq 0$. Hence the range of t starts from $h + \delta_i - \delta_{i+1} = h + \delta_{i+1}$.

As we increase the w_2-weights of all elements in I' and a subset of elements in $S' \setminus I'$, while leaving the w_1-weights unchanged, we have

(i) $\tilde{Z}_1(t) = Z_1(t)$ for all $t \in \mathbb{Z}_{\geq 0}$.

(ii) $I \cap \tilde{Z}'_1(t)$ is a base of $\mathbf{M}_1|\tilde{Z}'_1(t)$ for every integer $t \geq 1$.

(iii) $I \cap \tilde{Z}'_2(t)$ is a base of $\mathbf{M}_2|\tilde{Z}'_2(t)$ for every integer $t \geq \tilde{h} + \delta_{i+1}$.

(i) and (ii) are easy to see, since w_1-weights are unchanged and (1) holds before Step (2-2). For (iii), consider any integer $t \geq \tilde{h} + \delta_{i+1} = h + 2\delta_{i+1}$. We have that $I' \cap \tilde{Z}'_2(t) = I' \cap Z'_2(t - \delta_{i+1})$, where the latter is a base of $\mathbf{M}_2|Z'_2(t-\delta_{i+1})$ by ($\star$). As $\tilde{Z}'_2(t) \subseteq Z'_2(t-\delta_{i+1})$, we infer that $I' \cap \tilde{Z}'_2(t)$ is still a base of $\mathbf{M}_2|\tilde{Z}'_2(t)$.

Therefore, at the beginning of Phase $i + 1$, we have (1) and (2), and hence the proof follows from Claim 1. This completes the proof.

□

Lemma 3.4.6. *The common independent set I returned by Algorithm 2 is a maximum-weight common independent set, with respect to the final weight function $w_1 + w_2$.*

Proof. After the last time Step (2-1-5) is executed, by Lemma 3.4.5 and the fact that $S' = S$, $I \cap Z_1(t)$ is a base of $\mathbf{M}_1|Z_1(t)$ for every integer $t \geq 1$, and $I \cap Z_2(t)$ is a base of $\mathbf{M}_2|Z_2(t)$ for every integer $t \geq \delta_{\log_2 \epsilon W}$. Since $\delta_{\log_2 \epsilon W} = 1$, it follows from Lemma 3.2.1(ii) that I is w_1-maximum in $\mathbf{M}_1$ and w_2-maximum in $\mathbf{M}_2$. Therefore, for every common independent set J, we have

$$w_1(J) + w_2(J) \leq w_1(I) + w_2(I).$$

The proof follows.

□

Theorem 3.4.7. *Let I be the common independent set returned by Algorithm 2. Then I is a $(1-4\epsilon)$ approximation.*

Proof. For every $e \in S$, if it joins in Phase $j < \log_2 \epsilon W$, then by Lemma 3.4.3(2),

$$\begin{aligned} w^{\log_2 \epsilon W}(e) &\leq w_1(e) + w_2(e) \leq w^{\log_2 \epsilon W}(e) + 2\delta_j \\ &\leq (1+4\epsilon) w^{\log_2 \epsilon W}(e), \end{aligned}$$

where the last inequality holds since $\delta_j \leq 2\epsilon w^j(e) \leq 2\epsilon w^{\log_2 \epsilon W}(e)$ by Lemma 3.4.3(1). If $j = \log_2 \epsilon W$, then $w^{\log_2 \epsilon W}(e) = w_1(e) + w_2(e)$ by Lemma 3.4.3(3). Since $w^{\log_2 \epsilon W}(e) = w(e)$, we conclude that, for each $e \in S$,

$$w(e) \leq w_1(e) + w_2(e) \leq (1+4\epsilon) w(e).$$

Thus, letting I_{opt} be the maximum-weight common independent set, Lemma 3.4.6 implies

$$\begin{aligned} w(I_{\text{opt}}) &\leq w_1(I_{\text{opt}}) + w_2(I_{\text{opt}}) \leq w_1(I) + w_2(I) \\ &\leq (1+4\epsilon) w(I). \end{aligned}$$

The proof follows. □

3.5 Implementation of Unweighted Matroid Intersection

In this section, we discuss how to implement the procedure Unweighted_Matroid_Intersection and the actual complexities of our algorithms for various weighted matroid intersection problems.

Let $\mathbf{M}_1$ and $\mathbf{M}_2$ be two matroids, and w_1 and w_2 be weights. Suppose that a given common independent set I' of $\mathbf{M}'_1$ and $\mathbf{M}'_2$ is (w_1, w_2)-near-optimal in a subset $S' \subseteq S$ (recall that $\mathbf{M}'_\ell = \mathbf{M}_\ell^{w_\ell}|S'$ for $\ell = 1, 2$). We explain in the following sections how to find a maximum-cardinality common independent set I between $\mathbf{M}'_1$ and $\mathbf{M}'_2$ that is (w_1, w_2)-near-optimal in S'.

3.5.1 General Matroids

In [31], Cunningham shows how to find a maximum-cardinality common independent set, using $O(nr^{1.5})$ independence oracle calls. This is done by repeatedly finding an augmenting path in the auxiliary graph, as described in Section 3.2.2. We argue that if we apply his algorithm to $\mathbf{M}'_1$ and $\mathbf{M}'_2$ with I' as the initial common independent set, each new independent set resulting from augmentation will satisfy the same property as I'.

Lemma 3.5.1. *Suppose that I' is (w_1, w_2)-near-optimal in S', and let P be the shortest path from X_2 to X_1 in $G_{\mathbf{M}'_1, \mathbf{M}'_2}(I')$. Then, the set $I = I' \triangle P$ is also (w_1, w_2)-near-optimal.*

Proof. By Lemma 3.2.1(iii), in $G_{\mathbf{M}'_1, \mathbf{M}'_2}(I')$, an element $e \in (Z_1(t) \setminus Z_1(t+1)) \setminus I'$ has outgoing arcs to only other elements in $Z_1(t) \setminus Z_1(t+1)$ for every integer $t \geq 1$. Similarly, an element

$e \in (Z_2(t) \setminus Z_2(t+1)) \cap I'$ has only outgoing arcs towards other elements in $(Z_2(t) \setminus Z_2(t+1)) \setminus I'$ for every integer $t \geq p+1$, where $p = \min_{e \in S'} w_2(e)$.

These two facts imply that along the augmenting path P in $G_{\mathbf{M}'_1, \mathbf{M}'_2}(I')$, the number of elements in $(Z_1(t) \setminus Z_1(t+1)) \setminus I'$ is the same as the number of elements in $(Z_1(t) \setminus Z_1(t+1)) \cap I'$ for every integer $t \geq 1$. Similarly, the number of elements in $(Z_2(t) \setminus Z_2(t+1)) \cap I'$ is the same as that in $(Z_2(t) \setminus Z_2(t+1)) \setminus I'$ for every integer $t \geq p+1$. Thus, $|I \cap Z_1(t)| = |I' \cap Z_1(t)|$ for every integer $t \geq 1$, and $|I \cap Z_2(t)| = |I' \cap Z_2(t)|$ for every integer $t \geq p+1$. The proof follows.

□

Thus, the maximum-cardinality common independent set of $\mathbf{M}'_1 = (S', \mathcal{I}'_1)$ and $\mathbf{M}'_2 = (S', \mathcal{I}'_2)$ obtained by Cunningham's algorithm is (w_1, w_2)-near-optimal if so is the initial set. To apply Cunningham's algorithm [31] to $\mathbf{M}'_1$ and $\mathbf{M}'_2$, we need an independence oracle for $\mathbf{M}'_1$ and $\mathbf{M}'_2$ to find an augmenting path. More specifically, for $\ell = 1, 2$, we need to test whether $I' + e \in \mathcal{I}'_\ell$ and whether $I' + e - f \in \mathcal{I}'_\ell$ for a given independent set I', and given elements $e \in S' \setminus I'$ and $f \in I'$. This can be implemented by an independence oracle for $\mathbf{M}_1$ and $\mathbf{M}_2$ as follows. It follows from Lemma 3.2.1(iii) that if $I' + e \notin \mathcal{I}'_\ell$, then $I' + e - f \in \mathcal{I}'_\ell$ if and only if $I' + e - f \in \mathcal{I}_\ell$ and $w_\ell(e) = w_\ell(f)$. In addition, $I' + e \in \mathcal{I}'_1$ if and only if $I' + e \in \mathcal{I}_1$ and $w_1(e) = 0$, and $I' + e \in \mathcal{I}'_2$ if and only if $I' + e \in \mathcal{I}_2$ and $w'_2(e) = \min_{e \in S'} w_2(e)$. Thus Unweighted_Matroid_Intersection can be implemented in $O(nr^{1.5})$ independence oracle calls for $\mathbf{M}_1$ and $\mathbf{M}_2$.

We can perform Update_Weight in $O(nr)$ independence oracle calls. Therefore, we have the following theorem for two general matroids.

Theorem 3.5.2. *For two general matroids, we can solve the weighted matroid intersection problem exactly in $O(\tau W n r^{1.5})$ time, and $(1-\epsilon)$-approximately in $O(\tau \epsilon^{-1} n r^{1.5} \log r)$ time, where τ is the running time to check the independence of a set in the given matroids.*

For the exact algorithm, a slight sharpening in the running time is possible. Observe that in Round i, Cunningham's algorithm takes $O(\tau |S'| r^{1.5})$ time, where $S' = \{e \in S \mid w(e) \geq i\}$. Since

$$\sum_{i=1}^{W} |\{e \in S \mid w(e) \geq i\}| = \sum_{e \in S} w(e),$$

the total running time is $O(\tau (\sum_{e \in S} w(e)) r^{1.5})$. This is superior to the previous one only when the given weights are very "unbalanced."

3.5.2 Graphic Matroids

Suppose that $\mathbf{M}_1$ and $\mathbf{M}_2$ are graphic matroids. That is, $\mathbf{M}_\ell = (S, \mathcal{I}_\ell)$ $(\ell = 1, 2)$ is represented by a graph $G_\ell = (V_\ell, S)$ so that $\mathcal{I}_\ell$ is the family of edge subsets in S that are forests in G_ℓ. Note that the number of edges in G_ℓ is $n = |S|$, and the number of vertices is $O(r)$, since we may assume that there is no isolated vertex. Gabow and Xu [68] designed an algorithm that runs in $O(\sqrt{r} n \log r)$ time for the unweighted graphic matroid intersection. Their algorithm is an augmentation-type algorithm, that means it repeatedly finds an augmenting path in the auxiliary graph.

It is well known that, if $\mathbf{M}_\ell$ is graphic, then so is $\mathbf{M}'_\ell = \mathbf{M}_\ell^{w_\ell} | S'$ for a subset S' and $\ell = 1, 2$. Indeed, for a subset $X \subseteq S$, the restriction of G_ℓ to X (the subgraph induced by an edge subset

X), denoted by $G_\ell|X$, represents $\mathbf{M}_\ell|X$. Moreover, the graph obtained from G_ℓ by contracting X, denoted by G_ℓ/X, represents $\mathbf{M}_\ell/X$. Then, by Lemma 3.2.1(i), $\mathbf{M}'_\ell = \mathbf{M}_\ell^{w_\ell}|S'$ has a graph representation $G'_\ell|S'$, where G'_ℓ is in the form of

$$G'_\ell = \bigoplus_{t=0}^{W} (G_\ell|Z_\ell(t))/Z_\ell(t+1),$$

i.e., G'_ℓ is the disjoint union of graphs $(G_\ell|Z_\ell(t))/Z_\ell(t+1)$ obtained by restriction and contraction. Note that the numbers of vertices and edges in G' are $O(r)$ and n, respectively.

We apply Gabow and Xu's algorithm [68] for the unweighted problem to $\mathbf{M}'_1$ and $\mathbf{M}'_2$ with I' as the initial common independent set. Since I' is (w_1, w_2)-near-optimal, it follows from Lemma 3.5.1 that the obtained maximum-cardinality common independent set is (w_1, w_2)-near-optimal in S'. Thus the running time of Unweighted_Matroid_Intersection is $O(\sqrt{r}n \log r)$. Since the reachable set T in the procedure Update_Weight can be found in the end of Gabow and Xu's algorithm, we can perform Update_Weight in linear time. Therefore, we have the following.

Theorem 3.5.3. *For two graphic matroids, we can solve the weighted matroid intersection problem exactly in $O(W\sqrt{r}n \log r)$ time, and $(1-\epsilon)$-approximately in $O(\epsilon^{-1}\sqrt{r}n \log^2 r)$ time.*

3.5.3 Linear Matroids

In the case that $\mathbf{M}_1$ and $\mathbf{M}_2$ are linear, we can use a faster algorithm by Harvey [84] instead of the augmentation-type algorithm. His algorithm is an algebraic one for finding a common base of two linear matroids. We reduce our instance to the problem of finding a common base, that corresponds to a (w_1, w_2)-near-optimal maximum-cardinality common independent set.

We first describe basic properties of a linear matroid $\mathbf{M} = (S, \mathcal{I})$ of rank r. We assume that $\mathbf{M}$ is represented by an $r \times n$ matrix A whose column set is S and row set is denoted by R. We denote by $A[I, J]$ the submatrix consisting of row set I and column set J. For a set X, we denote the complement by $\overline{X}$.

It is known that the restriction and contraction of the linear matroid $\mathbf{M}$ are both linear. Indeed, for a subset $X \subseteq S$, $\mathbf{M}|X$ has the matrix representation $A|X = A[R, X]$. Moreover, taking a nonsingular submatrix of maximum size in $A[R, X]$, denoted by $A[Y, Z]$, we have the matrix representation A/X of the contraction $\mathbf{M}/X$ in the form of

$$A/X = A[\overline{Y}, \overline{X}] - A[\overline{Y}, Z]A[Y, Z]^{-1}A[Y, \overline{X}].$$

The row set of A/X is $\overline{Y} = R \setminus Y$. See e.g., [83] for more details. The direct sum of linear matroids $\mathbf{M}_1$ and $\mathbf{M}_2$ is also linear, whose matrix representation is the block diagonal matrix arranging the two matrices for $\mathbf{M}_1$ and $\mathbf{M}_2$ on the diagonal.

Suppose that we are given a weight function $w\colon S \to \{0, 1, \ldots, W\}$. Then, by Lemma 3.2.1(i), $\mathbf{M}^w$ is also linear, and its matrix representation A^w is in the form of

$$A^w = \bigoplus_{t=0}^{W} (A|Z(t))/Z(t+1), \tag{3.1}$$

where we recall $Z(t) = \{e \in S \mid w(e) \geq t\}$ for $t = 0, \ldots, W+1$. The size of A^w is the same as A; the ground set of $\mathbf{M}^w$ is S, and the row set of A^w is also R. We denote by $Y(t)$ the set of the nonzero rows in $A^w[R, Z(t)]$ for $t = 0, \ldots, W$. Thus A^w is a block-diagonal matrix whose blocks are $A^w[Y(t) \setminus Y(t+1), Z(t) \setminus Z(t+1)]$ for $t = 0, \ldots, W$, where $Y(W+1) = \emptyset$. Note that A^w can be computed in $O(nr^{\omega-1})$ time, since this can be obtained by Gaussian elimination (see [83]).

We now go back to the weighted matroid intersection. For $\ell = 1, 2$, let $\mathbf{M}_\ell$ be a linear matroid of rank r_ℓ on S, whose matrix representation is given by an $r_\ell \times n$ matrix A_ℓ with the same field. We also denote by R_ℓ the row set of A_ℓ for $\ell = 1, 2$. Then the following proposition is known in [84].

Proposition 3.5.4. *Two linear matroids $\mathbf{M}_1$ and $\mathbf{M}_2$ have a common base if and only if the matrix $N = -A_1 D^{-1} A_2^\top$ is nonsingular, where D is a diagonal matrix of order n such that the set of the diagonal entries is algebraically independent.*

Note that N can be computed in $O(nr^{\omega-1})$ time (see [84]).

Let us consider the procedure Unweighted_Matroid_Intersection. Given a weight-splitting w_1 and w_2 of w, recall $Z_\ell(t) = \{e \in S \mid w_\ell(e) \geq t\}$ for $t = 0, \ldots, W+1$ and $\ell = 1, 2$. Let $Y_\ell(t)$ be the set of the nonzero rows in $A_\ell^{w_\ell}[R_\ell, Z_\ell(t)]$ for $t = 0, \ldots, W$. For a subset S', let $Z'_\ell(t) = Z_\ell(t) \cap S'$.

Lemma 3.5.5. *For two linear matroids, suppose that I' is (w_1, w_2)-near-optimal in a subset S'. Then we can construct a common independent set I, in $O(nr^{\omega-1})$ time, that is simultaneously (i) a maximum-cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, and (ii) (w_1, w_2)-near-optimal in S'.*

Proof. We denote $A'_\ell = A_\ell^{w_\ell}[R_\ell, S']$, which is a matrix representation of $\mathbf{M}'_\ell | S'$, for $\ell = 1, 2$. We first show the following claim on (w_1, w_2)-near-optimality.

Claim 2. *A set J is (w_1, w_2)-near-optimal in a subset S' if and only if there exists $U_\ell \subseteq R_\ell$ ($\ell = 1, 2$) with $Y_1(1) \subseteq U_1$ and $Y_2(p+1) \subseteq U_2$, where $p = \min_{e \in S'} w_2(e)$, such that J is a common base of $A'_1[U_1, S']$ and $A'_2[U_2, S']$.*

Proof. Suppose J is (w_1, w_2)-near-optimal in S'. Then it follows from (3.1) that $J \cap Z'_1(t)$ is a base of $\mathbf{M}_1 | Z'_1(t)$ for every integer $t \geq 1$ if and only if each submatrix $A'_1[Y_1(t), J \cap Z'_1(t)]$ is nonsingular for every integer $t \geq 1$. Since $A'_1[Y_1(1), J \cap Z'_1(1)]$ is nonsingular, we can take $U_1 \subseteq R_1$ with $|U_1| = |J|$ such that $A'_1[U_1, J]$ is nonsingular and $Y_1(1) \subseteq U_1$. Similarly, there exists $U_2 \subseteq R_2$ with $|U_2| = |J|$ such that $A'_2[U_2, J]$ is nonsingular and $Y_2(p+1) \subseteq U_2$. Thus J is a common base of $A'_1[U_1, S']$ and $A'_2[U_2, S']$.

Conversely, suppose that we have row subsets U_1 and U_2 satisfying the conditions. Since $A'_1[R_1 \setminus Y_1(1), Z'_1(1)]$ is a zero matrix, the base J has a nonsingular submatrix $A'_1[Y_1(1), J \cap Z'_1(1)]$. Since the submatrix is block-diagonal, this is equivalent to that $J \cap Z'_1(t)$ is a base of $\mathbf{M}'_1 | S'$ for every $t \geq 1$. The case for A'_2 is analogous.

□

By the assumption that I' is (w_1, w_2)-near-optimal in S', there exist U_1 and U_2 such that $Y_1(1) \subseteq U_1$, $Y_2(p+1) \subseteq U_2$, and $A'_1[U_1, S']$ and $A'_2[U_2, S']$ have a common base. Among such U_1

and U_2, we take U_1^* and U_2^* with maximum size. We can find a common base I for $A'_1[U_1^*, S']$ and $A'_2[U_2^*, S']$ by Harvey's algorithm in $O(nr^{\omega-1})$ time [84]. Since I satisfies the conditions of the above claim, I is (w_1, w_2)-near-optimal with maximum size in S'. Thus I is a desired set.

It remains to show that we can find such maximum U_1^* and U_2^* in $O(nr^{\omega-1})$ time. Construct $N = -A'_1 D^{-1} A'^{\top}_2$ in $O(nr^{\omega-1})$ time, which has the row set R_1 and column set R_2. By Proposition 3.5.4, for $U_1 \subseteq R_1$ and $U_2 \subseteq R_2$, both $A'_1[U_1, S']$ and $A'_2[U_2, S']$ have a common base if and only if $N[U_1, U_2]$ is nonsingular, which follows from the fact that $N[U_1, U_2] = -A'_1[U_1, S']D^{-1}(A'_2[U_2, S'])^{\top}$. Therefore, it suffices to find $U_1^* \subseteq R_1$ and $U_2^* \subseteq R_2$ with maximum size such that $Y_1(1) \subseteq U_1^*$ and $Y_2(p+1) \subseteq U_2^*$ and $N[U_1^*, U_2^*]$ is nonsingular. This can be done in $O(r^{\omega})$ time, since the rank of N is at most r.

□

Since Update_Weight can be performed in $O(nr^{\omega-1})$ time, we can solve the weighted matroid intersection exactly in $O(Wnr^{\omega-1})$ time and approximately in $O(\epsilon^{-1}nr^{\omega-1}\log r)$ time.

Furthermore, using a preprocessing technique by Cheung, Kwok, and Lau [21], we can improve the computational time. Given a positive integer k, their algorithm reduces an $r \times n$ matrix A to an $O(k) \times n$ matrix A' such that, if a column set in A' of size at most k is independent, then it is independent in A with high probability. This can be done in $O(nr)$ time.

We simply use this algorithm where k is set to be the maximum size $r_* \leq r$ of a common independent set of $\mathbf{M}_1$ and $\mathbf{M}_2$. The size r_* can be computed in $O(nr\log r_* + nr_*^{\omega-1})$ time [21]. After we obtain two $O(r_*) \times n$ matrices by their method, apply our algorithm to obtain a maximum-weight common independent set. This takes $O(Wnr_*^{\omega-1})$ time for an exact algorithm and $O(\epsilon^{-1}nr_*^{\omega-1}\log r_*)$ time for an approximation algorithm.

Therefore, we have the following theorem.

Theorem 3.5.6. *For two linear matroids, we can solve the weighted matroid intersection exactly in $O(nr\log r_* + Wnr_*^{\omega-1})$ time and $(1-\epsilon)$-approximately in $O(nr\log r_* + \epsilon^{-1}nr_*^{\omega-1}\log r_*)$ time, where r_* is the size of a common independent set.*

It should be noted that our algorithm is simple in the sense that it involves only a constant matrix and does not need to manipulate a univariate-polynomial matrix [163], unlike the algorithms in [83, 153].

3.6 Rank-Maximal Matroid Intersection

In this section, we deal with the rank-maximal matroid intersection problem. As mentioned in the introduction, this problem can be reduced to a weighted matroid intersection problem whose weight w is drawn from $\{1, n, n^2, \ldots, n^{R-1}\}$. More generally, we consider the case where the weight w is drawn from a geometric series $\{1, u, u^2, \ldots, u^{R-1}\}$, where $u \geq 2$. Let d_k be the difference of two consecutive weights, i.e., $d_k = u^k - u^{k-1}$ for $k = R, R-1, \ldots, 1$. For convenience, we also define $d_0 = 1$.

Our algorithm for the geometric-series weight case is described as follows, where the only difference from our exact algorithm (Algorithm 1 in Section 3.3) is in the dual update Step (2-4): we update w_1 and w_2 with large weight d_k.

Algorithm 3: Exact algorithm for geometric-series weights

Input: two matroids $\mathbf{M}_1 = (S, \mathcal{I}_1)$ and $\mathbf{M}_2 = (S, \mathcal{I}_2)$, a weight function $w\colon S \to \{1, u, u^2, \cdots, u^{R-1}\}$, where $u \geq 2$.

Output: $I \in \mathcal{I}_1 \cap \mathcal{I}_2$ where I is a maximum-weight common independent set of $\mathbf{M}_1$ and $\mathbf{M}_2$.

Step 1. Set $k := R - 1$, $w_1 := 0$, $w_2 := w$, and $I' := \emptyset$.

Step 2. While $k > 0$ do the following steps.

(2-1) Set $S' := \{e \in S \mid w_2(e) \geq u^k\}$.

(2-2) Set $\mathbf{M}'_\ell := \mathbf{M}_\ell^{w_\ell}|S'$ for each $\ell = 1, 2$.

(2-3) Unweighted_Matroid_Intersection (I')
Construct I so that

(i) I is a maximum-cardinality common independent set of $\mathbf{M}'_1$ and $\mathbf{M}'_2$, and

(ii) I is (w_1, w_2)-near-optimal in S'.

(2-4) Update_Weight

(2-4-1) Let $T \subseteq S'$ be the set of elements reachable from X_2 in $G_{\mathbf{M}'_1, \mathbf{M}'_2}(I)$.

(2-4-2) For each $e \in T$, let $w_1(e) := w_1(e) + d_k$, $w_2(e) := w_2(e) - d_k$.

(2-5) Set $k := k - 1$ and $I' := I$.

Step 3. Return I.

In the following, we prove the correctness of Algorithm 3 by showing that it would have the same outcome as if we had run Algorithm 1 in Section 3.3 instead. For the purpose, we first need a technical lemma, whose proof exploits the property that each weight is at least double the previous weight in the given geometric series of weights w.

Lemma 3.6.1. *At Step* (2-1) *of Round* k $(k = R-1, R-2, \ldots, 1)$*, we have the following for* e *and* f *in* S'*.*

(i) *If* $w_1(e) \neq w_1(f)$*, then* $|w_1(e) - w_1(f)| \geq d_{k+1}$.

(ii) *If* $w_2(e) \neq w_2(f)$*, then* $|w_2(e) - w_2(f)| \geq d_{k+1}$.

Proof. We prove (i) by induction on k. When $k = R - 1$, all elements have w_1-weights 0. So (i) holds trivially. For the induction step with $k < R - 1$, if $w_1(e) \neq w_1(f)$, then at least one of them, say e, is part of S' in the last round (i.e., Round $k + 1$). To avoid confusion, the set S' in the last round is denoted by $\overline{S}'$. Also w_1 and w_2 at Step (2-1) of the last round are denoted by $\overline{w}_1$ and $\overline{w}_2$, respectively. Consider two possibilities.

Case 1. Suppose that $f \in \overline{S}'$. By induction hypothesis, either $|\overline{w}_1(e) - \overline{w}_1(f)| \geq d_{k+2}$, or $\overline{w}_1(e) = \overline{w}_1(f)$. In the former case, the difference between the w_1-weights of e and f is changed by at most d_{k+1} in the last round. Therefore, we have

$$|w_1(e) - w_1(f)| \geq d_{k+2} - d_{k+1} \geq d_{k+1},$$

where the last inequality holds because $u \geq 2$. In the latter case, either $w_1(e) = w_1(f)$ (if both $\overline{w}_1(e)$ and $\overline{w}_1(f)$ are updated or unchanged in the last round), or $|w_1(e) - w_1(f)| = d_{k+1}$ (if exactly one of them is updated).

Case 2. Suppose that $f \notin \overline{S}'$. Then $w_1(f) = 0$. If $w_1(e)$ has not been updated so far, then $w_1(e) = 0$. Otherwise, since $w_1(e)$ is increased at Round s for some $s \geq k+1$, we have $w_1(e) \geq d_s \geq d_{k+1}$. The induction step is completed.

(ii) can be proved symmetrically.

□

To avoid confusion, let i be the index of the rounds when we apply Algorithm 1 in Section 3.3, and J^i be the independent set obtained in Round i. Let k be the index of the rounds when we apply Algorithm 3, and I^k be the independent set obtained at Round k of Algorithm 3.

Lemma 3.6.2. *Define $i_k = u^k$ for $k = R-1, R-2, \ldots, 1$. For $k = R-1, R-2, \ldots, 1$ and $\ell = 1, 2$, the weights w_ℓ at Round i_k of Algorithm 1 are the same as the weights w_ℓ at Round k of Algorithm 3. Thus, for $k = R-1, R-2, \ldots, 1$, the auxiliary graph in Round k of Algorithm 3 coincides with one in Round i_k of Algorithm 1.*

Proof. We prove by induction in k. When $k = R-1$ and $i_k = u^{R-1}$, the lemma holds easily. For the induction step when $k < R-1$, we argue that the update of the w_1- and w_2-weights done in Round $k+1$ of Algorithm 3 are the same as the accumulated updates of the w_1- and w_2-weights done in Algorithm 1 from Round i_{k+1} down to Round $i_k + 1$. To be more precise, in Round $k+1$ in Algorithm 3, all elements in $S' \cap T$ have their w_2-weights decrease by the amount of d_{k+1} and their w_1-weights increase by the same amount. We show that from Round i_{k+1} down to Round $i_k + 1$ in Algorithm 1, the same set of elements have their w_2- and w_1-weights updated (and each round by the amount of one). This would prove the induction step.

For simplicity, we often denote $i_{k+1} - t$ with (t) for $t = 0, 1, \ldots, d_{k+1} - 1$. Let $G^{(t)}$ be the auxiliary graph at Round $i_{k+1} - t$ in Algorithm 1, and $T^{(t)}$ be the reachable set in $G^{(t)}$ found in Step (2-4) of Round $i_{k+1} - t$. We will show the following properties for Algorithm 1.

(1) The ground set $S^{(t)}$ is the same as $S^{(0)}$.

(2) The reachable set $T^{(t)}$ is the same as $T^{(0)}$.

(3) The independent set $J^{(t)}$ is the same as $J^{(0)}$.

To see (1), observe that, in Algorithm 1, all elements e not in $S^{(0)}$ have $w_2^{(0)}(e) \leq u^{k+1} - 1$. Since $w_2^{(0)}(e)$ is equal to $w(e)$, we see $w_2^{(0)}(e) \leq u^k$ for $e \notin S^{(0)}$ by the definition of w. Hence e is not contained in $S^{(t)}$, as $w_2^{(t)}(e) = w_2^{(0)}(e) \leq u^k < u^{k+1} - t$ for $0 \leq t \leq d_{k+1} - 1$. Therefore, $S^{(0)} = S^{(t)}$.

To see (2), we first show $T^{(t)} \supseteq T^{(0)}$. Note that, by Lemma 3.2.1(iii), for an arc ef in $G^{(0)}$, their w_1-weights $w_1^{(0)}(e)$ and $w_1^{(0)}(f)$ must be the same if $e \notin J^{(0)}$ and $f \in J^{(0)}$, and their w_2-weights $w_2^{(0)}(e)$ and $w_2^{(0)}(f)$ must be the same if $e \in J^{(0)}$ and $f \notin J^{(0)}$. Then, if both e and f are in $T^{(0)}$ or neither of them is in $T^{(0)}$, their w_2-weights (respectively w_1-weights) remain the same in the subsequent rounds, and hence the arc ef appears in $G^{(t)}$. Thus $T^{(t)} \supseteq T^{(0)}$.

To prove $T^{(t)} \subseteq T^{(0)}$, it suffices to show that, in the auxiliary graph $G^{(t)}$, there exists no new arc from an element e in $T^{(0)}$ to an element f not in $T^{(0)}$. Note that, by the definition of $T^{(0)}$, there exists no arc from e to f in $G^{(0)}$.

First suppose that $e \notin J^{(0)}$ and $f \in J^{(0)}$. Then $w_1^{(0)}(e) < w_1^{(0)}(f)$ holds if the arc ef appeared in $G^{(t)}$. It follows from the induction hypothesis of the lemma that $G^{(0)}$ coincides with $G_{\mathbf{M}_1',\mathbf{M}_2'}(I^{k+1})$ at Round $k+1$ of Algorithm 3. This implies by Lemma 3.6.1 that $w_1^{(0)}(f) - w_1^{(0)}(e) \geq d_{k+2}$. Hence, for any $t = 0, 1, \ldots, d_{k+1} - 1$, it holds that

$$\begin{aligned} w_1^{(t)}(f) - w_1^{(t)}(e) &\geq w_1^{(0)}(f) - (w_1^{(0)}(e) + t) \\ &\geq d_{k+2} - d_{k+1} + 1 > 0, \end{aligned}$$

where the last inequality follows from $u \geq 2$. Therefore, we always have $w_1^{(t)}(f) > w_1^{(t)}(e)$ for $0 \leq t \leq d_{k+1} - 1$, and thus the arc ef never appears in $G^{(t)}$. Similarly, if $e \in J^{(0)}$ and $f \notin J^{(0)}$, then $w_2^{(0)}(e) - w_2^{(0)}(f) \geq d_{k+2}$ by Lemma 3.6.1. Hence we have $w_2^{(t)}(e) - w_2^{(t)}(f) > 0$ for $t = 0, 1, \ldots, d_{k+1} - 1$. Therefore, (2) follows.

Finally, (3) follows from the fact that in each round, there is no augmentation happening, as $G^{(0)}$ has no augmenting path and by (1) and (2) neither does $G^{(t)}$. This completes the proof. □

Theorem 3.6.3. *The independent set I returned by Algorithm 3 is an optimal solution.*

Therefore, using Cunningham's algorithm for the unweighted matroid intersection problem as a subroutine, we have the following theorem.

Theorem 3.6.4. *The rank-maximal matroid intersection problem can be solved using $O(Rnr^{1.5})$ independence oracle calls.*

We note that even though the actual weights used in Algorithm 3 can be exponentially large, there is indeed no need to store them explicitly (otherwise, we incur the extra cost in time and space). We discuss in Section 3.6.1 the implementations details.

In addition, if the given two matroids are both graphic or both linear, then we can solve the problem fast.

Theorem 3.6.5. *The rank-maximal graphic matroid intersection problem can be solved in $O(R\sqrt{r}n \log r)$ time.*

Theorem 3.6.6. *The rank-maximal linear matroid intersection problem can be solved in $O(Rnr^{\omega-1})$ time.*

3.6.1 Implementation of Rank-Maximal Matroid Intersection

We discuss how to avoid storing the actual numerical values of the weights when implementing Algorithm 3. As discussed in Section 3.3, we need to draw an arc in the auxiliary graph properly and to do this, we just need to know that given any two elements e and f in S', whether $w_\ell(e)$ is larger than, equivalent to, or smaller than $w_\ell(f)$ for $\ell = 1, 2$. There is no need to know the actual values.

It is easy to see that, at Round k, we have $w_1(e) = 0 + \sum_{t=k+1}^{R-1} d_t \cdot \mathbf{1}_{e \in T^t}$ and $w_2(e) = w(e) - \sum_{t=k+1}^{R-1} d_t \cdot \mathbf{1}_{e \in T^t}$, where T^t is the reachable set found in Step (2-4) of Round t. Using this fact, Lemma 3.6.1, and the definition of d_k, we have

Lemma 3.6.7. *At Step (2-1) of Round k ($k = R-1, R-2, \ldots, 1$), we have the following for e and f in S'.*

(i) *If $w_1(e) > w_1(f)$, then, in Round $h = k-1, k-2, \ldots, 1$, $w_1(e) > w_1(f)$.*

(ii) *If $w_2(e) > w_2(f)$, then, in Round $h = k-1, k-2, \ldots, 1$, $w_2(e) > w_2(f)$.*

Below we only discuss how to compare the w_1-weight of the elements, since the w_2-weight can be handled symmetrically.

In Round k, we can divide the elements according to their weights w_1. There can be only $O(n)$ such groups, $g^k(1), g^k(2), \ldots$, ordered by their increasing weights. Inside each group $g^k(i)$, in the next round (Round $k-1$), the elements can be split into two subgroups, if a strict subset of the elements in $g^k(i)$ belongs to the reachable set T^k in Step (2-4). However, by Lemma 3.6.7, in the next round, we know that elements belonging to these two subgroups of $g^k(i)$ will still have weights w_1 smaller than the elements from the subgroups derived from $g^k(i+1), g^k(i+2), \ldots$, and larger than the elements from the subgroups derived from $g^k(i-1), g^k(i-2), \ldots$. Finally, for the elements e newly-added in Round $k-1$, we have $w_1(e) = 0$. These elements can be either added into the existing group with the smallest weight, or we can create a new group for them (and such a group necessarily has the smallest weight). The maintenance of such data structure as described can be easily done in $O(n)$ time in each round of the algorithm.

3.7 Relation to Other Algorithms

For both the weight matching and the weighted matroid intersection problem, when the largest weight is not overly large, faster algorithms have been designed to leverage this fact. The more well-known technique is that of scaling [37, 58, 64, 66, 69, 160]. In each phase, the weights are rounded to speed up the computation; the weight rounding becomes more fine-grained in each successive round. In contrast, in [90, 102, 144] and our present work, there is no rounding of weights. In each phase, only a subset of the edges (the ground set) is considered. Such subsets enlarge over the phases. We mentioned in passing that the recent result of Lee-Sidford-Wong [123] is a far department from the previous two general techniques, where a novel cutting-plane method is employed.

Since the weighted bipartite matching problem is a special case of the weighted matroid intersection problem, we can apply our exact algorithm to the special case: in fact our algorithm would behave similarly to the one by Kao et al.[102] with the same running time, though the data structures used are different.

It may be worthwhile contrasting our exact algorithm with the algorithm of Frank [54] for two general matroids. He uses the weights w_1 and w_2 to "suppress" some edges in the original auxiliary graph. It can be shown that the modified auxiliary graph in his algorithm would be identical to the auxiliary graph of our matroids defined in each round. He augments the current independent

set I repeatedly in the modified auxiliary graph, preserving the condition that I is a maximum-weight common independent set with size $|I|$. On the other hand, our algorithm only maintains the relaxed optimality condition, and dramatically augments I with the aid of unweighted matroid intersection algorithms.

We mentioned earlier that Chekuri and Quanrud [19] have further improved the running time of our approximation algorithm for general matroids. Their speeding-up is achieved by a more sophisticated weight adjustment. In particular, in Step 2 of Algorithm 2, instead of finding a maximum-cardinality common independent set as we have done (this takes $O(nr^{1.5}\tau)$ time), they only compute a common independent set whose size $(1-\epsilon)$-approximates the former (thus they only need $\tilde{O}(nr\epsilon^{-1}\tau)$ time).

Chapter 4

Fair Matchings and Related Problems

This paper first appeared in FSTTCS 2012 and its full version appears in Algorithmica 2016. It is joint-work with Kavitha Telikepalli, Kurt Mehlhorn, and Dimitrios Michail.

Let $G = (A \cup B, E)$ be a bipartite graph, where every vertex ranks its neighbors in an order of preference (with ties allowed) and let r be the worst rank used. A matching M is *fair* in G if it has maximum cardinality, subject to this, M matches the minimum number of vertices to rank r neighbors, subject to that, M matches the minimum number of vertices to rank $(r-1)$ neighbors, and so on. We show an efficient combinatorial algorithm based on LP duality to compute a fair matching in G. We also show a scaling based algorithm for the fair *b-matching* problem.

Our two algorithms can be extended to solve other profile-based matching problems. In designing our combinatorial algorithm, we show how to solve a generalized version of the minimum weighted vertex cover in bipartite graph, using a single-source shortest paths computation—this can be of independent interest.

4.1 Introduction

Let $G = (A \cup B, E)$ be a bipartite graph on n vertices and m edges, where each $u \in A \cup B$ has a list ranking its neighbors in an order of preference (ties are allowed). Such an instance is usually referred to a *stable marriage* instance with incomplete lists and ties. A matching is a collection of edges, no two of which share an endpoint.

The focus in stable marriage problems is to find matchings that are *stable*. However, there are many applications where stability is not the most critical objective: for instance, in matching students with counsellors or applicants with training posts, we cannot compromise on the size of the matching and a *fair* matching is a natural candidate for an optimal matching in such problems.

Definition 4.1.1. *A matching M is fair in $G = (A \cup B, E)$ if M has maximum cardinality, subject*

to this, M matches the minimum number of vertices to rank r neighbors, and subject to that, M matches the minimum number of vertices to rank $(r-1)$ neighbors, and so on, where r is the worst rank used in the preference lists of vertices.

The fair matching problem can be solved in polynomial time as follows: for an edge e with incident ranks i and j, let $w(e) = n^{i-1}+n^{j-1}$. It is easy to see that a maximum cardinality matching of minimum weight (under weight function w) is a fair matching in G. Such a matching can be computed via the maximum weight matching algorithm by resetting e's weight to $4n^r - n^{i-1} - n^{j-1}$, where r is the largest rank used in any preference list.

However this approach can be expensive even if we use the fastest maximum-weight bipartite matching algorithms [39, 56, 63, 138]. The running time will be $O(rmn)$ or $\tilde{O}(r^2 m\sqrt{n})$. Note that these complexities follow from the customary assumption that an arithmetic operation takes $O(r)$ time on weights of the order n^r. We present two different techniques to efficiently compute fair matchings and a generalization called fair *b-matchings*.

A combinatorial technique. Our first technique is an iterative combinatorial algorithm for the fair matching problem. The running time of this algorithm is $\tilde{O}(r^* m\sqrt{n})$ or $\tilde{O}(r^* n^\omega)$ with high probability, where r^* is the largest rank used in a fair matching and $\omega \approx 2.37$ is the exponent of matrix multiplication. This algorithm is based on linear programming duality and in each iteration i, we solve the following "dual problem" – dual to a variant of the maximum weight matching problem.

> *Generalized minimum weighted vertex cover problem.* Let $G_i = (A \cup B, E_i)$ be a bipartite graph with edge weights given by $w_i : E_i \to \{0, 1, \ldots, c\}$. Let $K_{i-1} \subseteq A \cup B$ satisfy the property that there is a matching in G that matches all $v \in K_{i-1}$. Find a cover $\{y_u^i\}_{u \in A \cup B}$ so that $\sum_{u \in A \cup B} y_u^i$ is minimized subject to (1) for each $e = (a, b)$ in E_i, we have $y_a^i + y_b^i \geq w_i(e)$, and (2) $y_u^i \geq 0$ if $u \notin K_{i-1}$.

When $K_{i-1} = \emptyset$, the above problem reduces to the standard weighted vertex cover problem. We show that the generalized minimum weighted vertex cover problem, where y_v^i for $v \in K_{i-1}$ can be negative, can be solved via a single-source shortest paths subroutine in directed graphs, by a non-trivial extension of a technique of Iri [92].

A scaling technique. Our second technique uses scaling in order to solve the fair matching problem, by the aforementioned reduction to computing a maximum weight matching using exponentially large edge weights. It starts by solving the problem when each edge weight is 0 and then iteratively solves the problem for better and better approximations of the edge weights. This technique is applicable in the more generalized problem of computing fair *b-matchings*, where each vertex has a capacity associated with it. We solve the fair b-matching problem, in time $\tilde{O}(rmn)$ and space $O(m)$, by solving the capacitated transshipment problem, while carefully maintaining "reduced costs" whose values are within polynomial bounds. Brute-force application of the fastest known minimum-cost flow algorithms would suffer from the additional cost of arithmetic and an $O(rm)$ space requirement. For instance, using [78] would result in $\tilde{O}(r^2 mn)$ running time and $O(rm)$ space.

4.1.1 Background

Fair matchings are a special case of the *profiled-based* matching problem. So far they have received little attention in the literature. Except the two pre-prints [88, 130] on which this work is based, the only work we are aware of is the Ph.D. thesis of Sng [161], where he gives an algorithm to find a fair b-matching [1] in $O(rQ \min\{m \log n, n^2\})$ time, where $Q = \sum_{v \in V} q(v)$, the sum of the capacity $q(v)$ of all vertices $v \in V$.

The first profiled-based matching problem was introduced by Irving [98] and is called "rank-maximal matching" problem. [2] This problem has been well-studied [95, 106, 131, 141].

Definition 4.1.2. *A matching M in $G = (A \cup B, E)$ is rank-maximal if M matches the maximum number of vertices to rank 1 neighbors, subject to this constraint, M matches the maximum number of vertices to rank 2 neighbors, subject to the above two constraints, M matches the maximum number of vertices to rank 3 neighbors, and so on.*

However the rank-maximal matching problem has been studied so far in a more restricted model called the the *one-sided* preference lists model. In this model, only vertices of A have preferences over neighbors while vertices in B have no preferences. Note that a problem in the one-sided preference lists model can also be modeled as a problem with two-sided preference lists by making every $b \in B$ assign rank r to every edge incident on it, where r is the worst rank in the preference lists of vertices in A.

The current fastest algorithm to compute a rank-maximal matching in the one-sided preference lists model takes time $O(\min\{r^* m\sqrt{n}, mn, r^* n^\omega\})$ [95], where r^* is the largest rank used in a rank-maximal matching. In the one-sided preference lists setting, each edge has a unique rank associated with it, thus the edge set E is partitioned into $E_1 \dot\cup E_2 \dot\cup \cdots \dot\cup E_r$ – this partition enables the problem of computing a rank-maximal matching to be reduced to computing r^* maximum cardinality matchings in certain subgraphs of G.

We show here that our fair matching algorithm can be easily modified to compute a rank-maximal matching in the two-sided preference lists model. Thus this problem can be solved in time $\tilde{O}(r^* m\sqrt{n})$ or $\tilde{O}(r^* n^\omega)$ with high probability, which almost matches its running time for the one-sided case. Another problem that our algorithm can solve is the "maximum cardinality" rank-maximal matching problem. A matching M is a *maximum cardinality rank-maximal* matching if M has maximum cardinality, and within the set of maximum cardinality matchings, M is rank-maximal.

Organization of the paper. Section 4.2.1 contains our algorithm for the generalized bipartite vertex cover problem, Section 4.2.2 has our algorithm for fair matchings, and Section 4.2.3 has the extension to rank-maximal matchings in two-sided preference lists. Section 4.3 has our scaling algorithm.

[1] Sng used the term "generous maximum matching."
[2] Irving called it "greedy matching."

4.2 Our Combinatorial Technique for fair matchings

Recall that our input here is $G = (A \cup B, E)$ and r is the worst or largest rank used in any preference list. The notion of *signature* will be useful to us in designing our algorithm. We first define edge weight functions w_i, for $1 \leq i \leq r-1$. The value $w_i(e)$, where $e = (a, b)$, is defined as follows:

$$w_i(e) = \begin{cases} 2 & \text{if both } a \text{ and } b \text{ rank each other as rank} \leq r-i \text{ neighbors} \\ 1 & \text{if exactly one of } \{a, b\} \text{ ranks the other as a rank} \leq r-i \text{ neighbor} \\ 0 & \text{otherwise} \end{cases}$$

Definition 4.2.1. *For any matching M in G, let* $\mathsf{signature}(M)$ *be* $(|M|, w_1(M), \ldots, w_{r-1}(M))$, *where* $w_i(M) = \sum_{e \in M} w_i(e)$, *for* $1 \leq i \leq r-1$.

Thus $\mathsf{signature}(M)$ is an r-tuple, where the first coordinate is the size of M, the second coordinate is the number of vertices that get matched to neighbors ranked $r-1$ or better, and so on. Let OPT denote a fair matching. Then $\mathsf{signature}(\mathsf{OPT}) \succeq \mathsf{signature}(M)$ for any matching M in G, where $\succeq$ is the lexicographic order on signatures.

In order to capture the first coordinate of $\mathsf{signature}(M)$ also via an edge weight function, let us introduce the function w_0 defined as: $w_0(e) = 1$ for all $e \in E$. Thus $|M| = w_0(M) = \sum_{e \in M} w_0(e)$. For any matching M and $0 \leq j \leq r-1$, let $\mathsf{signature}_j(M)$ denote the $(j+1)$-tuple obtained by truncating $\mathsf{signature}(M)$ to its first $j+1$ coordinates.

Definition 4.2.2. *A matching M is $(j+1)$-optimal if* $\mathsf{signature}_j(M) = \mathsf{signature}_j(\mathsf{OPT})$.

Our algorithm runs for r^* iterations, where $r^* \leq r$ is the largest index i such that $w_{i-1}(\mathsf{OPT}) > 0$. For any $j \geq 0$, in the $(j+1)$-st iteration, our algorithm solves the minimum weighted vertex cover problem in a subgraph G_j. This involves computing a maximum w_j-weight matching M_j in the graph G_j under the constraint that all vertices of a *critical* subset $K_{j-1} \subseteq A \cup B$ have to be matched. In the first iteration which corresponds to $j = 0$, we have $G_0 = G$ and $K_{-1} = \emptyset$.

The problem of computing M_j will be referred to as the primal program of the $(j+1)$-st iteration and the minimum weighted vertex cover problem becomes its dual. We will show M_j to be $(j+1)$-optimal. The problem of computing M_j can be expressed as a linear program (rather than an integer program) as the constraint matrix is totally unimodular and hence the corresponding polytope is integral. This linear program and its dual are given below. (Let $\delta(v)$ be the set of edges incident on vertex v.)

Lemma 4.2.3. *M_j and y^j are the optimal solutions to the primal and dual programs respectively, iff the following hold:*

1. *if u is unmatched in M_j (thus u has to be outside K_{j-1}), then $y^j_u = 0$;*
2. *if $e = (u, v) \in M_j$, then $y^j_u + y^j_v = w_j(e)$;*

$$\max \sum_{e \in E} w_j(e) x_e^j$$

$$\sum_{e \in \delta(v)} x_e^j \leq 1 \qquad \forall v \in A \cup B$$

$$\sum_{e \in \delta(v)} x_e^j = 1 \qquad \forall v \in K_{j-1}$$

$$x_e^j \geq 0 \qquad \forall e \text{ in } G_j.$$

$$\min \sum_{v \in V} y_v^j$$

$$y_a^j + y_b^j \geq w_j(e) \qquad \forall e = (a, b) \text{ in } G_j$$

$$y_v^j \geq 0 \qquad \forall v \in (A \cup B) \setminus K_{j-1}.$$

Proposition 4.2.3 follows from the complementary slackness conditions in the linear programming duality theorem. This suggests the following strategy once the primal and dual optimal solutions M_j and y^j are found in the $(j+1)$-st iteration.

- to prune "irrelevant" edges: if $e = (u, v)$ and $y_u^j + y_v^j > w_j(e)$, then no optimal solution of the j-th iteration primal program can contain e. So we prune such edges from G_j and let G_{j+1} denote the resulting graph. The graph G_{j+1} will be used in the next iteration.
- to grow the critical set K_{j-1}: if $y_u^j > 0$ and $u \notin K_{j-1}$, then u *has to* be matched in every optimal solution of the primal program of the $(j+1)$-st iteration. Hence u should be added to the critical set. Adding such vertices u to K_{j-1} yields the critical set K_j for the next iteration.

Below we first show how to solve the dual problem and then give the main algorithm.

4.2.1 Solving the dual problem

For any $0 \leq j \leq r-1$, let $G_j = (A \cup B, E_j)$ be the subgraph that we work with in the $(j+1)$-st iteration and let $K_{j-1} \subseteq A \cup B$ be the critical set of vertices in this iteration. Recall that for each $e \in E_j$, we have $w_j(e) \in \{0, 1, 2\}$. We now show how to solve the dual problem efficiently for a more general edge weight function, i.e., $w_j(e) \in \{0, 1, \ldots, c\}$ for each $e \in E_j$.

Let M_j be the optimal solution of the primal program (we discuss how to compute it at the end of this section). We know that M_j matches all vertices in K_{j-1}. We now describe our algorithm to solve the dual program using M_j. Our idea is built upon that of Iri [92], who solved the special case of $K_{j-1} = \emptyset$. Recall that if a vertex v is unmatched in M_j, then $v \notin K_{j-1}$.

- Add a new vertex z to A and let $A' = A \cup \{z\}$. Add an edge of weight 0 from z to each vertex in $B \setminus K_{j-1}$. For convenience, we call the edges from z to these vertices "virtual" edges. The matching M_j still remains an optimal feasible solution after this transformation. [Note that there are only $O(n)$ virtual edges.]
- Next direct all edges $e \in E_j \setminus M_j$ from A' to B and set the edge weight $d(e) = -w_j(e)$; also direct all edges in M_j from B to A' and let the edge weight $d(e) = w_j(e)$.
- Create a *source vertex* s and add a directed edge of weight 0 from s to each *unmatched* vertex in A'. See Figure 4.1.

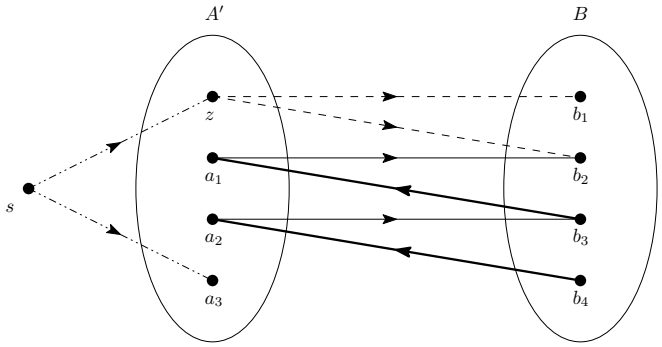

Figure 4.1: The bold edges are edges of M_j and are directed from B to A' while the edges of $E_j \setminus M_j$ are directed from A' to B.

Let $\mathcal{R}$ denote the set of all vertices in $A' \cup B$ that are reachable from s. In Figure 4.1, $\mathcal{R} = \{z, a_3, b_1, b_2\}$.

Lemma 4.2.4. *By the above transformation,*

1. $B \setminus K_{j-1} \subseteq \mathcal{R}$.
2. *There is no edge between* $A' \cap \mathcal{R}$ *and* $B \setminus \mathcal{R}$.
3. M_j *projects on to a perfect matching between* $A' \setminus \mathcal{R}$ *and* $B \setminus \mathcal{R}$.

Proof. Part (1) holds because there is a directed edge from s to z and directed edges from z to every vertex in $B \setminus K_{j-1}$. To show part (2), it is trivial to see that there can be no edge from $A' \cap \mathcal{R}$ to $B \setminus \mathcal{R}$ (by the definition of $B \setminus \mathcal{R}$). If there is an edge (b, a) from $B \setminus \mathcal{R}$ to $A' \cap R$, then this has to be an edge in M_j and hence it is a's only incoming edge. So for a to be reachable from s, it has to be the case that b is reachable from s, contradicting that $b \in B \setminus \mathcal{R}$.

For part (3), observe that if $b \in B \setminus \mathcal{R}$ is unmatched in M_j, then $b \notin K_{j-1}$ and such a vertex can be reached via z, contradicting the assumption that $b \in B \setminus \mathcal{R}$. If $a \in A' \setminus \mathcal{R}$ is unmatched in M_j, then such a vertex can be reached from s, contradicting the assumption that $a \in A' \setminus \mathcal{R}$. So all vertices in $(A' \cup B) \setminus \mathcal{R}$ are matched in M_j. By (2), a vertex $b \in B \setminus \mathcal{R}$ cannot be matched to vertices in $A' \cap \mathcal{R}$. If a vertex $a \in A' \setminus \mathcal{R}$ is matched to a vertex B in $\mathcal{R}$, then a is also in $\mathcal{R}$, a contradiction. This proves part (3). □

Note that there may exist some edges in $E_j \setminus M_j$ that are directed from $A' \setminus R$ to $B \cap \mathcal{R}$. Furthermore, some vertices of $A \setminus K_{j-1}$ can be contained in $A \setminus \mathcal{R}$. Delete all edges from $A' \setminus \mathcal{R}$ to $B \cap \mathcal{R}$ from G_j; let H_j denote the resulting graph. By Lemma 4.2.4.3, no edge of M_j has been deleted, thus M_j belongs to H_j and M_j is still an optimal matching in the graph H_j. Moreover, H_j is split into two parts: one part is $(A' \cup B) \cap \mathcal{R}$, which is isolated from the second part $(A' \cup B) \setminus \mathcal{R}$. See Figure 4.2.

Next add a directed edge from the source vertex s to each vertex in $B \setminus \mathcal{R}$. Each of these edges e has weight $d(e) = 0$. By Lemma 4.2.4.3, all vertices can be reached from s now. Also note that there can be no negative-weight cycle, otherwise, we can augment M_j along this cycle to get a

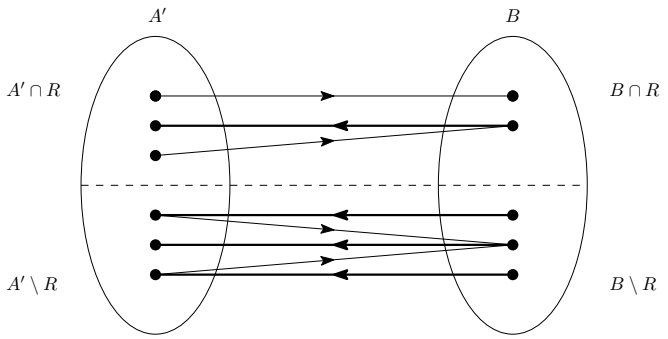

Figure 4.2: The set $A' \cup B$ in the graph H_j is split into two parts: $(A' \cup B) \cap \mathcal{R}$ and $(A' \cup B) \setminus \mathcal{R}$

matching of larger weight while still keeping the same set of vertices matched, which leads to a contradiction to the optimality of M_j.

Apply the single-source shortest paths algorithm [75, 152, 154, 174] from the source vertex s in this graph H_j where edge weights are given by $d(\cdot)$. Such algorithms take $O(m\sqrt{n})$ time or $\tilde{O}(n^\omega)$ time when the largest edge weight is $O(1)$. Let d_v be the distance label of vertex $v \in A' \cup B$.

We define an initial vertex cover as follows. If $a \in A'$, let $\tilde{y}_a := d_a$; if $b \in B$, let $\tilde{y}_b := -d_b$. (We will adjust this cover further later.)

Lemma 4.2.5. *The constructed initial vertex cover $\{\tilde{y}_v\}_{v \in A' \cup B}$ for the graph H_j satisfies the following properties:*

1. *For each vertex $v \in ((A \cup B) \cap \mathcal{R}) \setminus K_{j-1}$, $\tilde{y}_v \geq 0$.*
2. *If $v \in (A \cup B) \setminus K_{j-1}$ is unmatched in M_j, then $\tilde{y}_v = 0$.*
3. *For each edge $e = (a, b) \in H_j$, we have $\tilde{y}_a + \tilde{y}_b \geq w^j_e$.*
4. *For each edge $e = (a, b) \in M_j$, we have $\tilde{y}_a + \tilde{y}_b = w^j_e$.*

Proof. For part (1), suppose that $a \in (A \cap \mathcal{R}) \setminus K_{j-1}$ and $\tilde{y}_a < 0$. By Lemma 4.2.4.2 and the fact that all edges from $A' \setminus \mathcal{R}$ to $B \cap \mathcal{R}$ are absent, the shortest path from s to a cannot go through $(A \cup B) \setminus \mathcal{R}$. So there exists an alternating path P (of even length) starting from some *unmatched* vertex $a' \in (A' \cap \mathcal{R}) \setminus K_{j-1}$ and ending at a. The distance from a' to a along path P must be negative, since $d_a = \tilde{y}_a < 0$. Therefore,

$$\sum_{e \in M_j \cap P} w_e < \sum_{e \in P \setminus M_j} w_e.$$

Note that it is possible that the first edge $e = (a', b) \in P$ is a virtual edge, i.e., $a' = z$ and the first edge e connects z to some vertex $b \in (B \cap \mathcal{R}) \setminus K_{j-1}$. In this case, $d_e = 0$ and b is not an element of the critical set K_{j-1}. Therefore, irrespective of whether the first edge is virtual or not, we can replace the matching M_j by $M_j \oplus P$ (ignoring the first edge in P if it is virtual), thereby creating a feasible matching with larger weight than M_j, a contradiction.

So we are left to worry about the case when vertex $b \in (B \cap \mathcal{R}) \setminus K_{j-1}$. Recall that $\tilde{y}_b = -d_b$. We claim that $d_b \leq 0$. Suppose not. Then the shortest distance from s to b is strictly larger than 0. But this cannot be, since there is a path composed of edges (s, z) and (z, b), and such a path has total distance of exactly 0. This completes the proof of part (1).

To show part (2), by Lemma 4.2.4.3, an unmatched vertex must be in $\mathcal{R}$. First, assume that this unmatched vertex is $a \in (A \cap \mathcal{R}) \setminus K_{j-1}$. By our construction, there is only one path from s to a, which is simply the directed edge from s to a and its distance is 0. So $y_a = d_a = 0$. Next assume that this unmatched vertex is $b \in (B \cap \mathcal{R}) \setminus K_{j-1}$. Suppose that $\tilde{y}_b > 0$. Then $d_b = -\tilde{y}_b < 0$. By Lemma 4.2.4.2 and the fact that all edges from $A' \setminus \mathcal{R}$ to $B \cap \mathcal{R}$ have been deleted, the shortest path from s to b cannot go through $(A \cup B) \setminus \mathcal{R}$. So the shortest path from s to b must consist of the edge from s to some unmatched vertex $a \in (A' \cap \mathcal{R}) \setminus K_{j-1}$, followed by an augmenting path P (of odd length) ending at b. As in the proof of (1), we can replace M_j by $M_j \oplus P$ (irrespective of whether the first edge in P is virtual or not) so as to get a matching of larger weight while preserving the feasibility of the matching, a contradiction. This proves part (2).

For parts (3) and (4), first consider an edge $e = (a, b)$ outside M_j in H_j. Such an edge is directed from a to b. So $\tilde{y}_a - w_e^j = d_a + d(e) \geq d_b = -\tilde{y}_b$. This proves part (3). Next consider an edge $e = (a, b) \in M_j$. Such an edge is directed from b to a. Furthermore, e is the only incoming edge of a, implying that e is part of the shortest path tree rooted at s. As a result, $-\tilde{y}_b + w_e^j = d_b + d(e) = d_a = \tilde{y}_a$. This shows part (4). This completes the proof of Lemma 4.2.5. □

At this point, we possibly still do not have a valid cover for the dual program due to the following two reasons.

- Some vertex $a \in A \setminus K_{j-1}$ has $\tilde{y}_a < 0$. (However it cannot happen that some vertex $b \in B \setminus K_{j-1}$ has $\tilde{y}_b < 0$, since Lemma 4.2.4.1 states that such a vertex is in $\mathcal{R}$ and Lemma 4.2.5.1 states that $\tilde{y}_b$ must be non-negative.)
- The edges deleted from G_j (to form H_j) are not properly covered by the initial vertex cover $\{\tilde{y}_v\}_{v \in A \cup B}$.

We can remedy these two defects as follows. Define $\delta = \max\{\delta_1, \delta_2, 0\}$,

$$\text{where } \delta_1 = \max_{e=(a,b) \in E} \{w_e^j - \tilde{y}_a - \tilde{y}_b\} \quad \text{and} \quad \delta_2 = \max_{a \in A \setminus K_{j-1}} \{-\tilde{y}_a\}.$$

In $O(n + m)$ time, we can compute δ. If $\delta = 0$, the initial cover is already a valid solution to the dual program. In the following, we assume that $\delta > 0$ exists (if the initial cover is already a valid solution for the dual program, then the proof that it is also optimal is just the same as in Theorem 4.2.6.) We build the final vertex cover as follows. The proof of Theorem 4.2.6 is included in Appendix A.

1. For each vertex $u \in (A \cup B) \cap \mathcal{R}$, let $y_u = \tilde{y}_u$;
2. For each vertex $a \in A \setminus \mathcal{R}$, let $y_a = \tilde{y}_a + \delta$;
3. For each vertex $b \in B \setminus \mathcal{R}$, let $y_b = \tilde{y}_b - \delta$;

Theorem 4.2.6. *The final vertex cover $\{y_v\}_{v \in A \cup B}$ is an optimal solution for the dual program.*

Given M_j, it follows that the dual problem can be solved in time $O(m\sqrt{n})$ or $\tilde{O}(n^\omega)$. The problem of computing M_j can be solved by the following folklore technique: form a new graph $\tilde{G}_j$ by taking two copies of G_j and making the two copies of a vertex $u \notin K_{j-1}$ adjacent using an edge of weight 0. A maximum weight *perfect* matching in $\tilde{G}_j$ yields a maximum weight matching in G_j that matches all vertices in K_{j-1}, i.e., an optimal solution to the primal program of the j-th iteration. Since $c = O(1)$, a maximum weight perfect matching in $\tilde{G}_j$ can be found in $O(m\sqrt{n}\log n)$ time by the fastest bipartite matching algorithms [39, 56, 63, 138], or in $\tilde{O}(n^\omega)$ time with high probability by Sankowski's algorithm [154].

4.2.2 Our main algorithm

We now present our algorithm to compute a fair matching. Recall that r is the worst rank in the problem instance and r^* is the worst rank in a fair matching. We first present an algorithm that runs for r iterations and we show later in this section how to terminate our algorithm in r^* iterations.

1. *Initialization.* Let $G_0 = G$ and $K_{-1} = \emptyset$.
2. For $j = 0$ to $r - 1$ do
   (a) Find the optimal solution $\{y_u^j\}_{u \in A \cup B}$ to the dual program of the $(j+1)$-st iteration.
   (b) Delete from G_j every edge (a, b) such that $y_a^j + y_b^j > w_j(e)$. Call this subgraph G_{j+1}.
   (c) Add all vertices with positive dual values to the critical set, i.e., $K_j = K_{j-1} \cup \{u\}_{y_u^j > 0}$.
3. Return the optimal solution to the primal program of the last iteration.

The solution returned by our algorithm is a maximum (w_{r-1})-weight matching in the graph G_{r-1} that matches all vertices in K_{r-2}. By Proposition 4.2.3, this is, in fact, a matching in the subgraph G_r that matches all vertices in K_{r-1}. Lemma 4.2.8 proves the correctness of our algorithm. Lemma 4.2.7 (proof in Appendix A) guarantees that our algorithm is never "stuck" in any iteration due to the infeasibility of the primal or dual problem.

Lemma 4.2.7. *The primal and dual programs of the $(j+1)$-st iteration are feasible, for $0 \leq j \leq r - 1$.*

Lemma 4.2.8. *For every $0 \leq j \leq r - 1$, the following hold:*

1. *any matching M in G_j that matches all $v \in K_{j-1}$ is j-optimal;*
2. *conversely, a j-optimal matching in G is a matching in G_j that matches all $v \in K_{j-1}$.*

Proof. We proceed by induction. The base case is $j = 0$. As $K_{-1} = \emptyset$, $G_0 = G$, and all matchings are, by definition, 0-optimal, the lemma holds vacuously.

For the induction step $j \geq 1$, suppose that the lemma holds up to $j - 1$. As $K_{j-1} \supseteq K_{j-2}$ and G_j is a subgraph of G_{j-1}, M is a matching in G_{j-1} that matches all vertices of K_{j-2}. Thus

by induction hypothesis, M is $(j-1)$-optimal. For each edge $e=(a,b)\in M$ to be present in G_j, e must be a tight edge in the j-th iteration, i.e., $y_a^{j-1}+y_b^{j-1}=w_{j-1}(e)$. Furthermore, as $K_{j-1}\supseteq\{u\}_{y_u^{j-1}>0}$, we have

$$w_{j-1}(M)=\sum_{e=(a,b)\in M} w_{j-1}(e)=\sum_{e=(a,b)\in M} y_a^{j-1}+y_b^{j-1}\geq\sum_{u\in A\cup B} y_u^{j-1},$$

where the final inequality holds because all vertices v with positive y_v^{j-1} are matched in M. By linear programming duality, M must be optimal in the primal program of the j-th iteration. So the j-th primal program has optimal solution of value $w_{j-1}(M)$.

Recall that by definition, OPT is also $(j-1)$-optimal. By (2) of the induction hypothesis, OPT is a matching in G_{j-1} and OPT matches all vertices in K_{j-2}. So OPT is a feasible solution of the primal program in the j-th iteration. Thus $w_{j-1}(\mathsf{OPT})\leq w_{j-1}(M)$. However, it cannot happen that $w_{j-1}(\mathsf{OPT})<w_{j-1}(M)$, otherwise, $\mathsf{signature}(M)\succ\mathsf{signature}(\mathsf{OPT})$, since both OPT and M have the same first $j-1$ coordinates in their signatures. So we conclude that $w_{j-1}(\mathsf{OPT})=w_{j-1}(M)$, and this implies that M is j-optimal as well. This proves (1).

In order to show (2), let M' be a j-optimal matching in G. Since M' is j-optimal, it is also $(j-1)$-optimal and by (2) of the induction hypothesis, it is a matching in G_{j-1} that matches all vertices in K_{j-2}. So M' is a feasible solution to the primal program of the j-th iteration. As $\mathsf{signature}(M')$ has $w_{j-1}(\mathsf{OPT})$ in its j-th coordinate, M' must be an optimal solution to this primal program; otherwise there is a j-optimal matching with a value larger than $w_{j-1}(\mathsf{OPT})$ in the j-th coordinate of its signature, contradicting the optimality of OPT. By Proposition 4.2.3.2, all edges of M' are present in G_j and by Proposition 4.2.3.1, all vertices $u\notin K_{j-2}$ with $y_u^{j-1}>0$, in other words, all vertices in $K_{j-1}\setminus K_{j-2}$ have to be matched by the optimal solution M'. This completes the proof of (2). □

Since our algorithm returns a matching in G_r that matches all vertices in K_{r-1}, we know from Lemma 4.2.8.1 that this matching is r-optimal, thus the matching returned is fair. As mentioned earlier, our algorithm can be modified so that it terminates in r^* iterations. For that, we need to know the value of r^*.

We continue to use the weight function $w_0:E\to\{1\}$, however instead of $w_1,\ldots,w_r$, we should use the weight functions $\tilde{w}_1,\ldots,\tilde{w}_{r^*-1}$ where for $1\leq i\leq r^*-1$, $\tilde{w}_i$ is defined as: for any edge $e=(a,b)$, $\tilde{w}_i(e)$ is 2 if both a and b rank each other as rank $\leq r^*-i+1$ neighbors, it is 1 if exactly one of $\{a,b\}$ ranks the other as a rank $\leq r^*-i+1$ neighbor, otherwise it is 0. The value r^* can be easily computed right at the start of our algorithm as follows.

- Let M^* be a maximum cardinality matching in G. The value r^* is the smallest index j such that the subgraph $\bar{G}_j$ admits a matching of size $|M^*|$, where $\bar{G}_j$ is obtained by deleting all edges $e=(a,b)$ from G where either a or b (or both) ranks the other as a rank $>j$ neighbor.

- We compute r^* by first computing M^* and then computing a maximum cardinality matching in $\bar{G}_1,\bar{G}_2,\ldots$ and so on till we see a subgraph $\bar{G}_j$ that admits a matching of size $|M^*|$. This index $j=r^*$ and it can be found in $O(r^*m\sqrt{n})$ time [**?**] or in $O(r^*n^\omega)$ time [84, 132].

We now bound the running time of our algorithm. We showed how to solve the dual program in $O(m\sqrt{n})$ time once we have the solution to the primal program and we have seen that the primal program can be solved in $O(m\sqrt{n}\log n)$ time. Alternatively, both the primal and dual problems can be solved in $\tilde{O}(n^\omega)$ time with high probability. Theorem 4.2.9 follows.

Theorem 4.2.9. *A fair matching M in $G = (A \cup B, E)$ can be computed in $\tilde{O}(r^* m\sqrt{n})$ time, or in $\tilde{O}(r^* n^\omega)$ time with high probability, where r^* is the largest rank incident on an edge in M, $n = |A \cup B|$, $m = |E|$, and $\omega \approx 2.37$ is the exponent of matrix multiplication.*

4.2.3 Two-sided rank-maximality

In this section we show how our fair matching algorithm can be modified to solve the rank-maximal matching problem and the maximum cardinality rank-maximal matching problem in the domain of two-sided preference lists. For both these problems, we use the same algorithm that we designed for fair matchings – the only difference is in how the edge weight functions $w_0, \ldots, w_{r-1}$ are defined.

Consider the maximum cardinality rank-maximal matching problem. Here we use the same function w_0 that we used in the fair matching algorithm, i.e., $w_0(e) = 1$ for all $e \in E$. For $1 \leq i \leq r-1$ and $e = (a, b)$, $w_i(e) = 2$ if both a and b rank each other as rank i neighbors, it is 1 if exactly one of $\{a, b\}$ ranks the other as a rank i neighbor, otherwise it is 0.

It is easy to see that the matching OPT under the above weight functions is a maximum cardinality rank-maximal matching in G. When the algorithm given in Section 4.2.2 is run using these edge weight functions, the matching returned has signature $\mathsf{signature}(\mathsf{OPT})$ (by Lemmas 4.2.7 and 4.2.8), in other words, it is a maximum cardinality rank-maximal matching. This algorithm can again be made in terminate in r^* iterations (Appendix A has the details).

In the rank-maximal matching problem, we define the functions $w_0, \ldots, w_{r-1}$ from E to $\{0, 1, 2\}$ as follows. For $0 \leq i \leq r-1$ and $e = (a, b)$, $w_i(e) = 2$ if both a and b rank each other as rank $i+1$ neighbors, it is 1 if exactly one of $\{a, b\}$ ranks the other as a rank $i+1$ neighbor, otherwise it is 0. Using these edge weight functions and our algorithm from Section 4.2.2, we get a rank-maximal matching in the two-sided preference lists model. Again, this algorithm can be made in terminate in r^* iterations (via the same details as in the case above – see Appendix A). We conclude this section with the following theorem.

Theorem 4.2.10. *A rank-maximal/maximum cardinality rank-maximal in $G = (A \cup B, E)$ with two-sided preference lists, can be computed in $\tilde{O}(r^* m\sqrt{n})$ time, or in $\tilde{O}(r^* n^\omega)$ time with high probability, where r^* is the largest rank used in such a matching.*

4.3 The fair b-matching problem: our scaling technique

The fair matching problem can be generalized by introducing capacities on the vertices. We are given $G = (A \cup B, E)$ as before, along with the capacity function $q : V \to \mathcal{S}_{>0}$. What we seek is a subset E' of E where each vertex $v \in A \cup B$ is incident to at most $q(v)$ edges in E'. Such a subset E' is a *b-matching*. Our goal here is to find a fair b-matching, i.e., a b-matching M which has the largest possible size, subject to this constraint, M matches the minimum number of vertices to their rank r neighbors, and so on.

The fair b-matching problem can be reduced to the minimum-cost circulation problem as follows. Add two additional vertices s and t. For each vertex $a \in A$, add an edge (s, a) with capacity $q(a)$ and cost zero; for each vertex $b \in B$, add an edge (b, t) with capacity $q(b)$ and cost zero. Every edge (a, b) where $a \in A, b \in B$ has capacity one and is directed from A to B. If the incident ranks on edge e are i and j, then e will be assigned a cost of $-(4n^r - n^{i-1} - n^{j-1})$. The resulting instance has a trivial upper bound of $n^2/4$ on the maximum s-t flow. We also add an edge from t to s with zero cost and capacity larger than the $n^2/4$ upper bound. It is easy to verify that a minimum-cost circulation yields a fair b-matching.

We note however, that the above reduction involves costs that are exponential in the size of the original problem. We now present a general technique in order to handle these huge costs – we focus on solving the *capacitated transshipment* version of the minimum-cost flow problem [**?**]. Let $G = (V, E)$ be a directed network with a cost $c : E \to \mathcal{S}$ and capacity $u : E \to \mathcal{S}_{\geq 0}$ associated with each edge. With each $v \in V$ a real number $b(v)$ is associated, where $\sum_{v \in V} b(v) = 0$. If $b(v) > 0$, then v is a supply node, and if $b(v) < 0$, then v is a demand node. We assume G to be *symmetric*, i.e., $e \in E$ implies that the reverse arc $e^R \in E$. The reversed edges are added in the initialization step. The cost and capacity functions satisfy $c(e) = -c(e^R)$ for each $e \in E$, $u(e) \geq 0$ for the original edges and $u(e^R) = 0$ for the additional edges. From now on, E denotes the set of original and artificial edges.

A *pseudoflow* is a function $x : E \to \mathcal{S}$ satisfying the *capacity* and *antisymmetry* constraints: for each $e \in E$, $x(e) \leq u(e)$ and $x(e) = -x(e^R)$. This implies $x(e) \geq 0$ for the original edges. For a pseudoflow x and a node v, the *imbalance* $imb_x(v)$ is defined as $imb_x(v) = \sum_{(w,v) \in E} x(w, v) + b(v)$. A flow is a pseudoflow x such that, $imb_x(v) = 0$ for all $v \in V$. The cost of a pseudoflow x is $cost(x) = \sum_{e \in E} c(e)x(e)$. The minimum-cost flow problem asks for a flow of minimum cost.

For a given flow x, the residual capacity of $e \in E$ is $u_x(e) = u(e) - x(e)$. The residual graph $G(x) = (V, E(x))$ is the graph induced by edges with positive residual capacity. A *potential* function is a function $\pi : V \to \mathcal{S}$. For a potential function π, the *reduced cost* of an edge $e = (v, w)$ is $c^\pi(v, w) = c(v, w) + \pi(v) - \pi(w)$. A flow x is optimal if and only if there exists a potential function π such that $c^\pi(e) \geq 0$ for all residual graph edges $e \in E(x)$. For a constant $\varepsilon \geq 0$ a flow is ε-optimal if $c^\pi(e) \geq -\varepsilon$ for all $e \in E(x)$ for some potential function π. Consider an ε-optimal flow x and any original edge e. If $c^\pi(e) < -\varepsilon$, the residual capacity of e must be zero and hence e is saturated, i.e., $x(e) = u(e)$. If $c^\pi(e) > \varepsilon$, we have $c^\pi(e^R) = -c^\pi(e) < -\varepsilon$ and hence the residual capacity of e^R must be zero. Thus e^R is saturated, i.e., $x(e^R) = u(e^R) = 0$. So e is unused.

We are now ready to describe our scaling algorithm, which is presented in a concise form in Figure 4.3. The details are given in Appendix B. We conclude this section with Theorem 4.3.1, which follows from the edge cost values used in our reduction.

Theorem 4.3.1. *Given $G = (A \cup B, E)$ and a capacity function $q : A \cup B \to \mathcal{S}_{>0}$, the fair b-matching problem can be solved in time $O(rmn \log(n^2/m) \log n)$ using space $O(m)$.*

1. *Reduction.*
   (a) Add two additional vertices s and t. For each vertex $a \in A$, add an edge (s, a) with capacity $q(a)$ and cost zero; for each vertex $b \in B$, add an edge (b, t) with capacity $q(b)$ and cost zero. Add an edge from t to s with zero cost and capacity larger than $n^2/4$.
   (b) Direct any edge (a, b) where $a \in A$ and $b \in B$ from A to B, set its capacity to one and cost to $-(4n^r - n^{i-1} - n^{j-1})$.
   (c) Set the demand/supply values of all vertices to zero. Add, if required, additional edges to ensure that G is symmetric.
2. *Initialization Phase.*
   (a) Multiply all edge costs by $2^{1+\lceil \log n \rceil}$ to make them divisible by the same amount.
   (b) Let $K = \lceil \log C \rceil$ where C is the magnitude of the largest edge cost and let $\mathcal{E}_i$, $1 \leq i \leq K$ denote the set of all edges having a 1 in the i-th bit of their cost.
   (c) Initialize x_0 to any feasible flow and reduced cost $c_0(e) = 0$ for any $e \in E$.
3. *Scaling Phase.* For $i = 1$ to K do
   (a) Let $\tilde{c}_i(e) = 2c_{i-1}(e) + (1 \text{ if } e \in \mathcal{E}_i \text{ else } 0) \times sign(e)$, where $sign(e) = \pm 1$ depending on the sign of the original cost $c(e)$. The flow x_{i-1} is 3-optimal with respect to the cost function $\tilde{c}_i$ and the zero potential function, i.e., the potential of all the vertices is 0.
   (b) Use the results of [**?**] with input (i) the flow x_{i-1}, (ii) $\tilde{c}_i$ as the edge cost function and (iii) the zero potential function, to compute a 1-optimal flow and a potential function $\tilde{\pi}$ which proves the 1-optimality. Let x_i be this flow.
   (c) Compute new reduced costs as $c_i(u, v) = \tilde{c}_i(u, v) + \tilde{\pi}(u) - \tilde{\pi}(v)$.
   (d) Let d be the constant from Lemma 4.3.3 (in Appendix B). If any edge $e \in E$ has $|c_i(e)| > d \cdot n + 1$ fix it to empty or saturated by removing it (and its reversal) from the graph and modifying the imbalances of both its endpoints accordingly.
4. Return the b-matching induced by the flow x_K and any flow on edges which were fixed to either empty or saturated.

Figure 4.3: The scaling algorithm for the fair b-matching problem.

Appendix A: Missing proofs from Section 4.2

Proof of Theorem 4.2.6. We first argue that $\{y_v\}_{v \in A \cup B}$ is a feasible dual solution. By Lemma 4.2.5.1 and the choice of δ, all vertices $a \in A \setminus K_{j-1}$ have $y_a \geq 0$. By Lemma 4.2.4.1 and Lemma 4.2.5.1, all vertices $b \in B \setminus K_{j-1}$ have $y_b \geq 0$. Also by Lemma 4.2.4.2 and Lemma 4.2.5.3, and the choice of δ, all edges in E_j are properly covered. So $\{y_v\}_{v \in A \cup B}$ is feasible. We have:

$$\begin{aligned} w_j(M_j) = \sum_{e \in M_j} w_e^j &= \sum_{e=(a,b) \in M_j,\, b \in \mathcal{R}} \tilde{y}_a + \tilde{y}_b \;+ \sum_{e=(a,b) \in M_j,\, b \notin \mathcal{R}} (\tilde{y}_a + \delta) + (\tilde{y}_b - \delta) \\ &= \sum_{e=(a,b) \in M_j} y_a + y_b \quad \geq \quad \sum_{u \in A \cup B} y_u, \end{aligned}$$

where the last inequality holds because if a vertex u is unmatched, Lemma 4.2.5.2 states that $\tilde{y}_u = 0$ and since u must be in $\mathcal{R}$, we have $y_u = \tilde{y}_u = 0$. Now by the linear programming duality theorem, we conclude that the cover $\{y_v\}_{v \in A \cup B}$ is optimal. □

Proof of Lemma 4.2.7. By linear programming duality, if the primal program of the $(j+1)$-st iteration is bounded from above and admits a feasible solution, then there is also a feasible solution for its dual. It is obvious that the primal program is bounded from above since it is upper bounded by $\sum_{e \in E} w_j(e) \leq 2m$. Therefore, to prove this lemma, we just need to show the feasibility of the primal program.

The base case is $j = 0$. Since $K_{-1} = \emptyset$, any matching in $G_0 = G$ is a feasible solution for the first primal program. For $j \geq 1$, we need to show that the primal program of the $(j+1)$-st iteration is feasible. By induction hypothesis, assume that the primal program of the j-th iteration is feasible. Let M_{j-1} denote its optimal solution. Since M_{j-1} is a feasible point of the primal program of the j-th iteration, M_{j-1} uses only edges in G_{j-1} and matches all vertices in K_{j-2}. Since M_{j-1} is, in fact, an *optimal* solution to the primal program of the j-th iteration, we will show that M_{j-1} has to be a feasible point of the primal program of the $(j+1)$-st iteration by arguing that M_{j-1} does not use any of the edges pruned from G_{j-1} and all vertices in K_{j-1} are matched in M_{j-1}.

In step 2(c) of the j-th iteration, we remove only those edges $e = (a, b)$ such that $y_a^{j-1} + y_b^{j-1} > w_{j-1}(e)$ from G_{j-1} to form G_j. By the optimality of M_{j-1}, we know from Proposition 4.2.3.2 that M_{j-1} has no *slack* edges, thus all edges in M_{j-1} are retained in G_j.

We also know from Proposition 4.2.3.1 that if $y_u^{j-1} > 0$, then u must be matched in M_{j-1}. Therefore, all vertices in $K_{j-1} \backslash K_{j-2}$ are matched in M_{j-1}. Moreover, as M_{j-1} is also a feasible solution in the j-th primal program, all vertices in K_{j-2} are matched in M_{j-1}. This completes the proof of Lemma 4.2.7. □

Two-sided rank-maximal matchings: terminating in r^* iterations

In the problems of computing rank-maximal matchings and maximum cardinality rank-maximal matchings (from Section 4.2.3), we can terminate our algorithm from Section 4.2.2 in r^* iterations by inserting the following step at the end of Step 2 in every iteration. Also, this new step has the same complexity as the primal program.

> Step 2(d) (in the $(j+1)$-st iteration). Define the graph G'_{j+1} as follows: the edge set is exactly the same as in G_{j+1}; however the edge weight function w'_j is as follows. *For each edge e in G'_{j+1}: set $w'_j(e) = 1$ if $\sum_{i=j}^{r} w_i(e) > 0$, else set $w'_j(e) = 0$.*
>
> – Now find a maximum w'_j-weight matching M'_{j+1} in G'_{j+1} that matches all vertices in K_j. If the weight of M'_{j+1} is 0, then return M_j as the final solution, where M_j is the primal optimal solution of the $(j+1)$-st iteration.

Lemma 4.3.2. *The following claims hold:*

(1) For any $j \geq 0$, if the matching M'_{j+1} has weight 0, then M_j is a fair matching.

(2) If r^* is the largest rank incident on an edge in OPT*, then M'_{r^*} has weight 0.*

Proof. We first show (1). Suppose $w'_j(M'_{j+1}) = 0$. Since all edge weights are non-negative, we claim that any matching M in G_{j+1} that matches all vertices in K_j must have $w_j(M) = w_{j+1}(M) = \cdots = w_{r-1}(M) = 0$, otherwise, $w'_j(M) > 0$, a contradiction to the assumption that the maximum weight matching M'_{j+1} in G'_{j+1} satisfies $w'_j(M'_{j+1}) = 0$.

As a result, both M_j and OPT have 0 in the last $r - j$ coordinates in their signatures, since M_j (by its optimality in G_j) and OPT (by Lemma 4.2.8.2) are matchings in G_j that match all vertices of K_{j-1}. By Lemma 4.2.8.1, M_j is $(j+1)$-optimal. Hence $\mathsf{signature}(M_j) = \mathsf{signature}(\mathsf{OPT})$, thus M_j is fair and (1) is proved.

We prove (2) by contradiction. Suppose r^* is the largest rank on an edge incident in OPT and we have $w'_{r^*-1}(M'_{r^*}) > 0$. Since M'_{r^*} is a matching in G_{r^*} that matches all vertices in K_{r^*-1}, it follows from Lemma 4.2.8.1 that M'_{r^*} is r^*-optimal. While OPT has only 0 in its last $r - r^*$ coordinates, M'_{r^*} has some positive values in its last $r - r^*$ coordinates; so we have $M'_{r^*} \succ \mathsf{OPT}$. This is a contradiction to the optimality of OPT. □

Appendix B: Missing details from Section 4.3

We now give the details of our algorithm from Section 4.3, which was presented in Figure 4.3. Goldberg and Tarjan [**?**] gave a scaling algorithm for the capacitated transshipment problem. It scales the costs and runs in $O(\log(nC))$ phases where C is the magnitude of the largest edge cost. We assume that capacities, demands and supplies are polynomially bounded in n, but costs may be exponential in the input size. We modify the algorithm of Goldberg and Tarjan so that all numbers handled by the algorithm are polynomially bounded. The main ideas are to maintain node potentials *implicitly* by storing only reduced costs and to fix the flow across edges of large (positive or negative) reduced cost. Fixing flow or node potentials has been previously used in network problems in order to obtain strongly polynomial algorithms [**?** , page 397].

First we assume that all costs are divisible by $2^{1+\lceil \log n \rceil}$. In case they are not, we can always multiply them by $2^{1+\lceil \log n \rceil}$. This guarantees that a 1-optimal flow is optimal. We refer the reader to [**?**] for more details. Let $K = \lceil \log C \rceil$ where C is the magnitude of the largest edge cost. We assume that each cost is a signed integer $\pm b_1 b_2 \ldots b_K$, where b_1 is the most significant bit. For $0 \leq i \leq K$, let $c_i(e) = \lfloor c(e)/2^{K-i} \rfloor$ be the cost obtained by considering only the first i-bits of the binary representation of $c(e)$ when viewed as numbers with K bits.

Our scaling algorithm works in phases; in phase i it computes a 1-optimal flow with respect to the cost function c_i and some potential function π_i. During the algorithm we store explicitly *only* the reduced costs and not the potential function. Nevertheless, we use π_i to refer to the potential function in phase i. In phase zero we set the cost of every edge to zero and calculate any feasible flow x_0. The zero potential function on the vertices, i.e., the potential of all the vertices is 0, guarantees the 1-optimality of x_0. We therefore set the reduced cost $c_0^{\pi_0}(e) = 0$ for any $e \in E$.

Consider next any phase i with $i \geq 1$. At the beginning of the phase, we have a 1-optimal flow x_{i-1} and a reduced cost function $c_{i-1}^{\pi_{i-1}}$. We scale up by multiplying the potentials and the edge costs by 2 and adding ± 1 (after multiplying by 2) to the edge costs of edges in $\mathcal{E}_i$ (we either add

or subtract based on the sign of $c(e)$). The reduced costs scale up as follows:

$$\tilde{c}_i(e) = 2c_{i-1}^{\pi_{i-1}}(e) + (1 \text{ if } e \in \mathcal{E}_i \text{ else } 0) \times sign(e) \tag{4.1}$$

where $\mathcal{E}_i$ is the set of edges having a one in the i-th bit of their cost and $sign(e)$ is 1 or -1 depending on the sign of the original cost $c(e)$ of e.

At the end of phase $i-1$, the flow x_{i-1} is 1-optimal and therefore $c_{i-1}^{\pi_{i-1}}(e) \geq -1$ for any edge e with positive residual capacity. After scaling up, Equation (4.1) gives us that any edge with positive residual capacity has $\tilde{c}_i(e) \geq -3$. This is equivalent to saying that x_{i-1} is 3-optimal with respect to the edge costs $\tilde{c}_i$ and the zero potential function on the vertices, i.e., the potential function which is zero for all vertices. We use a variant of the algorithm of Goldberg and Tarjan [**?**] to transform the 3-optimal flow into a 1-optimal flow.

Lemma 4.3.3 (from [**?**], see also [**?**])**.** *Given the edge cost function $\tilde{c}_i$ and a 3-optimal flow x_{i-1} with respect to the zero potential function, in time $O(mn \log(n^2/m))$ one can compute a flow x_i and a potential function $\tilde{\pi}$ such that x_i is 1-optimal with respect to the potential function $\tilde{\pi}$. Potentials are only decreased, starting from zero, during the computation and $\tilde{\pi}(v) \geq -d \cdot n$ for some constant d and all v.*

At the end of the phase, we recompute the reduced costs as $c_i^{\pi_i}(u,v) = \tilde{c}_i(u,v) + \tilde{\pi}(u) - \tilde{\pi}(v)$. Recall that we do not store the potential function π_i which at this point would be $\pi_i(v) = 2\pi_{i-1}(v) + \tilde{\pi}(v)$. We next show that edges of large reduced cost can be discarded; more precisely, we can fix their flow to either zero or the capacity.

Handling large reduced costs. We say that the reduced cost of an edge e is *large positive* from phase i on, if $c_j^{\pi_j}(e) > 1$ at the end of every phase $j \geq i$, and is *large negative* from phase i on, if $c_j^{\pi_j}(e) < -1$ at the end of every phase $j \geq i$. If e is large positive from phase i on, we have $x_j(e) = 0$ for all $j \geq i$ and if e is large negative from phase i on, we have $x_j(e) = u(e)$ for all $j \geq i$.

Lemma 4.3.4. *Let $e = (u,v)$ and d be the constant from Lemma 4.3.3. If $\tilde{c}_i(e) > 2dn + 1$ at the beginning of phase i, e is large positive from phase i on, and if $\tilde{c}_i(e) < -2dn - 1$ at the beginning of phase i, e is large negative from phase i on.*

Proof. Assume $\tilde{c}_i(e) > 2dn+1$ at the beginning of some phase i. Then, at the end of phase i we have $c_i^{\pi_i}(e) = \tilde{c}_i(u,v) + \tilde{\pi}(u) - \tilde{\pi}(v) > dn + 1$ since we know from Lemma 4.3.3 that $-dn \leq \tilde{\pi}(v) \leq 0$ for all $v \in V$. Therefore, at the beginning of the next phase $\tilde{c}_{i+1}(e) = 2c_i^{\pi_i}(e) + (1 \text{ if } e \in \mathcal{E}_{i+1} \text{ else } 0) \times sign(e) > 2dn + 2 - 1 = 2dn + 1$. Thus, e is large positive from phase i onwards.

Assume next that $\tilde{c}_i(e) < -dn - 1$ at the beginning of some phase i. Then, at the end of phase i we have $c_i^{\pi_i}(e) = \tilde{c}_i(u,v) + \tilde{\pi}(u) - \tilde{\pi}(v) < -dn - 1$ since we know from Lemma 4.3.3 that $-dn \leq \tilde{\pi}(v) \leq 0$ for all $v \in V$. At the beginning of the next phase $\tilde{c}_{i+1}(e) = 2c_i^{\pi_i}(e) + (1 \text{ if } e \in \mathcal{E}_{i+1} \text{ else } 0) \times sign(e) < -2dn - 2 + 1 = -2dn - 1$. Thus, e is large negative from phase i onwards. □

Lemmas 4.3.3 and 4.3.4 allow us to fix the flow across an edge e once $|\tilde{c}_i(e)| > dn+1$. We remove the edge e (and its reversal e^R) from the graph and modify the imbalances of both endpoints accordingly, by adjusting supply and demand values. Thus, all reduced costs can be bounded by

$O(n)$ and all arithmetic operations involve polynomially bounded numbers. Theorem 4.3.5 now follows.

Theorem 4.3.5. *The minimum cost flow problem with polynomially bounded demands can be solved in* $O(T + mn \log(n^2/m) \log nC)$ *time and* $O(S + m)$ *space using arithmetic only on numbers of size polynomial in* n*, where* C *is the magnitude of the largest edge cost and* T *and* S *are the time and space bounds of an algorithm to sequentially enumerate the sets* $\mathcal{E}_i$*,* $1 \leq i \leq \lceil \log C \rceil$*, of all edges having a 1 in the* i*-th bit of their weight.*

Chapter 5

Streaming Algorithms for Maximizing Monotone Submodular Functions under a Knapsack Constraint

This paper appeared in APPROX 2017. It is joint-work with Naonori Kakimura and Yuichi Yoshida.

Abstract In this paper, we consider the problem of maximizing a monotone submodular function subject to a knapsack constraint in the streaming setting. In particular, the elements arrive sequentially and at any point of time, the algorithm has access only to a small fraction of the data stored in primary memory. For this problem, we propose a $(0.363 - \varepsilon)$-approximation algorithm, requiring only a single pass through the data; moreover, we propose a $(0.4 - \varepsilon)$-approximation algorithm requiring a constant number of passes through the data. The required memory space of both algorithms depends only on the size of the knapsack capacity and ε.

5.1 Introduction

A set function $f : 2^E \to \mathbb{R}_+$ on a ground set E is called *submodular* if it satisfies the *diminishing marginal return property*, i.e., for any subsets $S \subseteq T \subsetneq E$ and $e \in E \setminus T$, we have

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T).$$

A function is *monotone* if $f(S) \leq f(T)$ for any $S \subseteq T$. Submodular functions play a fundamental role in combinatorial optimization, as they capture rank functions of matroids, edge cuts of graphs, and set coverage, just to name a few examples. Besides their theoretical interests, submodular

functions have attracted much attention from the machine learning community because they can model various practical problems such as online advertising [7, 109, 162], sensor location [112], text summarization [125, 126], and maximum entropy sampling [120].

Many of the aforementioned applications can be formulated as the maximization of a monotone submodular function under a knapsack constraint. In this problem, we are given a monotone submodular function $f : 2^E \to \mathbb{R}_+$, a size function $c : E \to \mathbb{N}$, and an integer $K \in \mathbb{N}$, where $\mathbb{N}$ denotes the set of positive integers. The problem is defined as

$$\text{maximize} \quad f(S) \quad \text{subject to} \quad c(S) \leq K, \tag{5.1}$$

where we denote $c(S) = \sum_{e \in S} c(e)$ for a subset $S \subseteq E$. Throughout this paper, we assume that every item $e \in E$ satisfies $c(e) \leq K$ as otherwise we can simply discard it. Note that, when $c(e) = 1$ for every item $e \in E$, the constraint coincides with a cardinality constraint.

The problem of maximizing a monotone submodular function under a knapsack constraint is classical and well-studied. First introduced by Wolsey [172], the problem is known to be NP-hard but can be approximated within the factor of (close to) $1 - 1/e$; see e.g., [10, 20, 52, 116, 164].

In some applications, the amount of input data is much larger than the main memory capacity of individual computers. In such a case, we need to process data in a *streaming* fashion. That is, we consider the situation where each item in the ground set E arrives sequentially, and we are allowed to keep only a small number of the items in memory at any point. This setting effectively rules out most of the techniques in the literature, as they typically require random access to the data. In this work, we also assume that the function oracle of f is available at any point of the process. Such an assumption is standard in the submodular function literature and in the context of streaming setting [9, 18, 173]. Badanidiyuru *et al.* [9] discuss several interesting and useful functions where the oracle can be implemented using a small subset of the entire ground set E.

We note that the problem, under the streaming model, has so far not received its deserved attention in the community. Prior to the present work, we are aware of only two: for the special case of cardinality constraint, Badanidiyuru *et al.* [9] gave a single-pass $(1/2 - \varepsilon)$-approximation algorithm; for the general case of a knapsack constraint, Yu *et al.* [173] gave a single-pass $(1/3 - \varepsilon)$-approximation algorithm, both using $O(K \log(K)/\varepsilon)$ space.

We now state our contribution.

Theorem 5.1.1. *For the problem* (5.1)*,*

1. *there is a single-pass streaming algorithm with approximation ratio* $4/11 - \varepsilon \approx 0.363 - \varepsilon$*.*
2. *there is a multiple-pass streaming algorithm with approximation ratio* $2/5 - \varepsilon = 0.4 - \varepsilon$*.*

Both algorithms use $O(K \cdot \mathrm{poly}(\varepsilon^{-1})\mathrm{polylog}(K))$ *space.*

Our Technique

We begin by a straightforward generalization of the algorithm of [9] for the special case of cardinality constraint (Section 5.2). This algorithm proceeds by adding a new item into the current set only if its marginal-ratio (its marginal return with respect to the current set divided by its size)

exceeds a certain threshold. This algorithm performs well when all items in OPT are relatively small in size, where OPT is an optimal solution. However, in general, it only gives $(1/3-\varepsilon)$-approximation. Note that this technique can be regarded as a variation of the one in [173]. To obtain better approximation ratio, we need new ideas.

The difficulty in improving this algorithm lies in the following case: A new arriving item that is relatively large in size, passes the marginal-ratio threshold, and is part of OPT, but its addition would cause the current set to exceed the capacity K. In this case, we are forced to throw it away, but in doing so, we are unable to bound the ratio of the function value of the current set against that of OPT properly.

We propose a branching procedure to overcome this issue. Roughly speaking, when the function value of the current set is large enough (depending on the parameters), we create a secondary set. We add an item to the secondary set only if it passes the marginal-ratio threshold (with respect to the original set) but its addition to the original set would violate the size constraint. In the end, whichever set achieves the higher value is returned. In a way, the secondary set serves as a "back-up" with enough space in case the original set does not have it, and this allows us to bound the ratio properly. Sections 5.3 and 5.4 are devoted to explaining this branching algorithm, which gives $(4/11-\varepsilon)$-approximation with a single pass.

We note that the main bottleneck of the above single-pass algorithm lies in the situation where there is a large item in OPT whose size exceeds $K/2$. In Section 5.5, we show that we can first focus on only the large items (more specifically, those items whose size differ from the largest item in OPT by $(1+\varepsilon)$ factor) and choose $O(1)$ of them so that at least one of them, along with the rest of OPT (excluding the largest item in it), gives a good approximation to $f(\mathsf{OPT})$. Then in the next pass, we can apply a modified version of the original single-pass algorithm to collect small items. This multiple-pass algorithm gives a $(2/5-\varepsilon)$-approximation.

Related Work

Maximizing a monotone submodular function subject to various constraints is a subject that has been extensively studied in the literature. We are unable to give a complete survey here and only highlight the most representative and relevant results. Besides a knapsack constraint or a cardinality constraint mentioned above, the problem has also been studied under (multiple) matroid constraint(s), p-system constraint, multiple knapsack constraints. See [15, 17, 20, 50, 116, 121] and the references therein. In the streaming setting, other than the knapsack constraint that we have discussed before, there are also works considering a matroid constraint. Chakrabarti and Kale [16] gave 1/4-approximation; Chekuri *et al.* [18] gave the same ratio.

Notation

For a subset $S \subseteq E$ and an element $e \in E$, we use the shorthand $S+e$ and $S-e$ to stand for $S \cup \{e\}$ and $S \setminus \{e\}$, respectively. For a function $f : 2^E \to \mathbb{R}$, we also use the shorthand $f(e)$ to stand for $f(\{e\})$. The *marginal return* of adding $e \in E$ with respect to $S \subseteq E$ is defined as $f(e \mid S) = f(S+e) - f(S)$. We frequently use the following, which is immediate from the diminishing marginal return property:

Algorithm 1

procedure MarginalRatioThresholding(α, v) ▷ $\alpha \in (0,1], v \in \mathbb{R}_+$
  $S := \emptyset$.
  while item e is arriving **do**
    if $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) \leq K$ **then** $S := S + e$.
  return S.

Proposition 5.1.2. *Let $f : 2^E \to \mathbb{R}_+$ be a monotone submodular function. For two subsets $S \subseteq T \subseteq E$, it holds that $f(T) \leq f(S) + \sum_{e \in T \setminus S} f(e \mid S)$.*

5.2 Single-Pass $(1/3 - \varepsilon)$-Approximation Algorithm

In this section, we present a simple $(1/3-\varepsilon)$-approximation algorithm that generalizes the algorithm for a cardinality constraint in [9]. This algorithm will be incorporated into several other algorithms introduced later.

5.2.1 Thresholding Algorithm with Approximate Optimal Value

In this subsection, we present an algorithm MarginalRatioThresholding, which achieves (almost) 1/3-approximation given a (good) approximation v to $f(\mathsf{OPT})$ for an optimal solution OPT. This assumption is removed in Section 5.2.2.

Given a parameter $\alpha \in (0,1]$ and $v \in \mathbb{R}_+$, MarginalRatioThresholding attempts to add a new item $e \in E$ to the current set $S \subseteq E$ if its addition does not violate the knapsack constraint and e passes the *marginal-ratio threshold condition*, i.e.,

$$\frac{f(e \mid S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}. \tag{5.2}$$

The detailed description of MarginalRatioThresholding is given in Algorithm 1.

Throughout this subsection, we fix $\tilde{S} = \mathsf{MarginalRatioThresholding}(\alpha, v)$ as the output of the algorithm. Then, we have the following lemma.

Lemma 5.2.1. *The following hold:*

(1) *During the execution of the algorithm, the current set $S \subseteq E$ always satisfies $f(S) \geq \alpha v c(S)/K$. Moreover, if an item $e \in E$ passes the condition (5.2) with the current set S, then $f(S+e) \geq \alpha v c(S+e)/K$.*

(2) *If an item $e \in E$ fails the condition (5.2), i.e., $\frac{f(e|S)}{c(e)} < \frac{\alpha v - f(S)}{K - c(S)}$, then we have $f(e \mid \tilde{S}) < \alpha v c(e)/K$.*

Proof. We prove (1) by induction on the size of S. The base case $S = \emptyset$ is trivial. For induction step, suppose that $e \in E$ is the new item to be added into the current set $S \subseteq E$. Then

$$f(S+e) = f(S) + f(e \mid S) \geq f(S) + c(e)\frac{\alpha v - f(S)}{K - c(S)}$$

$$\geq \frac{\alpha v c(e)}{K - c(S)} + f(S)\frac{K - c(S) - c(e)}{K - c(S)} \geq \frac{\alpha v c(S + e)}{K},$$

where the last inequality follows from the induction hypothesis on the lower bound of $f(S)$. Thus (1) holds.

For (2), as the current set satisfies $S \subseteq \tilde{S}$, by the submodularity of f,

$$f(e \mid \tilde{S}) \leq f(e \mid S) < \frac{c(e)(\alpha v - f(S))}{K - c(S)} \leq \frac{\alpha v c(e)}{K},$$

where the last inequality follows from the first part of the lemma.

□

An item $e \in \mathsf{OPT}$ is not added to $\tilde{S}$ if either e does not pass the condition (5.2), or its addition would cause the size of S to exceed the capacity K. We name the latter condition as follows:

Definition 5.2.2. *An item $e \in \mathsf{OPT}$ is called* bad *if e passes the condition (5.2) but the total size exceeds K when added, i.e., $f(e \mid S) \geq \frac{\alpha v - f(S)}{K - c(S)}$, $c(S + e) > K$ and $c(S) \leq K$, where S is the set we have just before e arrives.*

The following lemma says that, if there is no bad item, then we obtain a good approximation.

Lemma 5.2.3. *If $v \leq f(\mathsf{OPT})$ and there have been no bad item, then $f(\tilde{S}) \geq (1 - \alpha)v$ holds.*

Proof. By the submodularity and the monotonicity, we have

$$v \leq f(\mathsf{OPT}) \leq f(\mathsf{OPT} \cup \tilde{S}) \leq f(\tilde{S}) + \sum_{e \in \mathsf{OPT} \setminus \tilde{S}} f(e \mid \tilde{S}).$$

Since we have no bad item, $f(e \mid \tilde{S}) \leq \alpha v c(e)/K$ for any $e \in \mathsf{OPT} \setminus \tilde{S}$ by Lemma 5.2.1 (2). Hence, we have $v \leq f(\tilde{S}) + \alpha v$, implying $f(\tilde{S}) \geq (1 - \alpha)v$.

□

Consider an algorithm Singleton, which takes the best singleton as shown in Algorithm 2. If some item $e \in \mathsf{OPT}$ is bad, then, together with $\tilde{S}' = \mathsf{Singleton}()$, we can achieve (almost) 1/3-approximation.

Theorem 5.2.4. *We have $\max\{f(\tilde{S}), f(\tilde{S}')\} \geq \min\{\alpha/2, 1 - \alpha\}v$. The right-hand side is maximized to $v/3$ when $\alpha = 2/3$.*

Proof. If there exists no bad item, we have $f(\tilde{S}) \geq (1 - \alpha)v$ by Lemma 5.2.3. Suppose that we have a bad item $e \in E$. Let $S_e \subseteq E$ be the set just before e arrives in MarginalRatioThresholding. Then, we have $f(S_e + e) \geq \alpha v c(S_e + e)/K$ by Lemma 5.2.1 (1). Since $c(S_e + e) > K$, this means $f(S_e + e) \geq \alpha v$. Since $f(S_e + e) \leq f(S_e) + f(e)$ by submodularity, one of $f(S_e)$ and $f(e)$ is at least $\alpha v/2$. Thus $f(\tilde{S}) \geq f(S_e) \geq \alpha v/2$ or $f(\tilde{S}') \geq f(e) \geq \alpha v/2$.

□

Therefore, if we have $v \in \mathbb{R}_+$ with $v \leq f(\mathsf{OPT}) \leq (1 + \varepsilon)v$, the algorithm that runs MarginalRatioThresholding$(2/3, v)$ and Singleton() in parallel and chooses the better output has the approximation ratio of $\frac{1}{3(1+\varepsilon)} \geq \frac{1}{3} - \varepsilon$. The space complexity of the algorithm is clearly $O(K)$.

Algorithm 2

procedure Singleton()
  $S := \emptyset$.
  while item e is arriving **do**
    if $f(e) > f(S)$ **then** $S := \{e\}$.
  return S.

Algorithm 3

procedure DynamicMRT(ε, α) ▷ $\varepsilon, \alpha \in (0, 1]$
  $\mathcal{V} := \{(1+\varepsilon)^i \mid i \in \mathbb{Z}_+\}$.
  For each $v \in \mathcal{V}$, set $S_v := \emptyset$.
  while item e is arriving **do**
    $m := \max\{m, f(e)\}$.
    $\mathcal{I} := \{v \in \mathcal{V} \mid m \le v \le Km/\alpha\}$.
    Delete S_v for each $v \notin \mathcal{I}$.
    for each $v \in \mathcal{I}$ **do**
      if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) \le K$ **then** $S_v := S_v + e$.
  return S_v for $v \in \mathcal{I}$ that maximizes $f(S_v)$.

5.2.2 Dynamic Updates

MarginalRatioThresholding requires a good approximation v to $f(\mathsf{OPT})$. This requirement can be removed with dynamic updates in a similar way to [9]. We first observe that $\max_{e \in S} f(e) \le f(\mathsf{OPT}) \le K \max_{e \in S} f(e)$. So if we are given $m = \max_{e \in S} f(e)$ in advance, a value $v \in \mathbb{R}_+$ with $v \le f(\mathsf{OPT}) \le (1+\varepsilon)v$ for $\varepsilon \in (0, 1]$ exists in the guess set $\mathcal{I} = \{(1+\varepsilon)^i \mid m \le (1+\varepsilon)^i \le Km, i \in \mathbb{Z}_+\}$. Then, we can run MarginalRatioThresholding for each $v \in \mathcal{I}$ in parallel and choose the best output. As the size of $\mathcal{I}$ is $O(\log(K)/\varepsilon)$, the total space complexity is $O(K \log(K)/\varepsilon)$.

To get rid of the assumption that we are given m in advance, we consider an algorithm, called DynamicMRT, which dynamically updates m to determine the range of guessed optimal values. More specifically, it keeps the (tentative) maximum value $\max f(e)$, where the maximum is taken over the items e arrived so far, and keeps the approximations v in the interval between m and Km/α. The details are provided in Algorithm 3. We have the following guarantee.

Theorem 5.2.5. *For $\varepsilon \in (0, 1]$, the algorithm that runs* DynamicMRT$(\varepsilon, 2/3)$ *and* Singleton() *in parallel and outputs the better output is a $(1/3 - \varepsilon)$-approximation streaming algorithm with a single pass for the problem (5.1). The space complexity of the algorithm is $O(K \log(K)/\varepsilon)$.*

Proof. Let $e \in E$ be an item arriving. We will show that, if $v > Km/\alpha$ (for $\alpha = 2/3$), then e always fails the condition (5.2) in DynamicMRT. Indeed, if $v > Km/\alpha$ and e passes the condition (5.2) with the current set S, then Lemma 5.2.1 (1) implies that

$$f(S+e) \ge \frac{\alpha v c(S+e)}{K} > c(S+e)m \ge |S+e| \max_{e' \in S+e} f(e'),$$

where the last inequality follows from the fact that $c(e) \ge 1$ and $m \ge \max_{e' \in S+e} f(e')$. On the other hand, $f(S+e) \le |S+e| \max_{e' \in S+e} f(e')$ as f is submodular, which is a contradiction.

Therefore, when an item $e \in E$ arrives, e may be added to the current set only if $v \leq Km/\alpha$. Moreover, since Singleton returns an item e with $f(e) \geq m$, we can discard the case when $v < m$ during the process of DynamicMRT. Thus DynamicMRT simulates all the values in $\mathcal{V}$, only keeping the values in the interval $[m, Km/\alpha]$. Since one of $v \in \mathcal{V}$ satisfies $v \leq f(\mathsf{OPT}) \leq (1+\varepsilon)v$, the output gives $(1/3-\varepsilon)$-approximation from Theorem 5.2.4.

There are $O(\log K/\varepsilon)$ streams, and each stream may have a solution with size $O(K)$. Thus, the total space is as desired.

□

5.3 Improved Single-Pass Algorithm for Small-Size Items

Let $\mathsf{OPT} = \{o_1, o_2, \ldots, o_\ell\}$ be an optimal solution with $c(o_1) \geq c(o_2) \geq \ldots \geq c(o_\ell)$. The main goal of this section is achieving $(2/5-\varepsilon)$-approximation, assuming that $c(o_1) \leq K/2$. The case with $c(o_1) > K/2$ will be discussed in Section 5.4.

5.3.1 Branching Framework with Approximate Optimal Value

We here provide a framework of a branching algorithm BranchingMRT as Algorithm 4. This will be used with different parameters in Section 5.3.2.

Let v and c_1 be (good) approximations to $f(\mathsf{OPT})$ and $c(o_1)/K$, respectively, and let $b \leq 1/2$ be a parameter. The value c_1 is supposed to satisfy $c_1 \leq c(o_1)/K \leq (1+\varepsilon)c_1$. This means that we can ignore items $e \in E$ with $c(e) > \min\{(1+\varepsilon)c_1, 1/2\}K$.

The basic idea of BranchingMRT is to take only items with large marginal ratios, similarly to MarginalRatioThresholding. The difference is that, once $f(S)$ exceeds a threshold λ, where $\lambda = \frac{1}{2}\alpha\,(1-b)\,v$, we store either the current set S or the latest added item as S'. This guarantees that $f(S') \geq \lambda$ and $c(S') \leq (1-b)K$, which means that S' has a large function value and sufficient room to add more elements. We call the process of constructing S' *branching*. We continue to add items with large marginal ratios to the current set S, and if we cannot add an item to S because it exceeds the capacity, we try to add the item to S'. Note that the set S', after branching, can have at most one extra item; but this extra item can be replaced if a better candidate comes along (See line 14–15).

Remark that the sequence of sets S in BranchingMRT is identical to that in MarginalRatioThresholding in Section [**?**]. We say that an item $e \in \mathsf{OPT}$ is *bad* if it is bad in the sense of MarginalRatioThresholding, i.e., it satisfies the condition in Definition 5.2.2. We have the following two lemmas.

Lemma 5.3.1. *For a bad item e with $c(e) \leq bK$, let S_e be the set just before e arrives in Algorithm 4. Then $f(S_e) \geq \lambda$ holds. Thus branching has happened before e arrives.*

Proof. Sine e is a bad item, we have $c(S_e) > K - c(e) \geq (1-b)K$. Hence $f(S_e) \geq \alpha(1-b)v \geq \lambda$ by Lemma 5.2.1 (1). Since the value of f is non-decreasing during the process, it means that branching has happened before e arrives.

□

Algorithm 4

1: **procedure** BranchingMRT($\varepsilon, \alpha, v, c_1, b$) ▷ $\varepsilon, \alpha \in (0,1]$, $v \in \mathbb{R}_+$, and $c_1, b \in [0, 1/2]$
2: $S := \emptyset$.
3: $\lambda := \frac{1}{2}\alpha(1-b)v$.
4: **while** item e is arriving **do**
5: Delete e with $c(e) > \min\{(1+\varepsilon)c_1, 1/2\}K$.
6: **if** $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) \leq K$ **then** $S := S + e$.
7: **if** $f(S) \geq \lambda$ **then break** // leave the While loop.
8: Let $\hat{e}$ be the latest added item in S.
9: **if** $c(S) \geq (1-b)K$ **then** $S'_0 := \{\hat{e}\}$ **else** $S'_0 := S$.
10: $S' := S'_0$.
11: **while** item e is arriving **do**
12: Delete e with $c(e) > \min\{(1+\varepsilon)c_1, 1/2\}K$.
13: **if** $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) \leq K$ **then** $S := S + e$.
14: **if** $\frac{f(e|S)}{c(e)} \geq \frac{\alpha v - f(S)}{K - c(S)}$ and $c(S+e) > K$ **then**
15: **if** $f(S') < f(S'_0 + e)$ **then** $S' := S'_0 + e$.
16: **return** S or S' whichever has the larger function value.

Lemma 5.3.2. *It holds that $f(S'_0) \geq \lambda$ and $c(S'_0) \leq (1-b)K$.*

Proof. We denote by S the set obtained right after leaving the while loop from Line 4. If $c(S) < (1-b)K$, then $f(S'_0) = f(S) \geq \lambda$ and $c(S'_0) = c(S) \leq (1-b)K$. Otherwise, since $c(S) \geq (1-b)K$, we have $f(S) \geq \alpha(1-b)v \geq 2\lambda$ by Lemma 5.2.1 (1). Hence $f(S'_0) = f(\hat{e}) \geq \lambda$ since $f(S - \hat{e}) < \lambda$ and the submodularity. The second part holds since $c(\hat{e}) \leq K/2 \leq (1-b)K$ by $b \leq 1/2$. □

Let $\tilde{S}$ and $\tilde{S}'$ be the final two sets computed by BranchingMRT. Note that we can regard $\tilde{S}$ as the output of MarginalRatioThresholding and $\tilde{S}'$ as the final set obtained by adding at most one item to S'_0.

Observe that the number of bad items depends on the parameter α. As we will show in Section 5.3.2, by choosing a suitable α, if we have more than two bad items, then the size of $\tilde{S}$ is large enough, implying that $f(\tilde{S})$ is already good for approximation (due to Lemma 5.2.1 (1)). Therefore, in the following, we just concentrate on the case when we have at most two bad items.

Lemma 5.3.3. *Let α be a number in $(0,1]$, and suppose that we have only one bad item o_b. If $v \leq f(\mathsf{OPT})$ and $b \in [c(o_\mathrm{b})/K, (1+\varepsilon)c(o_\mathrm{b})/K]$, then it holds that*

$$\begin{aligned}\max\{f(\tilde{S}), f(\tilde{S}')\} &\geq \frac{1}{2}\left(1 - \alpha\frac{K - c(o_\mathrm{b})}{2K}\right)v - \frac{\varepsilon\alpha c(o_\mathrm{b})}{4K}v \\ &= \left(\frac{1}{2}\left(1 - \alpha\frac{K - c(o_\mathrm{b})}{2K}\right) - O(\varepsilon)\right)v.\end{aligned}$$

Proof. Suppose not, that is, suppose that both of $f(\tilde{S})$ and $f(\tilde{S}')$ are smaller than βv, where $\beta = \frac{1}{2}(1 - \alpha\frac{K - c(o_\mathrm{b})}{2K}) - \frac{\alpha c(o_\mathrm{b})}{4K}\varepsilon$. We denote $O_\mathrm{s} = \mathsf{OPT} \setminus \{o_\mathrm{b}\}$.

Since the bad item o_b satisfies $c(o_\mathrm{b}) \leq bK$, it arrives after branching by Lemma 5.3.1. By Lemma 5.3.2, we have $c(S'_0 + o_\mathrm{b}) \leq K$. Since $f(\tilde{S}')$ is less than βv, we see that $f(S'_0 + o_\mathrm{b}) < \beta v$.

Since $f(S'_0) \geq \lambda$,

$$f(\mathsf{OPT}) \leq f(o_{\mathrm{b}} \mid S'_0) + f(S'_0 \cup O_{\mathrm{s}}) < (\beta v - \lambda) + f(S'_0 \cup O_{\mathrm{s}}). \tag{5.3}$$

Since $S'_0 \subseteq \tilde{S}$, submodularity implies that

$$f(S'_0 \cup O_{\mathrm{s}}) \leq f(\tilde{S} \cup O_{\mathrm{s}}) \leq f(\tilde{S}) + \sum_{e \in O_{\mathrm{s}} \setminus \tilde{S}} f(e \mid \tilde{S}). \tag{5.4}$$

Since $f(\tilde{S}) < \beta v$ and no item in O_{s} is bad, (5.3) and (5.4) imply by Lemma 5.2.1 (2) that

$$\begin{aligned} v \leq f(\mathsf{OPT}) &< (\beta v - \lambda) + f(S'_0 \cup O_{\mathrm{s}}) < (\beta v - \lambda) + \beta v + \frac{\alpha c(O_{\mathrm{s}})}{K} v \\ &\leq 2\beta v - \frac{1}{2}\alpha(1-b)v + \alpha\left(1 - \frac{c(o_{\mathrm{b}})}{K}\right) v. \end{aligned}$$

Therefore, we have

$$\beta > \frac{1}{2}\left(1 + \alpha \frac{2c(o_{\mathrm{b}})/K - b - 1}{2}\right).$$

Since $b \leq (1+\varepsilon)c(o_{\mathrm{b}})/K$, we obtain

$$\beta > \frac{1}{2}\left(1 - \alpha \frac{(K - c(o_{\mathrm{b}}))}{2K}\right) - \frac{\alpha c(o_{\mathrm{b}})}{4K}\varepsilon,$$

which is a contradiction. This completes the proof.

□

For the case when we have exactly two bad items, we obtain the following guarantee.

Lemma 5.3.4. *Let α be a number in $(0, 1]$, and suppose that we have exactly two bad items o_{b} and o_{m} with $c(o_{\mathrm{b}}) \geq c(o_{\mathrm{m}})$. If $v \leq f(\mathsf{OPT})$ and $b \in [c(o_{\mathrm{b}})/K, (1+\varepsilon)c(o_{\mathrm{b}})/K]$, then it holds that*

$$\begin{aligned} \max\{f(\tilde{S}), f(\tilde{S}')\} &\geq \frac{1}{3}\left(1 + \alpha\frac{c(o_{\mathrm{m}})}{K}\right) v - \frac{\alpha c(o_{\mathrm{b}})}{3K}\varepsilon v \\ &= \left(\frac{1}{3}\left(1 + \alpha\frac{c(o_{\mathrm{m}})}{K}\right) - O(\varepsilon)\right) v. \end{aligned}$$

Proof. Suppose not, that is, suppose that both of $f(\tilde{S})$ and $f(\tilde{S}')$ are smaller than βv, where $\beta = (1 + \alpha\frac{c(o_{\mathrm{m}})}{K})/3 - \frac{\alpha c(o_{\mathrm{b}})}{3K}\varepsilon$. We denote $O_{\mathrm{s}} = \mathsf{OPT} \setminus \{o_{\mathrm{b}}, o_{\mathrm{m}}\}$.

Since the bad items o_{b} and o_{m} have size at most bK, these two items arrive after branching by Lemma 5.3.1. By Lemma 5.3.2, $c(S'_0 + o_{\mathrm{b}}) \leq K$ and $c(S'_0 + o_{\mathrm{m}}) \leq K$. Since $f(\tilde{S}') < \beta v$, we know $f(S'_0 + o_{\mathrm{b}}) < \beta v$ and $f(S'_0 + o_{\mathrm{m}}) < \beta v$. Hence it holds that

$$\begin{aligned} f(\mathsf{OPT}) &\leq f(o_{\mathrm{b}} \mid S'_0) + f(o_{\mathrm{m}} \mid S'_0) + f(S'_0 \cup O_{\mathrm{s}}) \\ &< (\beta v - \lambda) + (\beta v - \lambda) + f(S'_0 \cup O_{\mathrm{s}}), \end{aligned} \tag{5.5}$$

since $f(S'_0) \geq \lambda$. Since $S'_0 \subseteq \tilde{S}$, we have

$$f(S'_0 \cup O_{\mathrm{s}}) \leq f(\tilde{S} \cup O_{\mathrm{s}}) \leq f(\tilde{S}) + \sum_{e \in O_{\mathrm{s}} \setminus \tilde{S}} f(e \mid \tilde{S}).$$

Since $f(\tilde{S}) < \beta v$ and no items in O_{s} are bad, this implies by Lemma 5.2.1 (2) that

$$f(S'_0 \cup O_{\mathrm{s}}) \leq \beta v + \alpha \frac{c(O_{\mathrm{s}})}{K} v.$$

Hence (5.5) can be transformed to

$$\begin{aligned} v \leq f(\mathsf{OPT}) &< (\beta v - \lambda) + (\beta v - \lambda) + \beta v + \alpha \frac{c(O_{\mathrm{s}})}{K} v \\ &\leq 3\beta v - 2\lambda + \alpha \left(1 - \frac{c(o_{\mathrm{b}})}{K} - \frac{c(o_{\mathrm{m}})}{K}\right) v \\ &= 3\beta v - \alpha(1-b)v + \alpha \left(1 - \frac{c(o_{\mathrm{b}})}{K} - \frac{c(o_{\mathrm{m}})}{K}\right) v. \end{aligned}$$

Therefore, since $b \leq (1+\varepsilon)c(o_{\mathrm{b}})/K$, we have

$$\beta > \frac{1}{3}\left(1 + \alpha \frac{c(o_{\mathrm{m}})}{K}\right) - \frac{\alpha c(o_{\mathrm{b}})}{3K}\varepsilon,$$

which is a contradiction.

□

5.3.2 Algorithms with Guessing Large Items

We now use BranchingMRT to obtain a better approximation ratio. In the new algorithm, we guess the sizes of a few large items in an optimal solution OPT, and then use them to determine the parameter α.

We first remark that, when $|\mathsf{OPT}| \leq 2$, we can easily obtain a 1/2-approximate solution with a single pass. In fact, since $f(\mathsf{OPT}) \leq \sum_{i=1}^{\ell} f(o_i)$ where $\ell = |\mathsf{OPT}|$, at least one of o_i's satisfies $f(o_i) \geq f(\mathsf{OPT})/\ell$, and hence Singleton returns a 1/2-approximate solution when $\ell \leq 2$. Thus, in what follows, we may assume that $|\mathsf{OPT}| \geq 3$.

We start with the case that we have guessed the largest two sizes $c(o_1)$ and $c(o_2)$ in OPT.

Lemma 5.3.5. *Let $\varepsilon \in (0,1]$, and suppose that $v \leq f(\mathsf{OPT})$ and $c_i \leq c(o_i)/K \leq (1+\varepsilon)c_i$ for $i \in \{1,2\}$. Then, $\tilde{S}' = \mathsf{BranchingMRT}(\varepsilon, \alpha, v, c_1, b)$ with $\alpha = 1/(2-c_2)$ or $2/(5-4c_2-c_1)$ and $b = \min\{(1+\varepsilon)c_1, 1/2\}$ satisfies*

$$f(\tilde{S}') \geq \left(\min\left\{\frac{1-c_2}{2-c_2}, \frac{2(1-c_2)}{5-4c_2-c_1}\right\} - O(\varepsilon)\right) v. \tag{5.6}$$

Proof. Let $\tilde{S} = \mathsf{MarginalRatioThresholding}(\alpha, v)$. Note that $f(\tilde{S}') \geq f(\tilde{S})$. If $\tilde{S}$ has size at least

$(1-(1+\varepsilon)c_2)K$, then Lemma 5.2.1 (1) implies that

$$f(\tilde{S}) \geq \alpha(1-(1+\varepsilon)c_2)v = \alpha(1-c_2)v - O(\varepsilon)v.$$

Otherwise, $c(\tilde{S}) < (1-(1+\varepsilon)c_2)K$. In this case, we see that only the item o_1 can have size more than $(1+\varepsilon)c_2K$, and hence only o_1 can be a bad item. If o_1 is not a bad item, then we have no bad item, and hence Lemma 5.2.3 implies that

$$f(\tilde{S}) \geq (1-\alpha)v.$$

If o_1 is bad, then Lemma 5.3.3 implies that

$$f(\tilde{S}') \geq \frac{1}{2}\left(1-\alpha\frac{1-c_1}{2}\right)v - O(\varepsilon)v.$$

Thus the approximation ratio is the minimum of the RHSes of the above three inequalities. This is maximized when $\alpha = 1/(2-c_2)$ or $\alpha = 2/(5-4c_2-c_1)$, and the maximum value is equal to the RHS of (5.6).

□

Note that the approximation ratio achieved in Lemma 5.3.5 becomes $1/3 - O(\varepsilon)$ when, for example, $c_1 = c_2 = 1/2$. Hence, the above lemma does not show any improvement over Theorem 5.2.4 in the worst case. Thus, we next consider the case that we have guessed the largest three sizes $c(o_1)$, $c(o_2)$, and $c(o_3)$ in OPT. Using Lemma 5.3.4, we have the following guarantee.

Lemma 5.3.6. *Let $\varepsilon \in (0,1]$, and suppose that $v \leq f(\mathsf{OPT})$ and $c_i \leq c(o_i)/K \leq (1+\varepsilon)c_i$ for $i \in \{1,2,3\}$. Then the better output $\tilde{S}'$ of $\mathsf{BranchingMRT}(\varepsilon,\alpha,v,c_1,b_1)$ and $\mathsf{BranchingMRT}(\varepsilon,\alpha,v,c_1,b_2)$ with $\alpha = 1/(2-c_3)$ or $2/(c_2+3)$, $b_1 = \min\{(1+\varepsilon)c_1, 1/2\}$, and $b_2 = \min\{(1+\varepsilon)c_2, 1/2\}$ satisfies*

$$f(\tilde{S}') \geq \left(\min\left\{\frac{1-c_3}{2-c_3}, \frac{c_2+1}{c_2+3}\right\} - O(\varepsilon)\right)v.$$

Proof. Let $\tilde{S} = \mathsf{MarginalRatioThresholding}(\alpha, v)$. If $\tilde{S}$ has size at least $(1-(1+\varepsilon)c_3)K$, then we have by Lemma 5.2.1 (1)

$$f(\tilde{S}) \geq \alpha(1-(1+\varepsilon)c_3)v = \alpha(1-c_3)v - O(\varepsilon)v.$$

Otherwise, $c(\tilde{S}) < (1-(1+\varepsilon)c_3)K$. In this case, we see that only o_1 and o_2 can have size more than $(1+\varepsilon)c_3$, and hence only they can be bad items. If we have no bad item, it holds by Lemma 5.2.3 that

$$f(\tilde{S}) \geq (1-\alpha)v.$$

Suppose we have one bad item. If it is o_1 then Lemma 5.3.3 with b_1 implies

$$f(\tilde{S}') \geq \left(\frac{1}{2}\left(1-\alpha\frac{1-c_1}{2}\right) - O(\varepsilon)\right)v,$$

and, if it is o_2, we obtain by Lemma 5.3.3 with b_2

$$f(\tilde{S}') \geq \left(\frac{1}{2}\left(1 - \alpha\frac{1-c_2}{2}\right) - O(\varepsilon)\right) v.$$

Moreover, if we have two bad items o_1 and o_2, then Lemma 5.3.4 implies

$$f(\tilde{S}') \geq \left(\frac{1}{3}(1 + \alpha c_2) - O(\varepsilon)\right) v.$$

Therefore, the approximation ratio is the minimum of the RHSes in the above five inequalities, which is maximized to

$$\min\left\{\frac{1-c_3}{2-c_3}, \frac{c_2+1}{c_2+3}\right\} - O(\varepsilon),$$

when $\alpha = 1/(2-c_3)$ or $\alpha = 2/(c_2+3)$.

□

We now see that we get an approximation ratio of $2/5 - O(\varepsilon)$ by combining the above two lemmas.

Theorem 5.3.7. *Let $\varepsilon \in (0,1]$ and suppose that $v \leq f(\mathsf{OPT}) \leq (1+\varepsilon)v$ and $c_i \leq c(o_i)/K \leq (1+\varepsilon)c_i$ for $i \in \{1,2,3\}$. If $c(o_1) \leq K/2$, then we can obtain a $(2/5 - O(\varepsilon))$-approximate solution with a single pass.*

Proof. We run the two algorithms with the optimal α shown in Lemmas 5.3.5 and 5.3.6 in parallel. Let $\tilde{S}$ be the output with the better function value. Then, we have $f(\tilde{S}) \geq \beta v$, where

$$\beta = \max\left\{\min\left\{\frac{1-c_2}{2-c_2}, \frac{2(1-c_2)}{5-4c_2-c_1}\right\}, \min\left\{\frac{1-c_3}{2-c_3}, \frac{c_2+1}{c_2+3}\right\}\right\} - O(\varepsilon).$$

We can confirm that the first term is at least $2/5$, and thus $\tilde{S}$ is a $(2/5 - O(\varepsilon))$-approximate solution.

□

To eliminate the assumption that we are given v, we can design a dynamic-update version of BranchingMRT by keeping the interval that contains the optimal value, similarly to Theorem 5.2.5. The algorithm, called DynamicBranchingMRT, is given in Algorithm 5. The proof for updating the interval $\mathcal{I}$ dynamically is the same as the proof of Theorem 5.2.5. The number of streams for guessing v is $O(\log(K)/\varepsilon)$. We also guess c_i for $i \in \{1,2,3\}$ from $\{(1+\varepsilon)^j \mid j \in \mathbb{Z}_+\}$. As $1 \leq c(o_i) \leq K/2$ for $i \in \{1,2,3\}$, the number of guessing for c_i is $O(\log(K)/\varepsilon)$. Hence, including v, there are $O((\log(K)/\varepsilon)^4)$ streams in parallel. To summarize, we obtain the following:

Theorem 5.3.8. *Suppose that $c(o_1) \leq K/2$. The algorithm that runs* DynamicBranchingMRT *and* Singleton *in parallel and takes the better output is a $(2/5-\varepsilon)$-approximation streaming algorithm with a single pass for the problem (5.1). The space complexity of the algorithm is $O(K(\log(K)/\varepsilon)^4)$.*

Algorithm 5

procedure DynamicBranchingMRT(ε)
  $\mathcal{V} := \{(1+\varepsilon)^i \mid i \in \mathbb{Z}_+\}$.
  For each $c_1, c_2, c_3 \in \mathcal{V}$ with $c_3 \le c_2 \le c_1 \le 1/2$ and each $b \in \{(1+\varepsilon)c_1, (1+\varepsilon)c_2, 1/2\}$, do the following with α defined based on Lemmas 5.3.5 and 5.3.6.
  For each $v \in \mathcal{V}$, set $S_v := \emptyset$.
  while item e is arriving **do**
    Delete e with $c(e) > (1+\varepsilon)c_1 K$.
    $m := \max\{m, f(e)\}$.
    $\mathcal{I} := \{v \in \mathcal{V} \mid m \le v \le Km/\alpha\}$.
    Delete S_v (along with $\hat{S}_v$ and S'_v if exists) such that $v \notin \mathcal{I}$.
    for $v \in \mathcal{V}$ **do**
      if $f(S_v) < \lambda$ **then**
        if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) \le K$ **then** $S_v := S_v + e$.
        if $f(S_v) \ge \lambda$ **then**
          if $c(S) \ge (1-b)K$ **then** $S' := \{e\}$ **else** $S' := S$.
          $\hat{S}_v := S'$.
      else
        if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) \le K$ **then** $S_v := S_v + e$.
        if $\frac{f(e|S_v)}{c(e)} \ge \frac{\alpha v - f(S_v)}{K - c(S_v)}$ and $c(S_v + e) > K$ **then**
          if $f(S'_v) < f(\hat{S}_v + e)$ **then** $S'_v := \hat{S}_v + e$.
  $S := S_v$ for $v \in \mathcal{I}$ that maximizes $f(S_v)$.
  $S' := S'_v$ for $v \in \mathcal{I}$ that maximizes $f(S'_v)$.
  return S or S' whichever has the larger function value.

5.4 Single-Pass $(4/11 - \varepsilon)$-Approximation Algorithm

In this section, we consider the case that $c(o_1)$ is larger than $K/2$. For the purpose, we consider the problem of finding a set S of items that maximizes $f(S)$ subject to the constraint that the total size is at most pK, for a given number $p \ge 2$. We say that a set S of items is a (p, α)-*approximate solution* if $c(S) \le pK$ and $f(S) \ge \alpha f(\mathsf{OPT})$, where OPT is an optimal solution of the original instance.

Theorem 5.4.1. *For a number $p \ge 2$, there is a $\left(p, \frac{2p}{2p+3} - \varepsilon\right)$-approximation streaming algorithm with a single pass for the problem (5.1). In particular, when $p = 2$, it admits $(2, 4/7 - \varepsilon)$-approximation. The space complexity of the algorithm is $O(K(\log(K)/\varepsilon)^3)$.*

The proof will be given in the next subsection. Using Theorem 5.4.1, we can design a $(4/11 - \varepsilon)$-approximation streaming algorithm for an instance having a large item.

Theorem 5.4.2. *For the problem (5.1), there exists a $(4/11 - \varepsilon)$-approximation streaming algorithm with a single pass. The space complexity of the algorithm is $O(K(\log(K)/\varepsilon)^4)$.*

Proof. Let o_1 be an item in OPT with the maximum size. If $c(o_1) \le K/2$, then Theorem 5.3.8 gives a $(2/5 - O(\varepsilon))$-approximate solution, and thus we may assume that $c(o_1) > K/2$. Note that there exists only one item whose size is more than $K/2$. Let β be the target approximation

ratio which will be determined later. We may assume that $f(o_1) < \beta f(\mathsf{OPT})$, as otherwise Singleton (Algorithm 2) gives β-approximation. Then, we see $f(\mathsf{OPT} - o_1) > (1-\beta)f(\mathsf{OPT})$ and $c(\mathsf{OPT} - o_1) < K/2$. Consider maximizing $f(S)$ subject to $c(S) \leq K/2$ in the set $\{e \in E \mid c(e) \leq K/2\}$. The optimal value is at least $f(\mathsf{OPT} - o_1) > (1-\beta)f(\mathsf{OPT})$. We now apply Theorem 5.4.1 with $p = 2$ to this problem. Then, the output $\tilde{S}$ has size at most K, and moreover, we have $f(\tilde{S}) \geq \left(\frac{4}{7} - O(\varepsilon)\right)(1-\beta)f(\mathsf{OPT})$. Thus, we obtain $\min\{\beta, (\frac{4}{7} - O(\varepsilon))(1-\beta)\}$-approximation. This approximation ratio is maximized to $4/11$ when $\beta = 4/11$.

□

5.4.1 Bicriteria Approximation for a Knapsack Constraint

We here present the proof of Theorem 5.4.1. Let $p \geq 2$.

The basic framework is the same as in Section 5.3; we design both a simple-thresholding algorithm and a branching algorithm, where the parameters are different and the analysis is simpler. It is sufficient to design algorithms assuming that a (good) approximation v to $f(\mathsf{OPT})$ is given, as we can get rid of the assumption by using the dynamic update technique.

We design a variant of MarginalRatioThresholding. The new algorithm is parameterized by a number $p \geq 2$. In the algorithm we allow to pack items to the total size at most pK. Also, we change the marginal-ratio threshold condition to the following:

$$\frac{f(s \mid S)}{c(e)} \geq \frac{\alpha p v - f(S)}{pK - c(S)}. \tag{5.7}$$

Let $\mathsf{MarginalRatioThresholding}'_p$ be the resulting algorithm.

Similarly to Lemma 5.2.1, the following lemma holds. The proof is omitted as it is almost identical to that of Lemma 5.2.1.

Lemma 5.4.3. *Let $\tilde{S} = \mathsf{MarginalRatioThresholding}'_p(\alpha, v)$ for some $\alpha \in (0,1]$ and $v \in \mathbb{R}_+$. Then, the following hold:*

(1) *During the execution of the algorithm, we have $f(S) \geq \alpha v c(S)/K$.*

(2) *If an item e fails the marginal-ratio threshold condition, i.e., $\frac{f(e|S)}{c(e)} < \frac{\alpha p v - f(S)}{pK - c(S)}$, then $f(e \mid \tilde{S}) < \alpha v c(e)/K$.*

Determining α using a good approximation to the largest size $c(o_1)$ in OPT gives the following approximation guarantee:

Lemma 5.4.4. *For $\varepsilon \in (0,1]$, suppose that $v \leq f(\mathsf{OPT})$ and $c_1 \leq c(o_1)/K \leq (1+\varepsilon)c_1$. Then, $\tilde{S} = \mathsf{MarginalRatioThresholding}'_p(\alpha, v)$, where $\alpha = 1/(p+1-c_1)$, satisfies*

$$f(\tilde{S}) \geq \left(\frac{p - c_1}{p + 1 - c_1} - O(\varepsilon)\right) v.$$

Proof. If the output $\tilde{S}$ has size at least $(p - (1+\varepsilon)c_1)K$, then we have by Lemma 5.4.3 (1)

$$f(\tilde{S}) \geq \alpha(p - (1+\varepsilon)c_1)v = \alpha(p - c_1)v - O(\varepsilon)v.$$

Otherwise, $c(\tilde{S}) < (p-(1+\varepsilon)c_1)K$, and hence there is no bad item. Similarly to Lemma 5.2.3, it follows from Lemma 5.4.3 (2) that we have

$$f(\tilde{S}) \geq (1-\alpha)v.$$

The approximation ratio is the minimum of the RHSes of the above two inequalities, which is maximized to $(p-c_1)/(p+1-c_1) - O(\varepsilon)$ by setting $\alpha = 1/(p+1-c_1)$.

□

Next, we design a branching algorithm based on BranchingMRT. Here, the parameter b should be at most 1, and the marginal-ratio threshold is replaced with (5.7). Also, λ is set to be

$$\lambda = \frac{1}{2}\alpha\,(p-b)\,v,$$

and, at Line 8 of Algorithm 4, the condition is changed to $(p-b)K$ instead of $(1-b)K$. Let $\mathsf{BranchingMRT}'_p$ be the resulting algorithm.

The analysis in Section 5.3 can be adapted:

Lemma 5.4.5. *The following hold for* $\mathsf{BranchingMRT}'_p$*:*

- *For a bad item $e \in E$ with $c(e) \leq bK$, let S_e be the set just before e arrives. Then $f(S_e) \geq \lambda$ holds. Thus, branching has happened before e arrives.*
- *It holds that $f(S'_0) \geq \lambda$ and $c(S'_0) \leq (p-b)K$.*

Note that in the second statement, we do not need the assumption that $b \leq 1/2$ as $c(\hat{e}) \leq K \leq (p-b)K$ since $b \leq 1$.

Determining α using good approximations to the largest two sizes $c(o_1)$ and $c(o_2)$ in OPT gives the following approximation guarantee:

Lemma 5.4.6. *For $\varepsilon \in (0,1]$, suppose that $v \leq f(\mathsf{OPT})$ and $c_i \leq c(o_i)/K \leq (1+\varepsilon)c_i$ for $i \in \{1,2\}$. Then $\tilde{S} = \mathsf{BranchingMRT}'_p(\varepsilon, \alpha, v, c_1, b)$ with $c_1 \leq b \leq (1+\varepsilon)c_1$ and $\alpha = \frac{2}{c_1+p+2}$ satisfies*

$$f(\tilde{S}) \geq \left(\frac{c_1+p}{c_1+p+2} - O(\varepsilon)\right) v.$$

Proof. If the output $\tilde{S}$ has size at least $(p-(1+\varepsilon)c_2)K$, then we have by Lemma 5.4.3 (1)

$$f(\tilde{S}) \geq \alpha(p-(1+\varepsilon)c_2)v = (\alpha(p-c_2) - O(\varepsilon))\,v.$$

Otherwise, $c(\tilde{S}) < (p-(1+\varepsilon)c_2)K$. In this case, we see that there exists at most one bad item. If we have no bad item, it holds by Lemma 5.4.3 (2) that

$$f(\tilde{S}) \geq (1-\alpha)v.$$

Suppose that we have one bad item, which must be o_1. Following the proof of Lemma 5.3.3, we

see that

$$f(\tilde{S}) \geq \left(\frac{1}{2}\left(1+\alpha\frac{c_1+p-2}{2}\right)-O(\varepsilon)\right)v.$$

The approximation ratio is the minimum of the RHSes of the above three inequalities. It is maximized to

$$\min\left\{\frac{p-c_2}{p+1-c_2}, \frac{c_1+p}{c_1+p+2}\right\}-O(\varepsilon).$$

when $\alpha = \frac{1}{p-c_2+1}$ or $\alpha = \frac{2}{c_1+p+2}$. This is in fact equal to $\frac{c_1+p}{c_1+p+2} - O(\varepsilon)$ with $\alpha = \frac{2}{c_1+p+2}$, since $p \geq 2$.

□

Therefore, if we apply both of the above algorithms and take the better one, we obtain a set $\tilde{S} \subseteq E$ satisfying

$$f(\tilde{S}) \geq \left(\max\left\{\frac{p-c_1}{p+1-c_1}, \frac{c_1+p}{c_1+p+2}\right\}-O(\varepsilon)\right)v.$$

This is minimized when $c_1 = p/3$, and hence we have

$$f(\tilde{S}) \geq \left(\frac{2p}{2p+3}-O(\varepsilon)\right)v.$$

Thus the set $\tilde{S}$ can be found in $O(K(\log(K)/\varepsilon)^2)$ space. This, together with the dynamic update technique to guess v, proves Theorem 5.4.1.

5.5 Multiple-Pass Streaming Algorithm

In this section, we provide a multiple-pass streaming algorithm with approximation ratio $2/5-\varepsilon$. In Section 5.5.1, we consider the monotone submodular function maximization with a different constraint and develop an algorithm for it. In Section 5.5.2, this algorithm is used as a subroutine for the original problem with a knapsack constraint.

5.5.1 Dealing with Large Items with Single Pass

We first consider a generalization of the original problem. Let $E_{\mathrm{r}} \subseteq E$ be a subset of the ground set E. For ease of presentation, we will call E_{r} the *red* items. Consider the problem defined below:

$$\text{maximize} \quad f(S) \quad \text{subject to} \quad c(S) \leq K, \quad |S \cap E_R| \leq 1. \tag{5.8}$$

In the following, we show that, given $\varepsilon \in (0,1]$, an approximation v to $f(\mathsf{OPT})$ with $v \leq f(\mathsf{OPT}) \leq (1+\varepsilon)v$, and an approximation θ to $f(o_{\mathrm{r}})$ for the unique item o_{r} in $\mathsf{OPT} \cap E_{\mathrm{r}}$, we can choose $O(1)$ of the red items so that one of them $e \in E_{\mathrm{r}}$ satisfies that $f(\mathsf{OPT}-o_{\mathrm{r}}+e) \geq (\Gamma(\theta)-O(\varepsilon))v$, where Γ is a piecewise linear function lower-bounded by $2/3$. For technical reasons, we will choose θ to be one of the geometric series $(1+\varepsilon)^i/2$ for $i \in \mathbb{Z}$.

Theorem 5.5.1. *Suppose that we are given $\varepsilon \in (0,1]$, $v \in \mathbb{R}_+$ with $v \leq f(\mathsf{OPT}) \leq (1+\varepsilon)v$, and $\theta \in \mathbb{R}_+$ with the following property:*

Algorithm 6

```
1: procedure SelectRedItems(ε, v, θ, t, x)        ▷ ε ∈ (0,1], v ∈ ℝ+, θ ≤ 1/2, t ∈ ℤ+, and x ∈ ℝ+
2:     S := ∅.
3:     while item e is arriving do
4:         if e ∈ E_r and f(e) ≥ θv/(1+ε) then
5:             if S = ∅ then
6:                 S := {e}.
7:             else
8:                 if f(e | S) > (θ − x|S|)v then S := S + e.
9:                 if |S| = t + 1 then return S.
10:    return S.
```

1. *if* $\theta \le 1/2$, $\theta v/(1+\varepsilon) \le f(o_{\mathrm{r}}) \le \theta v$,

2. *if* $\theta \ge 1/2$, $\theta v \le f(o_{\mathrm{r}}) \le (1+\varepsilon)\theta v \le v$.

Then, there is a single-pass streaming algorithm that chooses a set S of red items in E_{r} with constant size such that (i) *for any item $e \in S$, $\theta v/(1+\varepsilon) \le f(o_{\mathrm{r}}) \le \theta v$ when $\theta \le 1/2$ and $\theta v \le f(o_{\mathrm{r}}) \le (1+\varepsilon)\theta v \le v$ when $\theta \ge 1/2$, and* (ii) *some item $e \in S$ satisfies that $f(\mathsf{OPT} - o_{\mathrm{r}} + e) \ge (\Gamma(\theta) - O(\varepsilon))v$, where $\Gamma(\theta)$ is defined as follows: when $\theta \in (0, 1/2)$,*

$$\Gamma(\theta) = \max\Big\{\frac{t(t+3)}{(t+1)(t+2)} - \frac{t-1}{t+1}\theta \mid t \in \mathbb{Z}_+, t > \frac{1}{\theta} - 2\Big\}, \tag{5.9}$$

when $\theta \in [1/2, 2/3)$, $\Gamma(\theta) = 2/3$, and when $\theta \in [2/3, 1]$, $\Gamma(\theta) = \theta$.

In what follows, we prove Theorem 5.5.1 considering the cases $\theta \le 1/2$ and $\theta \ge 1/2$ separately.

Case 1: $\theta \le 1/2$

In this case, θ is supposed to satisfy $\theta v/(1+\varepsilon) \le f(o_{\mathrm{r}}) \le \theta v$. Then, we can just ignore all red items $e \in E_{\mathrm{r}}$ with $f(e) < \theta v/(1+\varepsilon)$ or $f(e) > \theta v$. Hence in the following, we assume that all the arriving red items e satisfy $f(e) \ge \theta v/(1+\varepsilon)$.

Our algorithm picks the first red item e_1 and then collects up to $t+1$ red items, where t is a constant determined later. Observe that, as $v \le f(\mathsf{OPT}) \le f(o_{\mathrm{r}}) + f(\mathsf{OPT} - o_{\mathrm{r}}) \le \theta v + f(\mathsf{OPT} - o_{\mathrm{r}})$, we have $f(\mathsf{OPT} - o_{\mathrm{r}}) \ge (1-\theta)v$. The algorithm guarantees that one of the chosen red items, along with $f(\mathsf{OPT} - o_{\mathrm{r}})$, gives the value of $(1 - \theta + x)v$, where x is the term we will try to maximize. Our algorithm, SelectRedItems, is given in Algorithm 6.

The following lemma follows immediately from the algorithm.

Lemma 5.5.2. *During the execution of SelectRedItems, $f(S) \ge v(\theta(\frac{1}{1+\varepsilon} + |S| - 1) - x\sum_{j=1}^{|S|-1} j)$.*

Proof. Since the first item e we pick satisfies $f(e) \ge \frac{\theta}{1+\varepsilon}v$, the condition at line 8 implies that

$$f(S) \ge \frac{\theta}{1+\varepsilon}v + \sum_{j=1}^{|S|-1}(\theta - xj)v.$$

Thus the lemma follows.

□

The next lemma states that if o_{r} is thrown away at line 8 of the algorithm, then one of the red items in S is already good for our purpose.

Lemma 5.5.3. *Suppose that $|S|< t+1$ holds for the current set $S \subseteq E_{\mathrm{r}}$ and the arriving item is o_{r} and is thrown away at line 8 of the algorithm. Then at least one red item $\overline{e} \in S$ satisfies $f(\mathsf{OPT} - o_{\mathrm{r}} + \overline{e}) \geq (1-\theta+x)v$.*

Proof. First suppose that $f(S \cup (\mathsf{OPT} - o_{\mathrm{r}})) \geq (1-\theta+|S|x)v$. Then

$$(1-\theta+|S|x)v \leq f(S \cup (\mathsf{OPT} - o_{\mathrm{r}})) \leq f(\mathsf{OPT} - o_{\mathrm{r}}) + \sum_{e \in S} f(e \mid \mathsf{OPT} - o_{\mathrm{r}}),$$

implying that at least one red item $\overline{e} \in S$ ensures that

$$f(\overline{e} \mid \mathsf{OPT} - o_{\mathrm{r}}) \geq \frac{(1-\theta+|S|x)v - f(\mathsf{OPT} - o_{\mathrm{r}})}{|S|}.$$

So we obtain

$$\begin{aligned} f(\mathsf{OPT} - o_{\mathrm{r}} + \overline{e}) &= f(\overline{e} \mid \mathsf{OPT} - o_{\mathrm{r}}) + f(\mathsf{OPT} - o_{\mathrm{r}}) \\ &\geq \left(x + \frac{1-\theta}{|S|}\right) v + \frac{|S|-1}{|S|} f(\mathsf{OPT} - o_{\mathrm{r}}) \geq (1-\theta+x)v, \end{aligned}$$

as $f(\mathsf{OPT} - o_{\mathrm{r}}) \geq (1-\theta)v$.

Next assume that $f(S \cup (\mathsf{OPT} - o_{\mathrm{r}})) < (1-\theta+|S|x)v$. But if this is the case, o_{r} would not have been thrown away by the algorithm in line 8, since

$$\begin{aligned} f(o_{\mathrm{r}} \mid S) &\geq f(o_{\mathrm{r}} \mid S \cup (\mathsf{OPT} - o_{\mathrm{r}})) = f(\mathsf{OPT} \cup S) - f(S \cup (\mathsf{OPT} - o_{\mathrm{r}})) \\ &\geq v - (1-\theta+|S|x)v = (\theta - |S|x)v. \end{aligned}$$

Thus the proof is complete.

□

The next lemma states that, if $|S|= t+1$, we can just ignore the rest, no matter o_{r} has arrived or not.

Lemma 5.5.4. *Suppose that $|S|= t+1$. Then at least one red item $\overline{e} \in S$ guarantees that*

$$f(\mathsf{OPT} - o_{\mathrm{r}} + \overline{e}) \geq \left(\frac{\theta(1-\varepsilon)+t}{t+1} - \frac{tx}{2}\right) v. \tag{5.10}$$

Proof. As $f(\mathsf{OPT} - o_{\mathrm{r}}) + \sum_{e \in S} f(e \mid \mathsf{OPT} - o_{\mathrm{r}}) \geq f(S)$, there exists an item $\overline{e} \in S$ so that

$f(\overline{e} \mid \mathsf{OPT} - o_{\mathrm{r}}) \geq \frac{f(S) - f(\mathsf{OPT} - o_{\mathrm{r}})}{|S|}$, implying that

$$\begin{aligned}
f(\mathsf{OPT} - o_{\mathrm{r}} + \overline{e}) &= f(\overline{e} \mid \mathsf{OPT} - o_{\mathrm{r}}) + f(\mathsf{OPT} - o_{\mathrm{r}}) \\
&\geq \frac{f(S)}{t+1} + \frac{t}{t+1} f(\mathsf{OPT} - o_{\mathrm{r}}) \\
&\geq \left(\frac{\theta\left(\frac{1}{1+\varepsilon} + t\right) - \frac{xt(t+1)}{2}}{t+1} + \frac{t}{t+1}(1-\theta) \right) v && \text{(By Lemma 5.5.2)} \\
&\geq \left(\frac{t + \theta(1-\varepsilon)}{t+1} - \frac{tx}{2} \right) v && \text{(Rearranging and using } 1/(1+\varepsilon) \geq 1-\varepsilon\text{)}
\end{aligned}$$

□

It follows from the two previous lemmas that the output is lower bounded by

$$\min\left\{1 - \theta + x, \frac{\theta + t}{t+1} - \frac{tx}{2}\right\} v - \frac{\theta}{t+1}\varepsilon v. \tag{5.11}$$

If $t > 1/\theta - 2$, then we can ignore the second term because it is $O(\varepsilon)v$. In what follows, we consider maximizing the first term of (5.11) subject to $t \in \mathbb{Z}_+$ with $t > 1/\theta - 2$ and $x \in [0, \theta]$, for a given parameter θ.

Suppose that t is a fixed number. Then, since both the terms in (5.11) are linear functions with respect to x, the maximum of (5.11) is attained when they are equal. That is, it is when

$$x_t^* = 2\frac{\theta t + 2\theta - 1}{(t+1)(t+2)} = 2\left(\frac{1}{t+2} - \frac{1-\theta}{t+1}\right). \tag{5.12}$$

We see $x_t^* \in [0, \theta]$ when $t > 1/\theta - 2$. Therefore, by substituting for (5.12), the first term of (5.11) is changed to

$$1 - \theta + x_t^* = \frac{t(t+3)}{(t+1)(t+2)} - \frac{t-1}{t+1}\theta.$$

Thus, if we run $\mathsf{SelectRedItems}(\varepsilon, v, \theta, t, x_t^*)$ with different t, then the best output $\tilde{S}$ satisfies $f(\tilde{S}) \geq (\Gamma(\theta) - O(\varepsilon))v$, where

$$\Gamma(\theta) = \max\left\{ \frac{t(t+3)}{(t+1)(t+2)} - \frac{t-1}{t+1}\theta \mid t \in \mathbb{Z}_+, t > \frac{1}{\theta} - 2 \right\}. \tag{5.13}$$

This is a piecewise convex non-increasing function of θ. See Figure 5.1 for the ratio calculated by only considering $t \leq 10$.

Case 2: $\theta \geq 1/2$

We now present another algorithm for the case of $\theta \geq 1/2$ and define the function Γ for the interval of $[1/2, 1]$. In this case, θ is supposed to satisfy $\theta v \leq f(o_{\mathrm{r}}) \leq \theta v(1+\varepsilon) \leq v$. Thus, in the following, we assume that $\theta v \leq f(e) \leq (1+\varepsilon)\theta v$ for all red items $e \in E_{\mathrm{r}}$.

If $\theta \geq 2/3$, just pick any red item e with $f(e) \geq \theta v$ gives trivially $f(\mathsf{OPT} - o_{\mathrm{r}} + e) \geq \theta v$. Thus, we can define $\Gamma(\theta) = \theta$ when $\theta \in [2/3, 1]$. The remaining case is when $\theta \in [1/2, 2/3)$. We

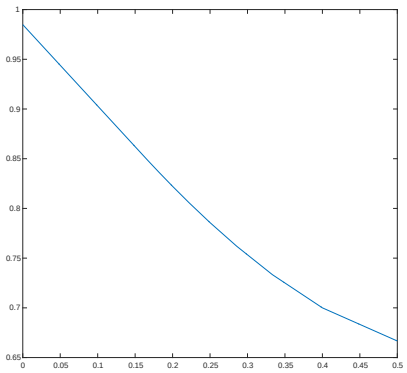

Figure 5.1: Function $\Gamma(\theta)$.

Algorithm 7

1: **procedure** SelectRedItems'(ε, v, θ) ▷ $\varepsilon \in (0,1]$, $v \in \mathbb{R}_+$ and $\theta \geq 1/2$
2: $S := \{e_1\}$, where e_1 is the first arriving item in E_{r}.
3: **while** item $e \in E_{\mathrm{r}}$ is arriving **do**
4: **if** $f(S+e) \geq (1/3 + \theta(1+\varepsilon))v$ **then** $e_2 := e$, $S := S + e_2$ **and return** S.
5: **return** S.

present an algorithm, SelectRedItems', for this case to guarantee that one of the chosen items e has $f(\mathsf{OPT} - o_{\mathrm{r}} + e) \geq (2/3 - \varepsilon)v$. Namely, we will just let $\Gamma(\theta) = 2/3$ for the interval $\theta \in [1/2, 2/3)$. The detail of the algorithm is provided in Algorithm 7.

To avoid triviality, we assume that $e_1 \neq o_{\mathrm{r}}$. The next lemma states that if there are two items in the returned set S, at least one serves the purpose.

Lemma 5.5.5. *Suppose that $S = \{e_1, e_2\}$. Then it cannot happen that $f(\mathsf{OPT} - o_{\mathrm{r}} + e_i) < 2/3v$ for both $i \in \{1, 2\}$.*

Proof. As $f(\mathsf{OPT} - o_{\mathrm{r}}) + f(o_{\mathrm{r}}) \geq f(\mathsf{OPT}) \geq v$ and $f(o_{\mathrm{r}}) \leq (1+\varepsilon)\theta v$, we have $f(\mathsf{OPT} - o_{\mathrm{r}}) \geq (1 - \theta(1+\varepsilon))v$. Now suppose that $f(\mathsf{OPT} - o_{\mathrm{r}} + e_i) < 2v/3$ for both $i \in \{1, 2\}$. Then we have

$$f(e_1 \mid \mathsf{OPT} - o_{\mathrm{r}}) = f(e_1 + \mathsf{OPT} - o_{\mathrm{r}}) + f(\mathsf{OPT} - o_{\mathrm{r}}) \leq \left(\frac{2}{3} - (1 - (1+\varepsilon)\theta)\right) v,$$

and

$$f(e_1 \mid \mathsf{OPT} - o_{\mathrm{r}} + e_2) \geq f(\{e_1, e_2\}) - f(\mathsf{OPT} - o_{\mathrm{r}} + e_2) \geq f(\{e_1, e_2\}) - \frac{2}{3}v.$$

As $f(e_1 \mid \mathsf{OPT} - o_{\mathrm{r}}) \geq f(e_1 \mid \mathsf{OPT} - o_{\mathrm{r}} + e_2)$ by submodularity, the above two inequalities imply that $f(\{e_1, e_2\}) \leq (1/3 + (1+\varepsilon)\theta)v$, contradicting Line 4 of the algorithm.

□

The next lemma states that if o_{r} is thrown away by the algorithm, e_1 itself is already good for approximation.

Lemma 5.5.6. *If $f(\{e_1, o_r\}) \leq (1/3 + (1+\varepsilon)\theta)v$, then we have $f(\mathsf{OPT} - o_r + e_1) \geq (2/3 - \varepsilon)v$.*

Proof. Suppose, for a contradiction, that $f(\mathsf{OPT} - o_r + e_1) < (2/3 - \varepsilon)v$. Then

$$\begin{aligned} f(o_r \mid \mathsf{OPT} - o_r + e_1) &= f(\mathsf{OPT} + e_1) - f(\mathsf{OPT} - o_r + e_1) \\ &\geq v - \left(\frac{2}{3} - \varepsilon\right) v = \left(\frac{1}{3} + \varepsilon\right) v. \end{aligned}$$

On the other hand, it holds that

$$f(o_r \mid e_1) = f(\{o_r, e_1\}) - f(e_1) \leq \left(\frac{1}{3} + (1+\varepsilon)\theta - \theta\right) v = \left(\frac{1}{3} + \varepsilon\theta\right) v.$$

By submodularity, $f(o_r \mid \mathsf{OPT} - o_r + e_1) \leq f(o_r \mid e_1)$ and the above two inequalities lead to a contradiction.

□

By the previous two lemmas, one of the red items in the returned set S, along with $\mathsf{OPT} - o_r$, gives $(2/3 - O(\varepsilon))v$. We then can define Γ as $2/3$ in the interval $\theta \in [1/2, 2/3)$.

5.5.2 Multiple-Pass $(2/5 - \varepsilon)$-Approximation Algorithm

In this section, we show that when $c(o_1) \geq K/2$, we can use multiple passes to get a $(2/5 - \varepsilon)$-approximation. Let $\mathsf{OPT} = \{o_1, o_2, \ldots, o_\ell\}$ be an optimal solution with $c(o_1) \geq c(o_2) \geq \ldots \geq c(o_\ell)$. Suppose that $c_1 \in \mathbb{R}_+$ satisfies $1/2 \leq c_1/(1+\varepsilon) \leq c(o_1)/K \leq c_1$. Such c_1 can be guessed with $O(\varepsilon^{-1})$ space by a geometric series $\{(1+\varepsilon)^i/2 \mid i \in \mathbb{Z}_+\}$.

First we observe the following claims.

Lemma 5.5.7. *When $c(o_1) \geq K/2$, we may assume that $\frac{3}{10} f(\mathrm{OPT}) < f(o_1) < \frac{2}{5} f(\mathrm{OPT})$.*

Proof. If $f(o_1) \geq \frac{2}{5} f(\mathsf{OPT})$, then Algorithm 2 returns a 2/5-approximate solution. So we may assume that $f(o_1) < \frac{2}{5} f(\mathsf{OPT})$.

Suppose to the contrary that $f(o_1) < \frac{3}{10} f(\mathsf{OPT})$. This implies $f(\mathsf{OPT} - o_1) \geq \frac{7}{10} f(\mathsf{OPT})$. Consider the problem of maximizing $f(S)$ subject to $c(S) \leq K/2$ in the set $\{e \in E \mid c(e) \leq K/2\}$. Since the optimal value is at least $f(\mathsf{OPT} - o_1) > \frac{7}{10} f(\mathsf{OPT})$, by applying the bicriteria approximation algorithm in Theorem 5.4.1 with $p = 2$, we obtain a solution $\tilde{S}$ satisfying

$$f(\tilde{S}) \geq \left(\frac{4}{7} - O(\varepsilon)\right) \frac{7}{10} f(\mathsf{OPT}) \geq \left(\frac{2}{5} - O(\varepsilon)\right) f(\mathsf{OPT}).$$

Thus the lemma holds.

□

Lemma 5.5.8. *We may assume that $c(o_1) \leq (1+\varepsilon)\frac{2}{3}K$.*

Proof. Suppose not. That is, suppose $c(o_1) > (1+\varepsilon)\frac{2}{3}K$. By Lemma 5.5.7, we may assume that $f(o_1) < \frac{2}{5} f(\mathsf{OPT})$, and hence $f(\mathsf{OPT} - o_1) > \frac{3}{5} f(\mathsf{OPT})$ by submodularity.

Consider the problem of maximizing $f(S)$ subject to $c(S) \le (1 - \frac{c_1}{1+\varepsilon})K$. Since $c(\mathsf{OPT} - o_1) \le K - c(o_1) \le (1 - \frac{c_1}{1+\varepsilon})K$, the set $\mathsf{OPT} - o_1$ is a feasible solution of the problem. Now apply the bicriteria approximation in Theorem 5.4.1 with $p = (1 - \frac{c_1}{1+\varepsilon})^{-1} \ge 3$. Then, since $p \ge 3$ and $f(\mathsf{OPT} - o_1) > \frac{3}{5} f(\mathsf{OPT})$, the output $\tilde{S}$ satisfies that

$$f(\tilde{S}) \ge \left(\frac{2p}{2p+3} - O(\varepsilon)\right) f(\mathsf{OPT} - o_1) \ge \left(\frac{2}{5} - O(\varepsilon)\right) f(\mathsf{OPT}).$$

Thus the lemma holds.

□

We use the first pass to estimate $f(\mathsf{OPT})$ as follows. For an error parameter $\varepsilon \in (0, 1]$, perform the single-pass algorithm in Theorem 5.2.5 to get a $(1/3 - \varepsilon)$-approximate solution $S \subseteq E$, which can be used to upper bound the value of $f(\mathsf{OPT})$, that is, $f(S) \le f(\mathsf{OPT}) \le (3 + \varepsilon) f(S)$. We then find the geometric series to guess its exact value. Thus, we may assume that we are given the value v satisfying $v \le f(\mathsf{OPT}) \le (1 + \varepsilon)v$ with $O(\varepsilon^{-1})$ space.

Below we show how to obtain a solution of value at least $(2/5 - O(\varepsilon))v$, using two more passes. Before we start, we introduce a slightly modified versions of the algorithms presented in Section 5.2; it will be used as a subroutine.

Lemma 5.5.9. *Consider the problem (5.1) with the knapsack capacity K'. Let $h \in \mathbb{R}_+$, and suppose that Algorithms 1 and 2 are modified as follows:*

- *(At Line 4 in Algorithm 1) A new item e is added into the current set S only if $\frac{f(e|S)}{c(e)} \ge \frac{\alpha v - f(S)}{hK' - c(S)}$ and $c(S + e) \le hK'$.*
- *(At Line 4 in Algorithm 2) A new item e is taken into account only if $c(e) \le hK'$.*

Then, the best returned set $\tilde{S}$ of the two algorithms with $\alpha = \frac{2h}{h+2}$ satisfies that $c(\tilde{S}) \le hK'$ and $f(\tilde{S}) \ge \frac{h}{h+2} v$.

Proof. Let v' be an approximation to the optimal value $f(\mathsf{OPT}')$ of the problem (5.1) with the knapsack capacity K'. It is straightforward to to check that Lemma 5.2.1 holds with slight variations: (1) $f(S) \ge \frac{\alpha v' c(S)}{hK}$ where S is the current set, and (2) if an item e fails the marginal-ratio threshold, then $f(e \mid \tilde{S}) < \frac{\alpha v' c(e)}{hK}$.

If there is no bad item, then $v' \le f(\mathsf{OPT}') \le f(\tilde{S}) + \sum_{e \in \mathsf{OPT}' \setminus \tilde{S}} f(e \mid \tilde{S}) \le f(\tilde{S}) + \frac{\alpha v'}{h}$, implying that $f(\tilde{S}) \ge (1 - \frac{\alpha}{h}) v'$. If there is a bad item, then the set S just before some bad item e arrives satisfies that $f(S + e) \ge \alpha v'$. Hence $f(\tilde{S})$ or some singleton has the value at least $\alpha v'/2$. Therefore, when $\alpha = \frac{2h}{h+2}$, the lower bound is maximized and the ratio in this case is $\frac{h}{h+2}$.

□

Let all items $e \in E$ whose sizes $c(e)$ satisfy $c_1/(1 + \varepsilon) \le c(e)/K \le c_1$ be the red items. By Theorem 5.5.1, we can select a set S of the red items so that one of them guarantees $f(\mathsf{OPT} - o_1 + e) \ge (\Gamma(\theta) - O(\varepsilon))v$, where θ satisfies the condition in Theorem 5.5.1. Note that θ can be guessed by a geometric series from the interval $[\frac{3}{10} v, \frac{2}{5}(1 + \varepsilon)v]$ by Lemma 5.5.7. The space required is $O(\varepsilon^{-1})$.

Algorithm 8

1: **procedure** MultiPassKnapsack$(\varepsilon, v, \theta, c_1)$ ▷ $\varepsilon \in (0,1]$, $v \in \mathbb{R}_+$, and $\theta, c_1 \in [0,1]$.
2: Use the algorithm in Theorem 5.5.1 to choose a set S of items e with $c_1/(1+\varepsilon) \le c(e)/K \le c_1$ so that one of them $e \in S$ satisfies $f(\mathsf{OPT} - o_1 + e) \ge (\Gamma(\theta) - O(\varepsilon))v$.
3: **for** each item $e \in S$ **do**
4: Define a submodular function $g_e(\cdot) = f(\cdot \mid e)$.
5: Apply the marginal-ratio thresholding algorithm (Lemma 5.5.9) with regard to function g_e, where

$$h = \frac{1-c_1}{1-(c_1/(1+\varepsilon))} \text{ and } K' = \left(1 - \frac{c_1}{1+\varepsilon}\right) K.$$

6: Let the resultant set be S_e.
7: **return** the solution $S_e \cup \{e\}$ with $\max_{e \in S} f(S_e + e)$.

In the next pass, for each $e \in S$, define a new monotone submodular function $g_e(\cdot) = f(\cdot \mid e)$ and apply the modified thresholding algorithm (Lemma 5.5.9) with $h = \frac{1-c_1}{1-(c_1/(1+\varepsilon))}$ and $K' = (1 - (c_1/(1+\varepsilon))K$. Let S_e be the output of the modified thresholding algorithm. Then our algorithm returns the solution $S_e \cup \{e\}$ with $\max_{e \in S} f(S_e + e)$. The detail is given in Algorithm 8.

The returned solution has size at most K, since $c(S_e) \le (1-c_1)K$ by Lemma 5.5.9. Moreover, it follows that the returned solution $\tilde{S}$ satisfies that $f(\tilde{S}) \ge (2/5 - O(\varepsilon))v$.

Theorem 5.5.10. *For $\varepsilon \in (0,1]$, suppose that $v \le f(\mathsf{OPT}) \le (1+\varepsilon)v$, $1/2 \le c_1/(1+\varepsilon) \le c(o_1)/K \le c_1$, and θ satisfies the condition in Theorem 5.5.1. After running* MultiPassKnapsack$(\varepsilon, v, \theta, c_1)$*, there exists an item $e \in S$ chosen in Line 2, which, along with S_e collected in Line 6, gives $f(S_e + e) \ge (2/5 - O(\varepsilon))v$.*

Proof. By Theorem 5.5.1, one item $e \in S$ has $f(\mathsf{OPT} - o_1 + e) \ge (\Gamma(\theta) - O(\varepsilon))v$ and $f(e) \ge \theta v/(1+\varepsilon)$.

Consider the problem of maximizing $g_e(S)$ subject to $c(S) \le h^{-1}(1-c_1)K$ with available space $(1-c_1)K$. Since $c(\mathsf{OPT} - o_1) \le K - c(o_1) \le h^{-1}(1-c_1)K$, the set $\mathsf{OPT} - o_1$ is a feasible solution of this problem. Therefore, it follows from Lemma 5.5.9 that the obtained solution S_e satisfies that

$$g_e(S_e) \ge \left(\frac{h}{h+2} - O(\varepsilon)\right) g_e(\mathsf{OPT} - o_1) \ge \left(1 - \frac{2}{h+2} - O(\varepsilon)\right) g_e(\mathsf{OPT} - o_1).$$

Now we have

$$h = 1 - \frac{\varepsilon c_1}{1+\varepsilon-c_1} \ge 1 - \frac{c_1}{1-c_1}\varepsilon = 1 - O(\varepsilon),$$

since $\frac{c_1}{1-c_1}$ is a constant by Lemma 5.5.8. Therefore, we have

$$g_e(S_e) \ge \left(\frac{1}{3} - O(\varepsilon)\right) g_e(\mathsf{OPT} - o_1) = \left(\frac{1}{3} - O(\varepsilon)\right) (f(\mathsf{OPT} - o_1 + e) - f(e)).$$

It follows that the output $S_e + e$ satisfies that

$$f(S_e + e) = f(e) + g_e(S_e) \quge f(e) + \frac{1}{3}(1 - O(\varepsilon))\left((\Gamma(\theta) - O(\varepsilon))v - f(e)\right)$$

$$
\begin{aligned}
&\geq \frac{2}{3} f(e) + \frac{1}{3}(1 - O(\varepsilon))\Gamma(\theta) v \\
&\geq \left(\frac{2}{3}\theta + \frac{1}{3}\Gamma(\theta) \right) (1 - O(\varepsilon)) v
\end{aligned}
$$

where the last inequality holds since $f(e) \geq \theta v/(1+\varepsilon)$. When $\theta \geq 1/2$, we have $\Gamma(\theta) \geq 2/3$ by Theorem 5.5.1, and hence the ratio is more than $2/5$. Consider the case when $\theta < 1/2$. We observe that $\frac{2}{3}\theta + \frac{1}{3}\Gamma(\theta)$ is a non-decreasing function. Hence the minimum is attained when $\theta = 3/10$ by Lemma 5.5.7. The ratio is bounded by the linear function in (5.13) when $t = 2$, and hence it is at least $2/5$.

□

The next theorem summarizes our results in this section.

Theorem 5.5.11. *Suppose that $c(o_1) > K/2$. There exists an algorithm that uses* MultiPassKnapsack *as a subroutine so that it returns $(2/5 - \varepsilon)$-approximation with 3 passes for the problem (5.1). The space complexity of the algorithm is $O(K\varepsilon^{-3})$.*

Chapter 6

Popularity, Mixed Matchings, and Self-duality

This paper appeared in SODA 2017. It is joint-work with Kavitha Telikepalli.

Abstract We consider the problem of matching applicants to jobs under two-sided preferences in a popular and utility-optimal manner. Our input instance is a bipartite graph $G = (A \cup B, E)$ with a utility function $w : E \to \mathbb{Q}$ where each vertex $u \in A \cup B$ has a preference list ranking its neighbors in a strict order of preference. For any two matchings M and T in G, let $\phi(M, T)$ be the the number of vertices that prefer M to T. A matching M is popular if $\phi(M, T) \geq \phi(T, M)$ for all matchings T in G. A mixed matching is defined as $\Pi = \{(M_0, p_0), \ldots, (M_k, p_k)\}$ for some matchings $M_0, \ldots, M_k$ and $\sum_{i=0}^{k} p_i = 1$, $p_i \geq 0$ for all i. The function $\phi(\cdot, \cdot)$ easily extends to mixed matchings; a mixed matching Π is popular if $\phi(\Pi, \Lambda) \geq \phi(\Lambda, \Pi)$ for all mixed matchings Λ in G.

Motivated by the fact that a popular mixed matching could have a much higher utility than all popular matchings, we study the popular fractional matching polytope $\mathcal{P}_G$. Our main result is that this polytope is half-integral (and in the special case where a stable matching is a perfect matching, this polytope is integral), implying that there is always a max-utility popular mixed matching Π such that $\Pi = \{(M_0, \frac{1}{2}), (M_1, \frac{1}{2})\}$ and Π can be computed in polynomial time. An immediate consequence is that in order to implement a max-utility popular mixed matching in G, we need just *one single* random bit.

We analyze $\mathcal{P}_G$ whose description may have exponentially many constraints via an extended formulation in $m + n$ dimensions with $O(m + n)$ constraints. The linear program that gives rise to this formulation has an unusual property: self-duality. In other words, this linear program is exactly identical to its dual program. This is a rare case where an LP of a natural problem has such a property. The self-duality of the LP plays a crucial role in our proof of the half-integrality of $\mathcal{P}_G$.

We also show that our result carries over to the roommates problem, where the given

graph G may not be bipartite. The polytope of popular fractional matchings is still half-integral here and thus we can compute a max-utility popular half-integral matching in G in polynomial time. To complement this result, we also show that the problem of computing a max-utility popular (integral) matching in a roommates instance G is NP-hard.

6.1 Introduction

Let $G = (A \cup B, E)$ be a bipartite graph on n vertices and m edges where A is called the set of applicants, B is called the set of jobs, and every vertex $u \in A \cup B$ has a preference list ranking its neighbors in a strict order of preference. Such a graph G is also referred to as an instance of the *stable marriage* problem with strict and possibly incomplete preference lists. Moreover, a utility function $w : E \to \mathbb{Q}$ is given, where $w(a, b)$ is the utility of matching applicant a with job b. Our goal is to match applicants to jobs such that this matching is *popular* and has the maximum utility, where the utility $w(M)$ of a matching M is the sum of utilities of all edges in M.

The notion of popularity was introduced by Gärdenfors [74] in 1975. A vertex $u \in A \cup B$ *prefers* matching M to matching M' if u is matched in M and unmatched in M' or it is matched in both and $M(u)$ ranks higher than $M'(u)$ in u's preference list. For any two matchings M and M' in G, let $\phi(M, M')$ be the number of vertices that prefer M to M'. We say M is *more popular than* M' if $\phi(M, M') > \phi(M', M)$.

Definition 6.1.1. *A matching M is* popular *if $\phi(M, M') \geq \phi(M', M)$ for every matching M' in G, i.e., $\Delta(M, M') \geq 0$ where $\Delta(M, M') = \phi(M, M') - \phi(M', M)$.*

Thus a popular matching never loses an election (where vertices cast votes) and so a popular matching can be considered to be "globally stable" since an election cannot force a migration from a popular matching to any other matching. The notion of popularity is naturally appealing and is less demanding than the notion of *stability*. A matching is stable if it has no blocking edges: an edge (a, b) blocks matching M if both a and b prefer each other to their respective assignments in M. The existence of stable matchings and the Gale-Shapley algorithm [70] to find one are classical results in algorithms. It is easy to show that every stable matching is popular [74].

There are many polynomial time algorithms to compute a max-utility stable matching in G [47, 48, 53, 96, 151, 165, 169] – several of these use linear programming on the stable matching polytope $\mathcal{S}_G$, which is the convex hull of the 0-1 edge incidence vectors of stable matchings in G. The utility of a max-utility popular matching could be much more than that of a max-utility stable matching; for instance, when all utilities are 1, a max-utility popular matching is the same as a max-size popular matching and it is known that a stable matching is a *min-size* popular matching [89].

Mixed matchings. Our objective is to find a max-utility matching under the constraints of popularity — so as to achieve highest utility, what we seek should be a popular *mixed matching* rather than a popular (pure) matching. A mixed matching is a probability distribution over matchings, i.e., a mixed matching $\Pi = \{(M_0, p_0), \ldots, (M_k, p_k)\}$, where $M_0, \ldots, M_k$ are matchings in G and $\sum_{i=0}^{k} p_i = 1$, $p_i \geq 0$ for all i, and the utility of Π is $\sum_{i=0}^{k} p_i \cdot w(M_i)$.

In economics, mixed matchings are called random assignments and they are used in the design of mechanisms to guarantee various desirable properties, such as efficiency, fairness, and strategy-proofness (see [13, 107] and the references therein). The function $\phi(M, M')$ defined earlier easily extends to $\phi(\Pi, M')$ as follows: $\phi(\Pi, M') = \sum_{i=0}^{k} p_i \cdot \phi(M_i, M')$, where $\Pi = \{(M_0, p_0), \ldots, (M_k, p_k)\}$. The definition of $\phi(M', \Pi))$ is analogous.

Definition 6.1.2. *A mixed matching Π is popular if $\phi(\Pi, M') \geq \phi(M', \Pi)$ for all matchings M' in G, i.e., $\Delta(\Pi, M') \geq 0$, where $\Delta(\Pi, M') = \phi(\Pi, M') - \phi(M', \Pi)$.*

Suppose $\Lambda = \{(N_0, q_0), \cdots, (N_h, q_h)\}$ is another mixed matching. Let $\Delta(\Pi, \Lambda) = \sum_{i=1}^{k} \sum_{j=1}^{h} p_i q_j \Delta(M_i, N_j)$. Then it follows easily from definition that if Π is popular then $\Delta(\Pi, \Lambda) \geq 0$ for all *mixed* matchings Λ in G. Thus a popular mixed matching "beats" all integral and mixed matchings. As an allocation mechanism, a popular mixed matching has several nice properties, such as population-consistency and composition-consistency. We refer the readers to [46] for details.

A popular mixed matching need not be a probability distribution over popular matchings in G. Such an example was shown in [104] and we show a much simpler example below in Fig. 6.1. Here $A = \{a_0, a_1, a_2\}$, $B = \{b_1, b_2\}$, and $E = A \times B$. The preference list of every applicant is the same: $b_1 \succ b_2$, i.e., b_1 is the top choice and then b_2; the preference list of every job is the same: $a_1 \succ a_2 \succ a_0$. This instance admits exactly one popular matching: this is the stable matching $S = \{(a_1, b_1), (a_2, b_2)\}$ (the blue matching in Fig. 6.1). Consider the mixed matching $\Pi = \{(S, \frac{1}{2}), (M, \frac{1}{2})\}$ where $M = \{(a_1, b_2), (a_2, b_1)\}$ (the red matching in Fig. 6.1). Note that Π is outside the convex hull of popular matchings – we will show in Section 6.2 that Π is popular. Whenever $w(a_1, b_2) + w(a_2, b_1)$ is larger than $w(a_1, b_1) + w(a_2, b_2)$, the utility of Π is higher than that of S. Thus the utility of a max-utility popular mixed matching can be much higher than that of a popular matching.

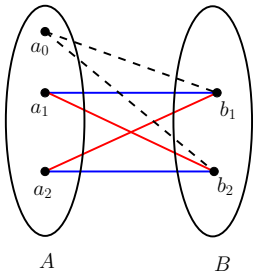

Figure 6.1: The blue matching $S = \{(a_1, b_1), (a_2, b_2)\}$ is the only popular matching here. The red matching $M = \{(a_1, b_2), (a_2, b_1)\}$ is not popular as $\{(a_0, b_2), (a_1, b_1)\}$ is more popular than M.

Mixed matchings are closely related to *fractional* matchings. A fractional matching $\vec{p}$ in G is a point in $\mathbb{R}^m_{\geq 0}$ that satisfies $\sum_{e \in E(v)} p_e \leq 1$, for every vertex v, where $E(v)$ is the set of edges incident on vertex v. In a bipartite graph, a mixed matching is equivalent to a fractional matching (Birkhoff-von Neumann theorem). The polytope $\mathcal{P}_G \subseteq \mathbb{R}^m$ of all popular mixed matchings in $G = (A \cup B, E)$ involves exponentially many constraints. However a compact extended formulation of this polytope was given in [105]. Thus a max-utility popular mixed matching can be computed in polynomial time.

However a potential drawback of generalizing from matchings to mixed matchings is that the optimal solution has become more complex to describe and more difficult to implement. A mixed matching can be interpreted as either a lottery over matchings or a time-sharing arrangement (when the mixed matching is viewed as a fractional matching) [150]: in the former case, we need access to several random bits to implement a lottery and the latter case involves sub-dividing jobs and assigning several fractional jobs to an applicant. Thus we may need to deal with an unstructured optimal solution. Our first result is the following.

Theorem 6.1.3. *Given an instance $G = (A \cup B, E)$ with strict preference lists and a utility function $w : E \to \mathbb{Q}$, G always has a max-utility popular mixed matching $\Pi = \{(M_0, \frac{1}{2}), (M_1, \frac{1}{2})\}$ and Π can be computed in polynomial time. Moreover, if G admits a perfect stable matching, i.e. a stable matching where no vertex is unmatched, then $\Pi = \{(M, 1)\}$, namely, Π is pure.*

Thus our result implies that to achieve max-utility, we just need a *single random bit* to implement the lottery or we can find a time-sharing arrangement that is simple and organized—every vertex has at most two partners and spends the same amount of time with each. Hence we can find a max-utility popular mixed matching that is highly structured. Note that it was already known how to compute a max-utility popular half-integral matching in G in polynomial time [104]. However it was not known whether this was a max-utility popular fractional matching or not. Our main contribution is to show that this is so by proving that the polytope $\mathcal{P}_G$ is half-integral. Moreover, $\mathcal{P}_G$ is integral when G admits a perfect stable matching (say, $|A| = |B|$ and preference lists are complete).

The complexity of finding a max-utility popular matching in $G = (A \cup B, E)$ is currently unknown. We show here that when G is a general graph, i.e., not necessarily bipartite (also called a *roommates* instance with strict preferences), the max-utility popular matching problem becomes NP-hard. In fact, the max-utility stable matching problem in a roommates instance is also NP-hard, as shown by Feder [47].

Theorem 6.1.4. *Given $G = (V, E)$ where each vertex has a strict preference list over its neighbors and a utility function $w : E \to \mathbb{Q}$, the decision question of computing a max-utility popular matching in G is NP-complete. Moreover, for any constant $\epsilon > 0$, it is UGC-hard to design a polynomial time ϵ-approximation algorithm for this problem.*

We show below that when integrality is relaxed to half-integrality, the above problem becomes tractable. Recall that a fractional matching needs not be a mixed matching in a non-bipartite graph but it is always a convex combination of half-integral matchings. Analogous to a popular mixed matching in a bipartite graph, a popular fractional matching in a general graph is a fractional matching that "beats" all other fractional matchings. Formally, if node v prefers node v' to node v'' or $v' = v''$, we write $v' \succeq_v v''$; if only the former, we write $v' \succ v$. We slighly abuse notation by writing $v' \succ_v \emptyset$ to denote that v prefers to be matched to any node v' than being unmatched.

Given a half-integral matching $M \in \{0, 1/2, 1\}^m$, each node v in M is either fully-matched, half-matched, or unmatched. In the first case, let $M(v, 1)$, $M(v, 2)$ be the two (not necessarily distinct) vertices that v is fractionally matched to and let $M(v, 1) \succeq_v M(v, 2)$; in the second case, $M(v, 1)$ is the node v is matched to and $M(v, 2) = \emptyset$; in the last case, $M(v, 1) = M(v, 2) = 0$. Given two

half-integral matchings M and M', $\phi(M, M') = \sum_{i=1}^{2} |\{v | M(v,i) \succ_v M'(v,i)\}| - \{v | M'(v,i) \succ_v M(v,i)\}|$. Let $p = \sum_{i=0}^{k} p_i M_i$, where $\sum_{i=0}^{k} p_i = 1$, be a fractional matching. Then $\phi(p, M') = \sum_{i=0}^{k} p_i \phi(M_i, M')$. The definition of $\phi(M', p)$ is analogous.

We can now define a popular fractional matching analogously to Definition 6.1.2.

Definition 6.1.5. *A fractional matching $p = \sum_{i=0}^{k} p_i M_i$, where $\sum_{i=0}^{k} p_i = 1$, is popular if $\phi(p, M') \geq \phi(M', p)$ for all half-integral matchings M' in G, i.e., $\Delta(p, M') \geq 0$, where $\Delta(p, M') = \phi(p, M') - \phi(M', p)$.*

Let $q = \sum_{j=0}^{h} q_j N_j$, where $\sum_{j=0}^{h} q_j = 1$, be another fractional matching. Furthermore, let $\Delta(p, q) = \sum_{i=1}^{k} \sum_{j=1}^{h} p_i q_j \Delta(M_i, N_j)$. It is clear that if p is popular, $\Delta(p, q) \geq 0$ for all *fractional* matchings q. We show that there is a max-utility popular fractional matching that is half-integral.

Theorem 6.1.6. *Given an instance $G = (V, E)$ with strict preference lists and a utility function $w : E \to \mathbb{Q}$, G always has a max-utility popular fractional matching $p \in \{0, \frac{1}{2}, 1\}^m$, and p can be computed in polynomial time.*

6.1.1 Our Techniques

The polytope $\mathcal{P}_G$ of popular fractional matchings has a compact extended formulation $\mathcal{P}'_G \subseteq \mathbb{R}^{m+n}$ and in this paper, we analyze the linear program that gives rise to $\mathcal{P}'_G$ and discover an unusual property of this linear program. This LP is *self-dual*, i.e., it is exactly identical to its dual program.[1] To the best of our knowledge, this seems to be the first time a natural problem has an LP with this property and our proof on the structure of $\mathcal{P}_G$ uses this self-duality crucially. Prior to our result, the only linear program that had a somewhat similar property is the one used by Roth, Rothblum and Vande Vate [150] to describe the stable matching polytope $\mathcal{S}_G$. Every stable fractional matching is an optimal solution to their original LP there but also gives rise to an optimal solution in its dual. In the original words of Roth et al. [150]: *"We know of no similarly rich class of linear programs whose primal and dual solutions are related in this way."*

By setting the n new variables (the ones added for the extended formulation) to 0, the description of $\mathcal{P}'_G$ reduces to the description of $\mathcal{S}_G$. This formulation of $\mathcal{S}_G$ is not independent of other descriptions of this polytope [150, 165], however it has a novel and intuitive interpretation where stability is interpreted as follows: the "sum of votes" of any adjacent pair of vertices for each other is less than or equal to 0 (see Section 6.2 for details).

Our technique to prove the integrality of $\mathcal{P}_G$ in the case when $G = (A \cup B, E)$ admits a perfect stable matching (one that matches all vertices) is inspired by the one used by Teo and Sethuraman [165] to show the integrality of $\mathcal{S}_G$. However as the description of the extended popular fractional matching polytope $\mathcal{P}'_G$ is more general than that of $\mathcal{S}_G$, our task is more involved here. We use the fact that G admits a perfect stable matching to first conclude that for any integral matching M here, there is a *witness* $(\alpha_u)_{u \in A \cup B} \in \{\pm 1\}^n$ to the popularity of M, where the α_u's are the n new variables that were added for the extended formulation. In order to find popular matchings in G whose convex combination forms a popular fractional matching $\vec{x}$, we use the

[1] We remark that the same LP was first introduced in [105]. However, its self-duality is observed and explicitly used the first time in the present work.

following new idea: for any vertex u, use the value of α_u to divide into two sub-arrays the ordered array of partners that u gets assigned in $\vec{x}$ and swap these two sub-arrays to get a reordered array.

We obtain the matchings $M_1, \ldots, M_k$ that form $\vec{x}$ from these reordered arrays. The popularity of these matchings will be proved by assigning an appropriate witness $\vec{\alpha}^i \in \{\pm 1\}^n$ to each M_i and showing that each M_i along with its witness $\vec{\alpha}^i$ belongs to $\mathcal{P}'_G$. The half-integrality of $\mathcal{P}_G$ for the general case (when G has no perfect stable matching) and for non-bipartite graphs follows from the integrality of $\mathcal{P}_G$ in this special case. For non-bipartite graphs, we show a simple reduction from the vertex cover problem to show that it is NP-hard to compute a max-utility popular matching.

Background and related work. As mentioned earlier, popular matchings were introduced by Gärdenfors [74] for two-sided preferences, i.e., vertices on both sides of the graph have preferences over their neighbors. For one-sided preferences (i.e., vertices of only one side have preferences over their neighbors), similar notions have been suggested by Kreweras [113] and Fishburn [51]. Popular matchings need not always exist in this setting and an efficient algorithm was given in [3] to determine if the given instance admits a popular matching or not. It was shown in [105] that popular mixed matchings always exist in the same setting and such a mixed matching can be efficiently computed. Popular mixed matchings as an allocation mechanism have been studied in [46, 80, 81].

In the domain of two-sided preferences, when preferences involve ties, the problem of determining if $G = (A \cup B, E)$ admits a popular matching or not is NP-hard [28, 139]. When preference lists are strict, popular matchings always exist in G and efficient algorithms for computing a max-size popular matching were given in [89, 103]. A subclass of max-size popular matchings called dominant matchings were studied in [29]. The convex hull of the $\{0, \frac{1}{2}, 1\}$-edge incidence vectors of popular half-integral matchings in G was described in [104] – this was done via the stable matching polytope of a larger graph G^* that was constructed using G.

Stable matchings have been extensively studied and there are several monographs [79, 110, 128, 147] on this subject. In practice, Roth [148, 149] discusses how stable matchings are compared with other types of matchings in the two-sided matching markets. The first description of the polytope $\mathcal{S}_G$ for $G = (A \cup B, E)$ was given by Vande Vate [169] in 1989 and several descriptions of $\mathcal{S}_G$ are now known [53, 140, 150, 151, 165]. Stable matchings need not exist in the roommates problem and efficient algorithms to determine if $G = (V, E)$ admits a stable matching or not was given in [97, 165]. Stable half-integral matchings always exist in G and it was shown in [2] that the polytope of all stable fractional matchings is half-integral.

Organization of the paper. In Section 6.2 we describe the extended popular fractional matching polytope $\mathcal{P}'_G$ and show the self-duality of the LP that gives rise to this description. Section 6.3 shows the integrality of the popular fractional matching polytope $\mathcal{P}_G$ when G is bipartite and admits a perfect stable matching. Section 6.4 shows the half-integrality of $\mathcal{P}_G$ in the general bipartite graph. The proofs of Theorems 6.1.4 and 6.1.6 are given Sections 6.5 and 6.6 respectively.

6.2 The extended popular fractional matching polytope $\mathcal{P}'_G$

Let $\mathcal{M}_G$ be the matching polytope of $G = (A \cup B, E)$: so $\mathcal{M}_G = \{x \in \mathbb{R}^m : \sum_{e \in E(u)} x_e \leq 1 \ \forall u \in A \cup B \text{ and } x_e \geq 0 \ \forall e \in E\}$, where $E(u)$ is the set of all edges incident on u in G. The popular fractional matching polytope of G is

$$\mathcal{P}_G = \{\vec{x} \in \mathcal{M}_G : \Delta(\vec{x}, M) \geq 0 \ \ \forall \text{matchings } M \text{ in } G\},$$

where $\Delta(\vec{x}, M) = \Delta(\Pi, M)$ where Π is a mixed matching corresponding to the fractional matching $\vec{x}$ (recall that in a bipartite graph, a mixed matching is equivalent to a fractional matching). Alternately, $\Delta(\vec{x}, M) = \sum_{u \in A \cup B} \mathsf{vote}_u(\vec{x}, M(u))$, where the function $\mathsf{vote}_u(\cdot, \cdot)$ is defined as follows.

For any vertex $u \in A \cup B$ and neighbors v, v' of u, we have $\mathsf{vote}_u(v, v') = 1$ if u prefers v to v', it is -1 if u prefers v' to v, else it is 0 (i.e., $v = v'$). The function $\mathsf{vote}_u(v, v')$ extends by linearity to $\mathsf{vote}_u(\vec{x}, v')$ where $\vec{x}$ is a fractional matching:

$$\begin{aligned} \mathsf{vote}_u(\vec{x}, v') &= \sum_v x_{(u,v)} \cdot \mathsf{vote}_u(v, v') \\ &= \sum_{v:\, v \succ_u v'} x_{(u,v)} - \sum_{v:\, v \prec_u v'} x_{(u,v)}, \end{aligned}$$

where $\{v : v \succ_u v'\}$ consists all neighbors of u that are ranked higher than v' in u's preference list and the set $\{v : v \prec_u v'\}$ consists of those who are ranked lower. We also let $\mathsf{vote}_u(v', x) = -\mathsf{vote}_u(x, v')$.

The description of $\mathcal{P}_G$ in (6.1) above involves possibly exponentially many constraints – one for each matching in G. Hence we will use the *extended* popular fractional matching polytope $\mathcal{P}'_G$ of G in $m + n$ dimensions (from [105]) that uses n new variables α_u for $u \in A \cup B$ along with the m variables x_e for $e \in E$. Note that the edge utilities given in the input instance G play no part in the description of either $\mathcal{P}_G$ or $\mathcal{P}'_G$.

It will be convenient to assume that each vertex $u \in A \cup B$ is completely matched in $\vec{x}$. So each vertex u gets matched to a distinct artificial vertex $\ell(u)$ (referred to as the last resort neighbor of u) placed at the bottom of u's preference list with $x_{(u,\ell(u))} = 1 - \sum_{e \in E(u)} x_e$. Let $\tilde{E}$ denote the edge set $E \cup \{(u, \ell(u)) : u \in A \cup B\}$ and let $\tilde{E}(u)$ be the set of edges in $\tilde{E}$ that are incident on u. Then $\sum_{e \in \tilde{E}(u)} x_e = 1$ for all $u \in A \cup B$. For convenience, we will continue to use $\vec{x}$ to denote the revised $\vec{x}$ in $[0, 1]^{m+n}$.

The graph $\tilde{G}_x$. The graph $\tilde{G}_x$ is the graph G augmented with last resort vertices and with edge set $\tilde{E}$ where the weight of edge (a, b) is equal to $\mathsf{vote}_a(b, \vec{x}) + \mathsf{vote}_b(a, \vec{x})$ (note that in case a or b is $l(u)$ for some $u \in A \cup B$, $\mathsf{vote}_a(b, \vec{x})$ or $\mathsf{vote}_b(a, \vec{x})$ is understood to be 0). For any matching M in $\tilde{G}_x$ that matches all vertices in $A \cup B$, observe that the weight of M in $\tilde{G}_x$ is exactly the same as $\Delta(M, \vec{x})$.

The polytope $\mathcal{P}'_G$ is based on the following observation: $\vec{x} \in \mathcal{M}_G$ is popular if and only if the maximum weight of a matching in the graph $\tilde{G}_x$ that matches all vertices in $A \cup B$ is 0; note that this value is always non-negative since the weight of the fractional matching $\vec{x}$ in $\tilde{G}_x$ is 0. Consider

the dual program (in variables α_u for $u \in A \cup B$) to the primal program which is the maximum weight matching problem in the graph $\tilde{G}_x$ that matches all vertices in $A \cup B$. This is LP1 described below.

$$\textbf{(LP1)} \quad \text{minimize} \sum_{u \in A \cup B} \alpha_u$$

$$\alpha_a + \alpha_b \quad \geq \quad \mathsf{vote}_a(b, \vec{x}) + \mathsf{vote}_b(a, \vec{x}) \quad \forall (a, b) \in \tilde{E}.$$

For any point $\vec{x} \in \mathcal{P}_G$, by LP-duality, the optimal dual value is 0 and so there is the optimal solution $\vec{\alpha}^x = (\alpha^x_u)_{u \in A \cup B}$ to LP1 such that $\sum_{u \in A \cup B} \alpha^x_u = 0$. We will regard $\vec{\alpha}^x$ as the *witness* to the popularity of $\vec{x}$. For example, consider the instance in Fig. 6.1: the half-integral matching $\vec{p} = (S + M)/2$ where $S = \{(a_1, b_1), (a_2, b_2)\}$ and $M = \{(a_1, b_2), (a_2, b_1)\}$ is popular as witnessed by the following α-values: $\alpha_{a_0} = \alpha_{a_1} = \alpha_{b_2} = 0$, $\alpha_{a_2} = -1$, and $\alpha_{b_1} = 1$. Note that $\alpha_{a_0} + \alpha_{a_1} + \alpha_{a_2} + \alpha_{b_1} + \alpha_{b_2} = 0$.

Instead of fixing a particular fractional matching $\vec{x}$ and regarding LP1 as a linear program in the n variables α_u, for $u \in A \cup B$, we could regard LP1 as a linear program in $m + n$ variables x_e, for $e \in \tilde{E}$, and α_u, for $u \in A \cup B$. This yields the following linear program (where $\mathsf{vote}_a(b, \vec{x})$ and $\mathsf{vote}_b(a, \vec{x})$ have been explicitly written in terms of x_e's).

$$\textbf{(LP2)} \quad \text{minimize} \sum_{u \in A \cup B} \alpha_u$$

$$\begin{aligned} \alpha_a + \alpha_b \geq \\ \sum_{b' : b' \prec_a b} x_{(a,b')} - \sum_{b' : b' \succ_a b} x_{(a,b')} + \sum_{a' : a' \prec_b a} x_{(a',b)} \\ - \sum_{a' : a' \succ_b a} x_{(a',b)} \quad \forall (a, b) \in \tilde{E}. \\ \sum_{e \in \tilde{E}(u)} x_e \; = 1 \quad \forall u \in A \cup B. \\ x_e \geq 0 \quad \forall e \in \tilde{E}. \end{aligned} \tag{6.1}$$

The polytope $\mathcal{P}'_G$ is the set of optimal solutions to LP2. Hence $\sum_{u \in A \cup B} \alpha_u = 0$ and thus the description of $\mathcal{P}'_G$ consists of the following constraints: (we will refer to constraint (6.1) as the *covering constraint* in $\mathcal{P}'_G$ for (a, b))

$$\begin{aligned} \alpha_a + \alpha_b \geq vote_a(b, \vec{x}) + \\ \mathsf{vote}_b(a, \vec{x}), \forall (a, b) \in \tilde{E}. \end{aligned} \tag{6.2}$$

$$\sum_{u \in A \cup B} \alpha_u = 0. \tag{6.3}$$

$$\sum_{e \in \tilde{E}(u)} x_e = 1, \forall u \in A \cup B, x_e \geq 0, \forall e \in \tilde{E}. \tag{6.4}$$

Observe that the description of $\mathcal{P}'_G$ involves just $O(m+n)$ constraints, far fewer than the exponentially many constraints in the description of $\mathcal{P}_G$. We now show a very interesting and special property of LP2.

Lemma 6.2.1. *LP2 is its own dual program, i.e., LP2 is* self-dual.

Proof. Consider the dual program corresponding to LP2. The dual variables are non-negative y_e for each $e = (a, b) \in \tilde{E}$ and β_u for each $u \in A \cup B$. This linear program is:

$$\textbf{(LP3)} \quad \text{maximize} \sum_{u \in A \cup B} \beta_u$$

$$\begin{aligned}
\beta_a + \beta_b \quad + \quad & \sum_{b' : b' \prec_a b} y_{(a,b')} - \sum_{b' : b' \succ_a b} y_{(a,b')} + \sum_{a' : a' \prec_b a} y_{(a',b)} \\
& - \sum_{a' : a' \succ_b a} y_{(a',b)} \;\leq\; 0 \quad \forall\, (a,b) \in \tilde{E}. \\
\sum_{e \in \tilde{E}(u)} y_e \quad &= \quad 1 \quad \forall u \in A \cup B. \\
y_e \quad &\geq \quad 0 \quad \forall e \in \tilde{E}.
\end{aligned}$$

Let us substitute $\gamma_u = -\beta_u$ for each $u \in A \cup B$. This makes LP3 exactly the same as LP2 – the variable γ_u is in place of α_u for each $u \in A \cup B$ and the variable y_e is in place of x_e for each $e \in \tilde{E}$. □

Thus LP2 is self-dual and so an optimal solution to LP2 is also an optimal solution to LP3. This property will be crucial to us. We now show the relationship between $\mathcal{P}'_G$ and the stable matching polytope $\mathcal{S}_G$.

The typical formulation of $\mathcal{S}_G$ contains the constraints that $\sum_{e \in \tilde{E}(u)} x_e = 1$ for all $u \in A \cup B$ and $x_e \geq 0$ for all $e \in \tilde{E}$ along with the *stability* constraint for each edge $(a, b) \in \tilde{E}$. The stability constraint for edge (a, b) in the description of $\mathcal{S}_G$ from [150] is given by Inequality (6.5) below and the stability constraint for each edge (a, b) in the description of $\mathcal{S}_G$ from [165] is given by Inequality (6.6) below.

$$\sum_{b' :\, b' \succ_a b} x_{(a,b')} \;+\; x_{(a,b)} \;+\; \sum_{a' :\, a' \succ_b a} x_{(a',b)} \;\geq\; 1. \tag{6.5}$$

$$\sum_{b' :\, b' \prec_a b} x_{(a,b')} \;+\; x_{(a,b)} \;+\; \sum_{a' :\, a' \prec_b a} x_{(a',b)} \;\leq\; 1. \tag{6.6}$$

In fact, the above two stability constraints are not independent and one can be derived from

the other.[2] By subtracting the first from the second, we get the following constraint which is equivalent to either of these constraints:

$$\left(\sum_{b' \colon b' \prec_a b} x_{(a,b')} - \sum_{b' \colon b' \succ_a b} x_{(a,b')}\right) + \left(\sum_{a' \colon a' \prec_b a} x_{(a',b)} - \sum_{a' \colon a' \succ_b a} x_{(a',b)}\right) \leq 0. \tag{6.7}$$

Thus constraint (6.7) for each edge (a, b) along with the constraints that $\sum_{e \in \tilde{E}(u)} x_e = 1$ for all $u \in A \cup B$ and $x_e \geq 0$ for all $e \in \tilde{E}$ is a description of $\mathcal{S}_G$. Observe that the first term in constraint (6.7) is $\mathsf{vote}_a(b, \vec{x})$ and the second term there is $\mathsf{vote}_b(a, \vec{x})$. Thus the description of $\mathcal{P}'_G$ is a natural generalization of the description of $\mathcal{S}_G$ where we have $\alpha_a + \alpha_b$ on the right side of constraint (6.7) along with the constraint that the sum of all α_u's has to be 0.

So $\mathcal{S}_G$ is the set of those points $\vec{x}$ in $\mathcal{P}_G$ that admit $\vec{\alpha} = 0$ as a witness to their popularity. While $\mathcal{S}_G$ is integral, we know that $\mathcal{P}_G$ is not integral (as shown by the example in Fig. 6.1). However we will be able to show in Section 6.3 that $\mathcal{P}_G$ is integral in an important special case.

6.3 Integrality of $\mathcal{P}_G$ in a special case

We will prove the following theorem in this section.

Theorem 6.3.1. *Let $G = (A \cup B, E)$ be an instance of the stable marriage problem with strict preference lists such that G admits a perfect stable matching. Then $\mathcal{P}_G$ is integral.*

Our assumption that G admits a perfect stable matching implies that every stable matching S in G is *perfect* [67], i.e., every $u \in A \cup B$ is matched in S. In fact, this implies that every popular matching in G has to be perfect – this is due to the fact that a stable matching is a minimum-size popular matching in G (see Corollary 1 in [89]). Also, this extends to all popular fractional matchings as well. That is, if $\vec{x} \in \mathcal{P}_G$, then $\vec{x}$ has to fully match every vertex in $A \cup B$ to genuine neighbors, otherwise we have $\Delta(\vec{x}, S) < 0$ where S is any stable matching[3], contradicting the popularity of $\vec{x}$. The following lemma shows an important property satisfied by popular perfect matchings in G.

Lemma 6.3.2. *If M is a popular matching in $G = (A \cup B, E)$ such that M is perfect, then M has a witness $\vec{\alpha}^M \in \{\pm 1\}^n$ to its popularity.*

Proof. A perfect popular matching in G is a *dominant* matching in G, i.e., a popular matching M with the property that M is strictly more popular than every larger matching. It was shown in [29] that every dominant matching allows a partition $A_0 \cup A_1$ of A and $B_0 \cup B_1$ of B such that the following two properties are satisfied:

[2] We can obtain (6.6) by summing up the two equations $\sum_{e \in E(a)} x_e = 1$ and $\sum_{e \in E(b)} x_e = 1$ and subtracting (6.5). (6.5) can be obtained from (6.6) analogously.

[3] $\vec{x} \in \mathcal{M}_G$, so $\vec{x}$ is a convex combination of some matchings $M_1, \ldots, M_r$ in G and if some M_j is *not* perfect, then it means $\Delta(M_j, S) < 0$ and we also have $\Delta(M_i, S) \leq 0$ for all matchings M_i by the popularity of S, so this implies $\Delta(\vec{x}, S) < 0$

(1) every blocking edge with respect to M is present in $A_0 \times B_1$

(2) if (a, b) is an edge in $A_1 \times B_0$ then both a and b prefer their respective partners in M to each other.

Set $\alpha_u^M = 1$ for each $u \in A_0 \cup B_1$ and set $\alpha_u^M = -1$ for each $u \in A_1 \cup B_0$. This vector $\vec{\alpha^M}$ witnesses M's membership in $\mathcal{P}_G$ since the covering constraints of all edges get satisfied by properties (1) and (2) given above; also $\sum_{u \in A \cup B} \alpha_u^M = \sum_{(a,b) \in M} (\alpha_a^M + \alpha_b^M)$ since M is perfect and this sum is 0 by our assignment of α^M-values.

□

Let $\vec{x} \in \mathcal{P}_G$. We seek to express $\vec{x}$ as a convex combination of some popular (integral) matchings $M_1, \ldots, M_k$ in G. For ease of the narrative, we say these matchings M_i *span* $\vec{x}$ in the following discussion.

We know from LP1 that there exists a witness $\vec{\alpha^x}$ to the popularity of $\vec{x}$. Since $\vec{x}$ has to fully match every vertex in $A \cup B$ to genuine neighbors, the covering constraint for $(u, \ell(u))$ in the description of $\mathcal{P}'_G$ becomes $\alpha_u^x \geq -1$. (Recall that $\ell(u)$ is the artificial last resort neighbor of u.) So $\alpha_u^x \geq -1$ for each $u \in A \cup B$. It can also be shown that $\alpha_u^x \leq 1$ for each $u \in A \cup B$. Before we prove this, we need the following very useful lemma. Recall that $\tilde{E} = E \cup \{(u, \ell(u)) : u \in A \cup B\}$.

Lemma 6.3.3. *For every $(a, b) \in \tilde{E}$, if $x_{(a,b)} > 0$ then the covering constraint in $\mathcal{P}'_G$ for (a, b) is tight. That is, we have:*

$$\alpha_a^x + \alpha_b^x = \sum_{b' : b' \prec_a b} x_{(a,b')} - \sum_{b' : b' \succ_a b} x_{(a,b')} + \sum_{a' : a' \prec_b a} x_{(a',b)} - \sum_{a' : a' \succ_b a} x_{(a',b)}.$$

Proof. This follows directly from Lemma 6.2.1 which proved that LP2 is self-dual. So $(\vec{x}, \vec{\alpha^x})$ which is an optimal solution to LP2 is also an optimal solution to its dual. Thus the following condition is implied by complementary slackness: if $x_{(a,b)} > 0$ then the constraint in LP2 for $x_{(a,b)}$ is tight. That is,

$$\alpha_a^x + \alpha_b^x = \sum_{b' : b' \prec_a b} x_{(a,b')} - \sum_{b' : b' \succ_a b} x_{(a,b')} + \sum_{a' : a' \prec_b a} x_{(a',b)} - \sum_{a' : a' \succ_b a} x_{(a',b)}.$$

□

Lemma 6.3.4. *For every vertex $u \in A \cup B$, we have $\alpha_u^x \leq 1$.*

Proof. Let $\{v_1, v_2, \ldots, v_d\}$ be the set of all neighbors of u such that $x_{(u,v_i)} > 0$. Let v_d be the least preferred neighbor of u in this set. If $x_{(u,v_d)} = \delta$, then $\sum_{u' : u' \prec_{v_d} u} x_{(u',v_d)} + \sum_{u' : u' \succ_{v_d} u} x_{(u',v_d)} = 1 - \delta$. So $\sum_{u' : u' \prec_{v_d} u} x_{(u',v_d)} - \sum_{u' : u' \succ_{v_d} u} x_{(u',v_d)}$ is at most $1 - \delta$. It follows from the definition of v_d that $\mathsf{vote}_u(v_d, x) = -(1 - \delta)$. Lemma 6.3.3 tells us that the covering constraint for edge (u, v_d) is tight. So $\alpha_u^x + \alpha_{v_d}^x \leq -(1 - \delta) + (1 - \delta) = 0$. Since $\alpha_{v_d}^x \geq -1$, it follows that $\alpha_u^x \leq 1$. □

Thus there exists a witness $\vec{\alpha}^x = (\alpha_u^x)_{u \in A \cup B} \in [-1,1]^n$ for the popularity of $\vec{x}$. We will use $\vec{\alpha}^x$ as follows.

- for each $a \in A$: determine the value p_a such that $p_a \cdot 1 + (1-p_a)\cdot(-1) = \alpha_a^x$, i.e., $2p_a - 1 = \alpha_a^x$.
- for each $b \in B$: determine the value p_b such that $p_b \cdot (-1) + (1-p_b)\cdot 1 = \alpha_b^x$, i.e., $1 - 2p_b = \alpha_b^x$.

The values p_a and p_b. The interpretation of p_a is as follows: we would like to come up with popular matchings spanning x. We know from Lemma 6.3.2 that every popular matching M in G has a witness vector $\vec{\alpha}^M \in \{\pm 1\}^n$ and so for any $a \in A$, we have $\alpha_a^M \in \{\pm 1\}$. Suppose p_a fraction of the popular matchings that span $\vec{x}$ assign a's α-value to 1 and so $1 - p_a$ of them assign a's α-value to -1. Then $p_a \cdot 1 + (1-p_a)\cdot(-1) = 2p_a - 1 = \alpha_a^x$.

Let X_a denote the array containing a's assignment in $\vec{x}$: each cell in the array X_a corresponds to a neighbor b of a such that $x_{(a,b)} > 0$. These neighbors of a are arranged in X_a in *increasing* order of a's preferences and the cell containing b has length $x_{(a,b)}$. Thus the total length of X_a is 1. We use the value p_a to partition X_a into a *positive* sub-array and a *negative* sub-array as defined below.

Definition 6.3.5. *For any $a \in A$, the initial p_a fraction of X_a (i.e., the least preferred p_a fraction of X_a) will be called the* positive *sub-array of X_a and the remaining part (i.e., the most preferred $1 - p_a$ fraction of X_a) will be called the* negative *sub-array of X_a.*

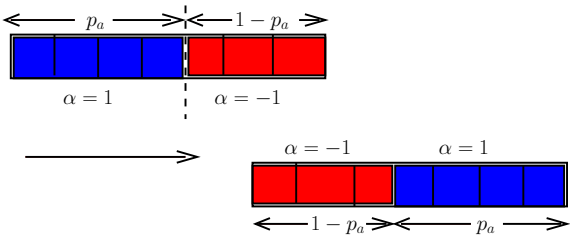

Figure 6.2: We will reorder X_a (the array on the left) by swapping the positive and negative sub-arrays as shown above.

In Fig. 6.2, the array on the left is X_a and the positive sub-array of X_a is colored blue and the negative sub-array of X_a is colored red. We will assume that the positive sub-array of X_a has a's α-value set to 1 and the negative sub-array of X_a has a's α-value set to -1.

Our main idea here is the following: *reorder* X_a as shown in Fig. 6.2. That is, we cut X_a at the end of its positive sub-array and move its entire negative sub-array (in the same order) to the left of its positive sub-array. Note that neither the order within the positive sub-array nor within the negative sub-array is changed by this. The reordered array is shown on the right on Fig. 6.2, call this array X'_a. In case the line cutting X_a into these 2 sub-arrays went through a cell, that cell is now split into 2 cells (one negative and one positive). Thus each cell in X'_a is either positive or negative. Recall that each negative cell corresponds to a's α-value being -1 and each positive cell corresponds to a's α-value being 1.

Similar to the interpretation of p_a, the interpretation of p_b is that if we assume p_b fraction of the popular matchings that span $\vec{x}$ assign b's α-value to -1 and so $(1-p_b)$ of them assign b's α-value to 1, then $p_b \cdot (-1) + (1-p_b) \cdot 1 = 1 - 2p_b = \alpha_b$. As done for each $a \in A$, here also we form the array X_b which is b's assignment in $\vec{x}$, however here the neighbors of b are arranged in *decreasing* order of b's preference.

Definition 6.3.6. *For any $b \in B$, the initial p_b fraction of X_b (i.e., the most preferred p_b fraction of X_b) will be called the* negative *sub-array of X_b and the remaining part (i.e., the least preferred $1-p_b$ fraction of X_b) will be called the* positive *sub-array of X_b.*

Refer to Fig. 6.3 – in the array on the left, the red part is the negative sub-array of X_b and the blue part is the positive sub-array of X_b. As before, we will assume that the negative sub-array of X_b has b's α-value set to -1 and the positive sub-array of X_b has b's α-value set to 1. We will cut X_b at the end of its negative sub-array and move its entire positive sub-array to the left of its negative sub-array as shown in Fig. 6.3. Call this reordered array X'_b.

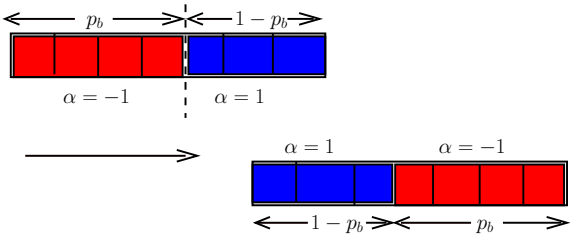

Figure 6.3: The array on the left is X_b (b's neighbors in $\vec{x}$ in decreasing order of b's preference) and the array on the right is X'_b.

Finding the popular matchings that span $\vec{x}$. Form the table T whose rows are the reordered arrays X'_u, for $u \in A \cup B$. The table T has width 1 and the number of cells in u's row is at most $\deg(u)+1$, where $\deg(u)$ is u's degree in G. For any $t \in [0,1)$, define the matching M_t as follows:

- draw the vertical line L_t at distance t from the left end of T;
- the line L_t intersects (or touches the left boundary of) some cell in X'_u for each $u \in A \cup B$;

$$M_t = \{(u,v) : u \in A \cup B \text{ and } v \text{ is } u\text{'s neighbor in this cell}\}.$$

In fact, it is not obvious from the above construction that M_t is a matching: we need to show that if b is in the cell in X'_a at distance t from the left end of T, then a has to be in the cell in X'_b at distance t from the left. We will show this in Theorem 6.3.8. Before we show this, we show the following simple lemma which will be used in the proof of Theorem 6.3.8.

Lemma 6.3.7. *For any $(a,b) \in \tilde{E}$, we have $x_{(a,b)} + \sum_{b' \prec_a b} x_{(a,b')} + \sum_{a' \prec_b a} x_{(a',b)} \leq p_a + (1-p_b)$. Moreoever, if $x_{(a,b)} > 0$ then this inequality is tight.*

Proof. We know by the covering constraint in $\mathcal{P}'_G$ that for any $(a,b) \in \tilde{E}$:

$$\begin{aligned}\alpha_a^x + \alpha_b^x \quad \geq \quad & \sum_{b' \prec_a b} x_{(a,b')} - \sum_{b' \succ_a b} x_{(a,b')} + \\ & \sum_{a' \prec_b a} x_{(a',b)} - \sum_{a' \succ_b a} x_{(a',b)}\end{aligned}$$

Rewrite the above constraint in a simpler form by substituting $\sum_{b' \succ_a b} x_{(a,b')} = 1 - x_{(a,b)} - \sum_{b' \prec_a b} x_{(a,b')}$ and similarly substitute $\sum_{a' \succ_b a} x_{(a',b)} = 1 - x_{(a,b)} - \sum_{a' \prec_b a} x_{(a',b)}$. Replace α_a^x with $2p_a - 1$ and α_b^x with $1 - 2p_b$. This results in the following simpler looking inequality:

$$\begin{aligned} & x_{(a,b)} + \sum_{b' \prec_a b} x_{(a,b')} + \sum_{a' \prec_b a} x_{(a',b)} \\ \leq \; & 1 + p_a - p_b \;=\; p_a + (1 - p_b). \end{aligned} \tag{6.8}$$

Note that it follows from Lemma 6.3.3 that when $x_{(a,b)} > 0$, Inequality (6.8) is tight. Thus we have shown the lemma. □

We are now ready to show that M_t is a valid matching in G. For any edge (a,b), if $x_{(a,b)} > 0$, we need to show that the cell corresponding to b in X'_a and the cell corresponding to a in X'_b are perfectly *aligned* in the vertical direction.

Theorem 6.3.8. *M_t is a matching in G.*

Proof. Recall that in X'_a, a's increasing order of preference of partners in $\vec{x}$ begins from the start of its positive sub-array (the blue region) in a left to right orientation and it wraps around. Suppose $\sum_{b' \prec_a b} x_{(a,b')} \geq p_a$. Let $d = \sum_{b' : b' \prec_a b} x_{(a,b')} - p_a$. Then after traversing length d from the start of X'_a (refer to Fig. 6.4), we reach the cell in X'_a that contains b – this is the darkened red cell in X'_a in Fig. 6.4 and it has length $x_{(a,b)}$.

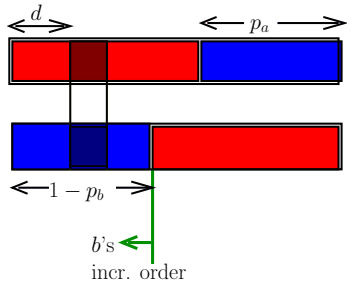

Figure 6.4: The top array is X'_a and the darkened cell there contains b. The cell exactly below this in the blue sub-array of X'_b contains a.

Similarly in X'_b, b's increasing order of preference of partners in $\vec{x}$ begins from the end of its positive sub-array and this order is from right to left (as indicated by the arrow in Fig. 6.4). Let $d' = \sum_{a' : a' \prec_b a} x_{(a',b)}$. After traversing length d' from this vertical line in X'_b (marking the end of the positive sub-array) from right to left, we reach the cell that contains a. Note that this cell is within

the positive sub-array of X'_b since $d'+x_{(a,b)} \leq 1-p_b$ because $\sum_{b' \prec_a b} x_{(a,b')}+d'+x_{(a,b)} = p_a+(1-p_b)$ (by Lemma 6.3.7) and we are in the case where $\sum_{b' \prec_a b} x_{(a,b')} \geq p_a$.

Refer to Fig. 6.4, where the cell in X'_b that contains a is the darkened blue cell and it has length $x_{(a,b)}$. Since $d + x_{(a,b)} + d' = 1 - p_b$, it follows that the cell containing a in X'_b and the cell containing b in X'_a are exactly aligned with each other in the vertical direction.

The picture is absolutely symmetric when $\sum_{a' \prec_b a} x_{(a',b)} \geq (1 - p_b)$. Then the cell containing b is in the positive sub-array of X'_a and the cell containing a is in the negative sub-array of X'_b.

The only case left is when $\sum_{b' \prec_a b} x_{(a,b')} < p_a$ *and* $\sum_{a' \prec_b a} x_{(a',b)} < (1 - p_b)$. So in X_a, the line separating the positive sub-array from the negative sub-array went through the cell containing b and similarly, in X_b, the line separating the negative sub-array from the positive sub-array went through the cell containing a. Let $d_0 = \sum_{b' \prec_a b} x_{(a,b')}$ and $d_1 = \sum_{a' \prec_b a} x_{(a',b)}$ (see Fig. 6.5).

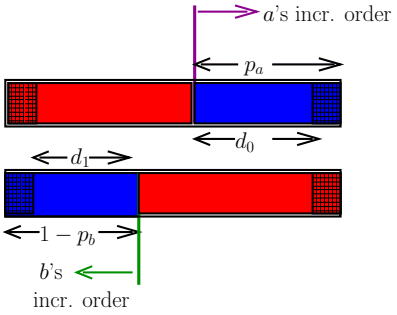

Figure 6.5: Both the leftmost (the dashed red) cell and the rightmost (the dashed blue) cell in X'_a contain b. Symmetrically both the leftmost (the dashed blue) cell and the rightmost (the dashed red) cell in X'_b contain a. Let the length of the leftmost cell in X'_a be $x^0_{(a,b)}$ and let the length of the rightmost cell in X'_b be $x^1_{(a,b)}$.

Let the length of the rightmost cell in X'_a be $x^0_{(a,b)}$ and let the length of the leftmost cell in X'_b be $x^1_{(a,b)}$. So $x^0_{(a,b)} = p_a - d_0$ and $x^1_{(a,b)} = (1 - p_b) - d_1$. Lemma 6.3.7 tells us that $x_{(a,b)} + d_0 + d_1 = p_a + (1 - p_b)$. Hence $x_{(a,b)} = x^0_{(a,b)} + x^1_{(a,b)}$. We know that the length of the cell containing b in X_a is $x_{(a,b)}$. So is the length of the cell containing a in X_b. Hence the length of the leftmost cell in X'_a is $x_{(a,b)} - x^0_{(a,b)} = x^1_{(a,b)}$ and the length of the rightmost cell in X'_b is $x_{(a,b)} - x^1_{(a,b)} = x^0_{(a,b)}$. Thus in this case as well we have perfect alignment between the two cells in X'_a that contain b and the two cells in X'_b that contain a. So no matter which of these cells is intersected by the line L_t, we have exact alignment between the cell containing a in X'_b and the cell containing b in X'_a. □

The popularity of matching M_t. We now need to show that M_t is a popular matching in G. We do this by showing a witness vector $\vec{\alpha}^t$ such that $\vec{\alpha} = \vec{\alpha}^t$ and $\vec{x} = \mathcal{I}_{M_t}$ satisfy the constraints of $\mathcal{P}'_G$, where $\mathcal{I}_{M_t}$ is the 0-1 edge incidence vector of M_t. We define $\vec{\alpha}^t$ as follows. Recall that we defined the matching M_t via the vertical line L_t that intersected table T.

- For each $u \in A \cup B$ do: if the cell intersected by L_t is in the *negative* sub-array of X'_u, then set $\alpha^t_u = -1$; else (the cell is in the *positive* sub-array of X'_u) set $\alpha^t_u = 1$.

Observe that $\alpha_a^t + \alpha_b^t = 0$ for each edge $(a, b) \in M_t$. This is because it follows from the proof of Theorem 6.3.8 that, for each edge $(a, b) \in M_t$, either b's cell is in the negative sub-array of X'_a and a's cell is in the positive sub-array of X'_b or vice-versa. Since M_t is a perfect matching, Corollary 6.3.9 follows.

Corollary 6.3.9. $\sum_{u \in A \cup B} \alpha_u^t = 0$.

We now need to show that $\vec{\alpha}^t$ and $\mathcal{I}_{M_t}$ also satisfy the "covering constraints" in $\mathcal{P}_G$. Refer to Fig. 6.6. The vertical line in X'_a (with the left to right arrow adjacent to it) denotes the start of a's preference order of its partners in $\vec{x}$ in increasing order and this wraps around. Similarly, the vertical line in X'_b (with the right to left arrow adjacent to it) denotes the start of b's preference order of its partners in $\vec{x}$ in increasing order and this also wraps around.

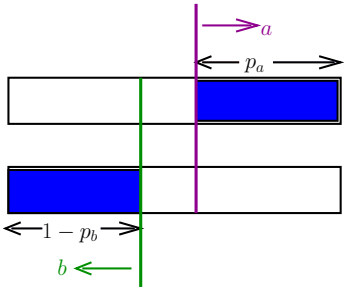

Figure 6.6: The array on top is X'_a and the rightwards arrow there denotes a's increasing order. The array on bottom in X'_b and the leftwards arrow there denotes b's increasing order (this is from right to left).

Let the line L_t intersect cell c in X'_a and cell c' in X'_b. Using Lemma 6.3.7, it is easy to make the following observations:

(I) suppose $x_{(a,b)} = 0$: so $\sum_{b' \prec_a b} x_{(a,b')} + \sum_{a' \prec_b a} x_{(a',b)} \leq p_a + (1 - p_b)$. Note that $p_a + (1 - p_b)$ is the sum of lengths of positive sub-arrays in X'_a and X'_b. Hence for both a and b to be matched in M_t to worse partners than each other, both c and c' must be in their respective *positive* sub-arrays. So if only one of c, c' is in its positive sub-array, then at least one of a, b is matched in M_t to a better partner than the other.

(II) suppose both c and c' are in their respective negative sub-arrays: then we claim that *both* a and b get matched in M_t to better partners than each other. This is because $x_{(a,b)} + \sum_{b' \prec_a b} x_{(a,b')} + \sum_{a' \prec_b a} x_{(a',b)} \leq p_a + (1 - p_b)$ and $p_a + (1 - p_b)$ is the sum of the lengths of the positive sub-arrays in X'_a and in X'_b.

Lemma 6.3.10. *For each $(a, b) \in \tilde{E}$, we have $\alpha_a^t + \alpha_b^t \geq \mathsf{vote}_a(b, M_t(a)) + \mathsf{vote}_b(a, M_t(b))$.*

Proof. Let $(a, b) \in E$ and let the line L_t intersect cell c in X'_a and cell c' in X'_b. There are three possible cases regarding the "signs" of the cells c and c':

(1) Both c and c' are positive: so $\alpha_a^t = \alpha_b^t = 1$ and thus $\alpha_a^t + \alpha_b^t = 2$; hence the covering constraint for edge (a, b) holds because the right side of this constraint is always at most 2.

(2) One of c, c' is positive and the other is negative: so (α_a^t, α_b^t) is either $(-1, 1)$ or $(1, -1)$. We know from Observation (I) and the proof of Theorem 6.3.8 that when exactly one of the cells is positive, either (i) at least one of a, b is matched in M_t to a better partner or (ii) a and b are matched to each other. Thus the edge (a, b) is covered in both these sub-cases.

(3) Both cells c and c' are negative: so $\alpha_a^t = \alpha_b^t = -1$. We know from Observation (II) that when both c and c' are negative, then both a and b are matched in M_t to better partners than each other. Thus here also the edge (a, b) is covered.

Thus $\vec{\alpha}^t$ and $\mathcal{I}_{M_t}$ together satisfy the covering constraints in $\mathcal{P}_G$ for all edges (a, b) in E. For each $u \in A \cup B$, we have $\alpha_u^t \geq -1$ and so the covering constraint for the edge $(u, \ell(u))$ is also satisfied (because $\mathsf{vote}_u(\ell(u), M_t) = -1$). We have also shown that $\sum_u \alpha_u^t = 0$. Thus we can conclude that $\mathcal{I}_{M_t} \in \mathcal{P}_G$, i.e., M_t is a popular matching in G.

□

We are now ready to express $\vec{x}$ as a convex combination of popular matchings: these matchings are obtained by sweeping a vertical line from the left end to the right end of table T. Whenever a new cell begins in some row in T (say, at distance t from the left end of T), we define a new matching M_t as described above. The leftmost cell in T begins at distance 0 from the left end of T, let the second leftmost cell in T begin at distance t_1 from the left side of T, and so on, i.e., let the i-th leftmost cell in T begin at distance t_{i-1} from the left side of T. Then we have:

$$\vec{x} = t_1 \cdot M_0 + (t_2 - t_1) \cdot M_{t_1} + \cdots + (1 - t_{k-1}) \cdot M_{t_{k-1}}.$$

The total number of matchings k that we construct here is at most $m + |A|$ since $m + |A| = \sum_{a \in A} (\deg(a) + 1)$ is an upper bound on the total number of distinct cells in T. Thus every popular fractional matching in G can be expressed as a convex combination of at most $m + n$ popular matchings in G. This finishes the proof of Theorem 6.3.1, in other words, if $G = (A \cup B, E)$ admits a perfect stable matching then the polytope $\mathcal{P}_G$ is integral.

6.4 Half-integrality of $\mathcal{P}_G$ in the general bipartite graph

In this section we are in the general case: we have an instance $G = (A \cup B, E)$ with strict preference lists and G need not admit a stable matching that matches all vertices. We know that $\mathcal{P}_G$ need not be integral here. We will show the following theorem.

Theorem 6.4.1. *The popular fractional matching polytope $\mathcal{P}_G$ in $G = (A \cup B, E)$ is half-integral.*

We will show the above theorem with the help of Theorem 6.3.1. Using the given instance G, we will construct a new instance $H = (V \cup V', E')$ as follows: let $V = A_0 \cup B_1$ and let $V' = B_0 \cup A_1$, where $A_i = \{a_i : a \in A\}$ and $B_i = \{b_i : b \in B\}$, for $i = 0, 1$ (see Fig. 6.7).

The edge set of H is $E' = E_0 \cup E_1 \cup \{(u_0, u_1) : u \in A \cup B\}$, where $E_i = \{(a_i, b_i) : (a, b) \in E\}$, for $i = 0, 1$. For $i = 0, 1$ and for each vertex u_i in H, u_i's preference list is the same as it was in G, with a subscript i added to each of its neighbors (in the same order of preference) along with u_{1-i} added as u_i's least preferred neighbor in H.

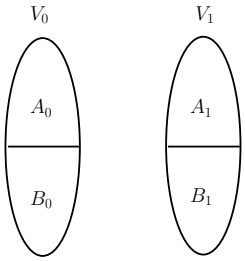

Figure 6.7: The vertex set of the graph H is two copies of vertex set of the graph G.

Lemma 6.4.2. *H admits a perfect stable matching.*

Proof. Let S be a stable matching in H. Let $S_0 = S \cap E_0$ and $S_1 = S \cap E_1$. Thus both S_0 and S_1 are stable matchings in G. Since all stable matchings in G match exactly the same vertices [67], it follows that u_0 is left unmatched in S_0 if and only if u_1 is left unmatched in S_1. Thus $S = S_0 \cup S_1 \cup \{(u_0, u_1) : u \text{ is an unstable vertex in } G\}$. Thus S is perfect. □

We can now use Theorem 6.3.1 to conclude that $\mathcal{P}_H$ (the popular fractional matching polytope of H) is integral. The rest of this section will use the integrality of $\mathcal{P}_H$ to prove that $\mathcal{P}_G$ is half-integral. In order to do this, we define a mapping f from $\mathcal{P}_G$ to the set of fractional matchings in H. Let $\vec{x} \in \mathcal{P}_G$, where $\vec{x} = (x_e)_{e \in \tilde{E}}$; we know that there exists a witness $\vec{\alpha}^x = (\alpha^x_v)_{v \in A \cup B}$ such that $\vec{x}$ and $\vec{\alpha}^x$ satisfy the constraints of $\mathcal{P}'_G$.

Define the vector $f(\vec{x}) = \vec{z} = (z_e)_{e \in E'}$ as follows: for every edge $(a, b) \in E$, let $z_{(a_0,b_0)} = z_{(a_1,b_1)} = x_{(a,b)}$ and for every $u \in A \cup B$, let $z_{(u_0,u_1)} = x_{(u,\ell(u))}$. It is easy to see that $\vec{z}$ is a fractional matching in H.

Lemma 6.4.3. *$f(\vec{x})$ is a popular fractional matching in H.*

Proof. We need to show a witness vector $\vec{\beta} = (\beta_v)_{v \in V \cup V'}$ such that $\vec{\beta}$ and $f(\vec{x})$ satisfy the constraints in $\mathcal{P}'_H$. We define $\vec{\beta}$ as follows: for each $u \in A \cup B$, let $\beta_{u_0} = \beta_{u_1} = \alpha^x_u$. Let $(a, b) \in E$. Given the fact that $\vec{x}$ and $\vec{\alpha}^x$ satisfy the covering constraints in $\mathcal{P}_G$ for (a, b), it immediately follows that $f(\vec{x})$ along with the vector $\vec{\beta}$ satisfies covering constraints in $\mathcal{P}'_H$ for the edges (a_0, b_0) and (b_1, a_1).

Consider the edge $(u, \ell(u))$ in G for any $u \in A \cup B$: we have $\alpha^x_u \geq x_{(u,\ell(u))} - 1$. In the graph H, the right side of the edge covering constraint for the edge (u_0, u_1) is $2(x_{(u,\ell(u))} - 1)$ and as the left hand side is $\alpha^x_u + \alpha^x_u$, it follows that the covering constraint for the edge (u_0, u_1) is also satisfied by $f(\vec{x})$ and $\vec{\beta}$. Thus the covering constraints in the description of $\mathcal{P}'_H$ for all edges in E' are satisfied by $f(\vec{x})$ and $\vec{\beta}$. Since $\beta_{u_i} = \alpha^x_u \geq -1$, the covering constraint in $\mathcal{P}'_H$ for the edge $(u_i, \ell(u_i))$ is trivially satisfied for all $u \in A \cup B$ and $i = 0, 1$.

□

Our goal now is to define a mapping h, which is the inverse of f, from $\mathcal{P}_H$ to $\mathcal{P}_G$, i.e., $h \circ f(\vec{x}) = \vec{x}$ for any popular fractional matching $\vec{x}$ in G. Let $\vec{z} = (z_e)_{e \in E'}$ be any popular fractional matching in H. We define $h(\vec{z}) = \vec{y} = (y_e)_{e \in \tilde{E}}$ as given below and note that $h \circ f(\vec{x}) = \vec{x}$ for any $x \in \mathcal{P}_G$.

$$y_{(a,b)} \quad = \quad (z_{(a_0,b_0)} + z_{(b_1,a_1)})/2 \quad \text{for every } (a, b) \in E$$

$$y_{(u,\ell(u))} \quad = \quad z_{(u_0,u_1)} \quad \text{for every } u \in A \cup B.$$

For $\vec{y}$ to belong to $\mathcal{P}_G$, it is necessary that $\vec{y}$ satisfies $\sum_{e \in \tilde{E}(u)} y_e = 1$ for each $u \in A \cup B$. Since $\sum_{e \in E'} z_e = 1$, it is simple to see that the above constraint is satisfied. We will now show that $\vec{y}$ satisfies the other constraints defining $\mathcal{P'}_G$.

For this we need to show a witness vector $\vec{\alpha}$: we know that there exists a witness vector $(\beta_v)_{v \in V \cup V'}$ that is a witness to $\vec{z}$'s popularity in H. For each $u \in A \cup B$, let $\alpha_u = (\beta_{u_0} + \beta_{u_1})/2$. Since $\sum_{u \in A \cup B} (\beta_{u_0} + \beta_{u_1}) = 0$, we have $\sum_{u \in A \cup B} \alpha_u = 0$. Using the fact that $\vec{\beta}$ and $\vec{z}$ satisfy covering constraints for all edges in E', it is straightforward to show that $\vec{\alpha}$ and $h(\vec{z})$ satisfy covering constraints for all edges in $\tilde{E}$. Thus we can conclude the following lemma.

Lemma 6.4.4. *For any popular fractional matching $\vec{z}$ in H, $h(\vec{z})$ is a popular fractional matching in G.*

Theorem 6.4.1 follows from Lemma 6.4.5. Thus the polytope $\mathcal{P}_G$ is half-integral.

Lemma 6.4.5. *Let $\vec{x} \in \mathcal{P}_G$. Then $\vec{x} = \sum_{i=1}^{r} \lambda_i p_i$, where $p_1, \ldots, p_r$ are popular half-integral matchings in G and $\lambda_i \geq 0$ for $1 \leq i \leq r$ along with $\sum_i \lambda_i = 1$.*

Proof. Let $\vec{x} \in \mathcal{P}_G$ and let $\vec{z} = f(\vec{x})$. We know from Lemma 6.4.3 that $\vec{z} \in \mathcal{P}_H$. Since $\mathcal{P}_H$ is integral, it follows that $\vec{z} = \sum_{i=1}^{r} \lambda_i M_i$, where $M_1, \ldots, M_r$ are popular matchings in H and for each i, we have $\lambda_i \geq 0$ along with $\sum_i \lambda_i = 1$.

We know that $h(\vec{z}) = \vec{x}$. So $\vec{x} = h(\vec{z}) = \sum_{i=1}^{r} \lambda_i \cdot h(M_i)$. Lemma 6.4.4 tells us that $h(M_1), \ldots, h(M_r)$ belong to $\mathcal{P}_G$. Since $M_1, \ldots, M_r$ are integral matchings in H, it follows from the definition of h that $h(M_1), \ldots, h(M_r)$ are half-integral matchings in G. Thus $\vec{x}$ is a convex combination of popular half-integral matchings in G. □

6.5 Half-integrality of $\mathcal{P}_G$ in a roommates instance

The input instance here is a graph $G = (V, E)$ (not necessarily bipartite) and popular matchings need not always exist here. Consider the instance G on the left of Fig. 6.8 on 3 vertices a, b, and c. Suppose the preferences are cyclic, i.e., a prefers b to c, b prefers c to a, and c prefers a to b. It is easy to see that G has no popular matching.

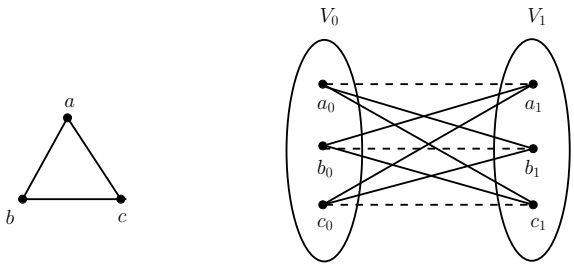

Figure 6.8: To the left is the graph G which is the triangle on a, b, and c. To the right is the graph H which is essentially the 6-cycle a-b'-c-a'-b-c'-a along with the new edges $(a, a'), (b, b'), (c, c')$ which are the dashed edges above.

We first define the fractional matching polytope FM_G of G:

$$\mathsf{FM}_G = \{\vec{x} \in \mathbb{R}^m : \sum_{e \in E(u)} x_e \leq 1 \ \forall u \in V \ \text{ and } \ x_e \geq 0 \ \forall e \in E\}.$$

The polytope $\mathcal{P}_G$ is the set of points $\vec{x} \in \mathsf{FM}_G$ such that $\Delta(\vec{x}, \vec{y}) \geq 0$ for all $\vec{y} \in \mathsf{FM}_G$. It is known that FM_G is *half-integral*, where a half-integral matching $\vec{p} = (p_e)_{e \in E}$ is a point in FM_G such that $p_e \in \{0, \frac{1}{2}, 1\}$ for each edge e. Thus $\mathcal{P}_G$ can also be defined as $\Delta(\vec{x}, \vec{p}) \geq 0$ for all half-integral matchings p in G.

As done in the case of bipartite instances in Section 6.2, there is an extended formulation $\mathcal{P}'_G$ on $m + n$ variables as given by constraints (6.2)-(6.4). We will use this extended formulation to prove the half-integrality of $\mathcal{P}_G$ as done in Section 6.4.

Corresponding to G, we define the bipartite graph $H = (V \cup V', E')$ where V is the vertex set of G and $V' = \{v' : v \in V\}$. Thus V' is another copy of V. The edge set $E = \{(u, v') : (u, v) \in E\} \cup \{(v, v') : v \in V\}$. We show an example in Fig. 6.8 where the graph H on the right is the bipartite graph corresponding to the graph G on the left. The preference lists of vertices are identical to how we defined them in Section 6.4. Thus for each $v \in V$, we have v' as v's last choice and symmetrically, v is the v''s last choice. The following lemma will be useful to us.

Lemma 6.5.1. *H admits a perfect stable matching.*

Proof. Let S be a stable matching in H. We claim that S must be perfect. Suppose, for a contradiction, that v inV is unmatched (the case $v' \in V'$ is unmatched follows symmetrically). As S is stable, v' must be matched to some vertex u_0 which ranks higher than v. Then u'_0 is matched to some vertex u_1 ranking higher than v (otherwise (v, u'_0) blocks S). As u'_0 prefers u_1 to v, u_0 prefers u'_1 to v'. Thus, u'_1 must be matched to some vertex u_2 ranking higher than u_0, otherwise (u_0, u'_1) blocks S. Repeating this reasoning, we find a path $\rho = \langle v, v', u_0, u'_0, u_1, u'_1, \ldots \rangle$, where $s(v') = u_0$ and $s(u'_i) = u_{i+1}$ for $i \geq 0$. Such a path is of finite length and must loop back to v, i.e., there exists some node u'_k so that $s(u'_k) = v$, contradicting the assumption that v is unmatched. □

Thus we can conclude that the popular fractional matching polytope $\mathcal{P}_H$ of H is integral (by Theorem 6.3.1) The rest of the proof is identical to the proof in Section 6.4. For any $\vec{x} \in \mathcal{P}_G$, define the function $f(\vec{x}) = \vec{z} = (z_e)_{e \in E'}$ where $z_{(u,v')} = z_{(v,u')} = x_{(u,v)}$ for every edge $(u, v) \in E$ and $z_{(v,v')} = x_{(v,\ell(v))}$ for any $v \in A \cup B$. As done in the proof of Lemma 6.4.3, it is easy to show that $f(\vec{x})$ is a popular fractional matching in H. Thus f is a function from $\mathcal{P}_G$ to $\mathcal{P}_H$.

We then define a mapping h from $\mathcal{P}_H$ to $\mathcal{P}_G$ such that $h \circ f(\vec{x}) = \vec{x}$ for any popular fractional matching $\vec{x}$ in G. Let $\vec{z} = (z_e)_{e \in E'}$ be any popular fractional matching in H. We define $h(\vec{z}) = \vec{y} = (y_e)_{e \in \tilde{E}}$ as follows: $y_{(u,v)} = (z_{(u,v')} + z_{(v,u')})/2$ for every $(u, v) \in E$ and $y_{(v,\ell(v))} = z_{(v,v')}$ for every $v \in V$. It is again easy to show that $\vec{y} \in \mathcal{P}_G$. So for any popular fractional matching $\vec{z}$ in H, $h(\vec{z})$ is a popular fractional matching in G.

As done in the proof of Lemma 6.4.5, we can now show that if $\vec{x} \in \mathcal{P}_G$ then $\vec{x} = \sum_{i=1}^{r} \lambda_i p_i$, where $p_1, \ldots, p_r$ are popular half-integral matchings in G and $\lambda_i \geq 0$ for $1 \leq i \leq r$ along with $\sum_i \lambda_i = 1$. Thus we can conclude Theorem 6.1.6.

6.6 Hardness of max-utility popular matching in roommates instances

In this section, we prove Theorem 6.1.4 by showing that it is NP-hard to compute a max-utility popular matching in a roommate instance. Here we use the characterization of popular matchings from [89] that uses *edge labels* – for edge (u, v) in $E \setminus M$, assign the label $(\mathsf{vote}_u(v, M(u)), \mathsf{vote}_v(u, M(v))$. Note that all edges with the label (1,1) are *blocking edges* to M. Let G_M be the graph obtained by deleting all edges labeled $(-1, -1)$ from G. The matching M is popular in G if and only if the following three conditions hold in G_M:

(1) There is no alternating cycle with respect to M that contains a $(1, 1)$ edge.

(2) There is no alternating path starting from an unmatched vertex that contains a $(1, 1)$ edge.

(3) There is no alternating path with respect to M that contains more than one $(1, 1)$ edge.

Let $G = (V, E)$ be an instance of VERTEX COVER. We construct a new graph H now. H will be a complete graph. For every vertex $v_i \in V$, there will be four vertices $a_0^i, a_1^i, a_2^i, a_3^i$ in H and they have the following preferences (here $\pi(X)$ means a permutation of elements in X in an arbitrary order and $\pi(\cdots)$ means an arbitrary permutation of all vertices not explicitly listed so far in the preference list of a_t^i, $0 \leq t \leq 3$):

$$
\begin{aligned}
a_0^i &: a_1^i \succ \pi(\cup_{\forall v_j, (v_i, v_j) \in E} a_j^0) \succ a_i^2 \succ a_i^3 \succ \pi(\cdots) \\
a_1^i &: a_0^i \succ a_2^i \succ a_3^i \succ \pi(\cdots) \\
a_2^i &: a_1^i \succ a_0^i \succ a_3^i \succ \pi(\cdots) \\
a_3^i &: a_1^i \succ a_2^i \succ a_0^i \succ \pi(\cdots)
\end{aligned}
$$

We now define edge utilities in H as follows: $w(a_0^i, a_2^i) = w(a_1^i, a_3^i) = 2$, $\forall v_i \in V$; all other edges e have $w(e) = 1$.

Lemma 6.6.1. *Let $C \subseteq V$ be a vertex cover of G. Then there exists a popular matching M in H with $w(M) = 4n - 2k$, where $k = |C|$ and $n = |V|$.*

Proof. We construct a popular matching M in H according to the vertex cover C in G. If $v_i \in C$, then $\{(a_0^i, a_1^i), (a_2^i, a_3^i)\} \subseteq M$; if $v_i \in V \setminus C$, then $\{(a_0^i, a_2^i), (a_1^i, a_3^i)\} \subseteq M$. Observe that G_M consists of the following edges. See Fig. 6.9 and Fig. 6.10.

1. Edges in M.
2. $\forall v_i \in C$: $(a_0^i, a_2^i), (a_1^i, a_2^i), (a_1^i, a_3^i)$ – these are $(1, -1)$ edges.
3. $\forall v_i \in V \setminus C$: $(a_0^i, a_1^i), (a_1^i, a_2^i)$ – these are $(1, 1)$ edges.
4. (a_0^i, a_0^j) is a $(1, -1)$ edge and such an edge exists in G_M if and only if $(v_i, v_j) \in E$, and exactly one of v_i, v_j is in C.

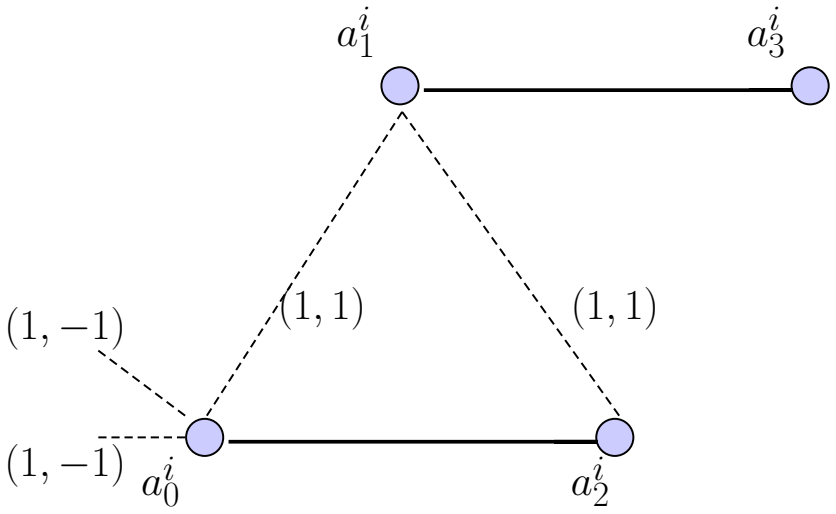

Figure 6.9: The vertices corresponding to $v_i \notin C$ in G_M: note that only a_0^i has edges connecting it to other vertices a_0^k with $v_k \in C$.

Figure 6.10: The vertices corresponding to $v_j \in C$ in G_M: note that only a_0^j has edges connecting it to other vertices a_0^k with $v_k \notin C$.

Note that if $(v_i, v_j) \in E$, then (a_0^i, a_0^j) cannot be a $(1,1)$ edge since C is a vertex cover.

We now show that M is popular. As M is a perfect matching, we only need to show that conditions (1) and (3) characterizing popular matchings (from [89]) stated earlier hold in G_M. That is, there is no alternating path containing two or more $(1,1)$ edges and no alternating cycle containing at least a $(1,1)$ edge.

The only $(1,1)$ edges in G_M are (a_0^i, a_1^i) and (a_1^i, a_2^i) when $v_i \in V \backslash C$. The alternating path (cycle) containing (a_0^i, a_1^i) must contain $(a_3^i, a_1^i) \in M$ and $(a_0^i, a_2^i) \in M$ and neither a_3^i nor a_2^i has any other incident edges in G_M. So we can rule out such paths (cycles).

Next suppose that the alternating path (cycle) contains (a_1^i, a_2^i). Such a path (cycle) contains $(a_1^i, a_3^i) \in M$ and $(a_0^i, a_2^i) \in M$ and as a_3^i has no other incident edge in G_M, we can rule out the possibility of the alternating cycle containing (a_1^i, a_2^i). Observe that a_0^i may have incident edges (a_0^i, a_0^j) labeled as $(1,-1)$ in G_M, when $v_j \in C$. However, the alternating path, if it includes (a_0^i, a_0^j), must also include $(a_0^j, a_1^j) \in M$. As a_1^j has only (a_1^j, a_2^j) and (a_1^j, a_3^j) as incident edges, it is easy to see that this alternating path must be "trapped" (see Figure 6.10) inside the vertices corresponding to v_j (i.e., this path cannot continue on to vertices corresponding to other nodes $v_{k \neq j}$), without encountering a second $(1,1)$ edge. This proves the popularity of M. □

We now proceed on the other direction of the reduction.

Lemma 6.6.2. *No edge $(a_t^i, a_{t'}^j)$, $0 \leq t, t' \leq 3$, $i \neq j$, belongs to a popular matching M.*

Proof. First assume that $t, t' \in \{0, 1\}$. If $(a_t^i, a_{t'}^j) \in M$, then the path v_{1-t}^i-v_t^i-$v_{t'}^j$-$v_{1-t'}^j$ contains two $(1, 1)$ edges. Next assume that $t \in \{0, 1\}$ and $t' \in \{2, 3\}$. Suppose first that $t' = 2$. If $(a_1^j, a_3^j) \notin M$, then v_{1-t}^i-v_t^i-v_2^j-v_3^j has two $(1, 1)$ edges; if $(a_1^j, a_3^j) \in M$, then v_{1-t}^i-v_t^i-v_2^j-v_0^j has two $(1, 1)$ edges ((v_2^j, v_0^j) is a $(1, 1)$ edge since we already established that v_0^j cannot be matched to $v_0^{j' \neq j}$ in M). Suppose next that $t' = 3$. If a_2^j is matched to neither a_0^j nor a_1^j, then v_{1-t}^i-v_t^i-v_3^j-v_2^j has two $(1, 1)$ edges; if a_2^j is matched to $a_{t''}^j$ where $t'' \in \{0, 1\}$, then v_{1-t}^i-v_t^i-v_3^j-$v_{1-t''}^j$ has two $(1, 1)$ edges.

So we have established that a_0^i, a_1^i cannot be matched to vertices corresponding to other nodes $v_j \in V$. The remaining case is $t, t' \in \{2, 3\}$. Suppose that $t = 2$. If $(a_1^i, a_3^i) \in M$, then (a_2^i, a_1^i) is a $(1, 1)$ edge; if $(a_1^i, a_0^i) \in M$, then (a_2^i, a_3^i) is $(1, 1)$. Suppose that $t = 3$. If (a_0^i, a_1^i) (respectively $(a_0^i, a_2^i), (a_1^i, a_2^i)$) is part of M, then (a_3^i, a_2^i) (respectively, $(a_3^i, a_1^i), (a_3^i, a_0^i)$) is an $(1, 1)$ edge. This applies to t' analogously. We then have a path of length 3 with two $(1, 1)$ edges (with $(a_t^i, a_{t'}^j)$ in between) to arrive at a contradiction. □

Lemma 6.6.3. *Suppose M is popular in the instance H.*

(i) For the vertices $a_0^i, a_1^i, a_2^i, a_3^i$, either $\{(a_0^i, a_1^i), (a_2^i, a_3^i)\} \subseteq M$ or $\{(a_0^i, a_2^i), (a_1^i, a_3^i)\} \subseteq M$.

(ii) If $(v_i, v_j) \in E$, then it cannot happen that $\{(a_0^i, a_2^i), (a_1^i, a_3^i), (a_0^j, a_2^j), (a_1^j, a_3^j)\} \subseteq M$.

Proof. For (i), first note that M has to be a perfect matching, otherwise, the edge between two unmatched vertices is a $(1, 1)$ edge, contradicting the popularity of M. So by Lemma 6.6.2, we just need to rule out the possibility of $\{(a_0^i, a_3^i), (a_1^i a_2^i)\} \subseteq M$. If this happens, then a_0^i-a_1^i-a_2^i-a_3^i-a_0^i is a cycle with a $(1, 1)$ and a $(1, -1)$ edge in it, contradicting the popularity of M.

For (ii), if the described situation really happens, then the alternating path a_0^i-a_0^j-a_2^j-a_1^j has two $(1, 1)$ edges, a contradiction.

□

Lemma 6.6.4. *Suppose M is a popular matching with $w(M) = 4n - 2k$. Then we can construct a vertex cover C with $|C| = k$.*

Proof. By Lemma 6.6.3 (i), either $\{(a_0^i, a_1^i), (a_2^i, a_3^i)\} \subseteq M$ or $\{(a_0^i, a_2^i), (a_1^i, a_3^i)\} \subseteq M$. In the former case, put $v_i \in C$; in the latter case, leave v_i ouside of C. It follows from Lemma 6.6.3 (ii) that the set C is a vertex cover in G. The fact that $|C| = k$ can be easily verified.

□

Using Lemmas 6.6.1 and 6.6.4, we can draw the following conclusion.

Theorem 6.6.5. *It is NP-hard to compute a max-utility popular matching in the roommates instance, even when all preferences are strict and complete, and each edge utility is either 1 or 2.*

Suppose the edge utilities are non-negative arbitrary values. We can also show the following inapproximability result.

Theorem 6.6.6. *Suppose all utilities are non-negative. Then there is no polynomial time ϵ-approximation algorithm, for any $\epsilon > 0$, to compute a max-utility popular matching in a roommates instance, even when all preferences are strict and complete, unless the unique games conjecture fails.*

Proof. In the above reduction, let us modify the edge utilities as follows: $w(a_0^i, a_2^i) = w(a_1^i, a_3^i) = 1$, $\forall v_i \in V$; all other edges e have $w(e) = 0$. Lemmas 6.6.1 and 6.6.4 can be rephrased as there is a vertex cover of size k if and only if there is a popular matching of utility $2n - 2k$.

According to [23], VERTEX COVER is hardest to approximate when the given instance $G = (V, E)$ has a perfect matching. Thus we can assume that $|\mathsf{OPT_{VC}}| \geq n/2$.

First suppose $|\mathsf{OPT_{VC}}| \geq (1/2 + \epsilon)n$. Then returning the entire set of vertices V would give a $\frac{n}{(1/2+\epsilon)n} = 2 - \Theta(\epsilon)$ approximation. Next suppose $n/2 \leq |\mathsf{OPT_{VC}}| \leq (1/2+\epsilon)n$. Then the maximum utility of a popular matching in H is at least $2n - 2((1/2 + \epsilon)n) = (1 - 2\epsilon)n$.

Suppose we have an ϵ-approximation algorithm for some $\epsilon > 0$. Then the utility of the returned matching M is at least $\epsilon(1 - 2\epsilon)n$. As $w(M) = 2n - 2k \geq \epsilon(1 - 2\epsilon)n$, we have $k \leq (1 - \epsilon/2 + \epsilon^2)n$. By assumption $|\mathsf{OPT_{VC}}| \geq n/2$, so we have a vertex cover whose size is at most $\frac{(1-\epsilon/2+\epsilon^2)n}{n/2} = 2 - \epsilon + \epsilon^2 = 2 - \Theta(\epsilon)$ times the size of $\mathsf{OPT_{VC}}$. In both cases, we have a $2 - \Theta(\epsilon)$ approximation for VERTEX COVER. This would break the unique games conjecture. □

Acknowledgement.

We are grateful to the anonymous reviewers for their helpful comments. The second author wishes to thank Naveen Garg for fruitful discussions.

Bibliography

[1]

[2] H. G. Abeledo and U. G. Rothblum. Stable matchings and linear inequalities. *Discrete Applied Mathematics*, 54(1-27), 1994.

[3] D. Abraham, R. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.

[4] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Pearson, 1993.

[5] R. Ahuja, J. Orlin, C. Stein, and R. Tarjan. Improved algorithms for bipartite network flow. *SIAM J. Comput.*, 23(5):906–933, 1994.

[6] M. Aigner and T. A. Dowling. Matching theory for combinatorial geometries. *Transactions of the American Mathematical Society*, 158(1):231–245, 1971.

[7] N. Alon, I. Gamzu, and M. Tennenholtz. Optimizing budget allocation among channels and influencers. In *Proceedings of the 21st International Conference on World Wide Web (WWW)*, pages 381–388, 2012.

[8] J. Aráoz, W. Cunningham, J. Edmonds, and J. Green-Krótki. Reductions to 1-matching polyhedra. *Networks*, 13:455–473, 1983.

[9] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: massive data summarization on the fly. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 671–680, 2014.

[10] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1497–1514, 2013.

[11] M. L. Balinski. Establishing the matchig polytope. *Journal of Combinatorial Theory B*, pages 1–13, 1972.

[12] C. Berge. Sur le couplage maximum d'un graphe. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 247(258-359), 1958.

[13] A. Bogomolnaia and H. Moulin. A new solution to the random assignment problem. *Journal of Economic theory*, 100:295–328, 2001.

[14] C. Brezovec, G. Cornuéjols, and F. Glover. Two algorithms for weighted matroid intersection. *Mathematical Programming*, 36(1):39–53, 1986.

[15] G. Calinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[16] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

[17] T.-H. H. Chan, Z. Huang, S. H.-C. Jiang, N. Kang, and Z. G. Tang. Online submodular maximization with free disposal: Randomization beats for partition matroids online. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1204–1223, 2017.

[18] C. Chekuri, S. Gupta, and K. Quanrud. Streaming algorithms for submodular function maximization. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 9134, pages 318–330, 2015.

[19] C. Chekuri and K. Quanrud. Fast approximations for matroid intersection. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2016.

[20] C. Chekuri, J. Vondrák, and R. Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.

[21] H. Y. Cheung, T. C. Kwok, and L. C. Lau. Fast matrix rank algorithms and applications. *Journal of the ACM*, 60(5):31, 2013.

[22] H. Y. Cheung, L. C. Lau, and K. M. Leung. Algebraic algorithms for linear matroid parity problems. *ACM Transacitons on Algorithms*, 10(3):10, 2014.

[23] M. Chlebík and J. Chlebíková. Minimum 2sat-deletion: Inapproximability results and relations to minimum vertex cover. *Discrete Applied Mathematics*, 155(2):172–179, 2007.

[24] P. Christiano, J. A. Kelner, A. Madry, D. A. Spielman, and S. Teng. Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC*, pages 273–282, 2011.

[25] N. Christofides. Worst case analysis of a new heuristic for the travelling salesman problem. Tech. rep., Graduate School of Industrial Administration, Carnegie Mellon University, 1976.

[26] W. J. Cook, W. Cunningham, W. R. Pulleyblank, and A. Schrijver. *Combinatorial Optimization*. Wiley-Blackwell, 1997.

[27] D. Coppersmith and S. Winograd. Matrix multiplication via arithmetic progressions. *Journal of Symbolic Computation*, 9(3):251–280, 1990.

[28] A. Cseh, C.-C. Huang, and T. Kavitha. Popular matchings with two-sided preferences and one-sided ties. *SIAM Journal on Discrete Mathematics*, 31(4):2348–2377, 2017.

[29] A. Cseh and T. Kavitha. Popular edges and dominant matchings. *Mathematical Programming, Series B*, 172(1-2):209–229, 2018.

[30] W. Cunningham and A. Marsh. A primal algorithm for optimum matching. *Polyhdedral Combinatorics—Dedicated to the Memory of D.R. Fulkerson*, pages 50–72, 1978.

[31] W. H. Cunningham. Improved bounds for matroid partition and intersection algorithms. *SIAM Journal on Computing*, 15(4):948–957, 1986.

[32] M. Cygan, H. Gabow, and P. Sankowski. Algorithmic applications of baur-strassen's theorem: shortest cycles, diameter and matchings. In *Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 2012.

[33] U. Derigs. A shortest augmenting path method for solving minimal perfect matching problems. *Networks*, 11:379–390, 1981.

[34] R. Dougherty, C. Freiling, and K. Zeger. Network coding and matroid theory. *Proceedings of the IEEE*, 99(3):388–405, 2011.

[35] R. Duan and S. Pettie. Linear-time approximation for maximum weight matching. *Journal of the ACM*, 61(1):1, 2014.

[36] R. Duan and S. Pettie. Linear-time approximation for maximum weight matching. *J. of ACM*, 61(1):1–23, 2014.

[37] R. Duan, S. Pettie, and H.-H. Su. Scaling algorithms for weighted matching in general graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 781–800, 2017.

[38] R. Duan, S. Pettie, and H.-H. Su. Scaling algorithms for weighted matching in general graphs. *Journal ACM Transactions on Algorithms*, 14(1), 2018.

[39] R. Duan and H.-H. Su. A scaling algorithm for maximum weight matchings in bipartite graphs. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1413–1424, 2012.

[40] A. Dulmage and N. Mendelsohn. Coverings of bipartite graphs. *Canadian J. of Mathematics*, 10:517–534, 1958.

[41] J. Edmonds. Maximum matching and a polyhedron with $(0, 1)$ vertices. *Journal of Research National Bureau of Standards Section*, 69(B):125–130, 1965.

[42] J. Edmonds. Paths, trees, and flowers. *Canadian J. of Mathematics*, 17(49-467), 1965.

[43] J. Edmonds. Submodular functions, matroids, and certain polyhedra. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, pages 69–87. Gordon and Breach, 1970.

[44] J. Edmonds. Matroids and the greedy algorithm. *Mathematical Programming*, 1(1):127–136, 1971.

[45] J. Edmonds. Matroid intersection. *Annals of Discrete Mathematics*, 4:39–49, 1979.

[46] F. B. F. Brandl and H. G. Seedig. Consistent probabilistic social choice. *Econometrica*, 84(5), 2016.

[47] T. Feder. A new fixed point approach for stable networks and stable marriages. *Journal of Computer and System Sciences*, 45(2):233–284, 1992.

[48] T. Feder. Network flow and 2-satisfiability. *Algorithmica*, 11(3), 1994.

[49] U. Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

[50] Y. Filmus and J. Ward. A tight combinatorial algorithm for submodular maximization subject to a matroid constraint. *SIAM Journal on Computing*, 43(2):514–542, 2014.

[51] P. Fishburn. Probabilistic social choice based on simple voting comparisons. *Review of Economic Studies*, 51(167):683–692, 1984.

[52] M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions ii. *Mathematical Programming Study*, 8:73–87, 1978.

[53] T. Fleiner. A fixed-point approach to stable matchings and some applications. *Mathematics of Operations Research*, 28(1):103–126, 2003.

[54] A. Frank. A weighted matroid intersection algorithm. *Journal of Algorithms*, 2(4):328–336, 1981.

[55] A. Frank. A quick proof for the matroid intersection weight-splitting theorem. Technical Report QP-2008-03, Egerváry Research Group on Combinatorial Optimization, 2008.

[56] M. Fredman and R. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. of ACM*, 34(3):596–615, 1987.

[57] S. Fujishige. *Submodular functions and optimization*. Elsevier, 2nd edition, 2005.

[58] S. Fujishige and X. Zhang. An efficient cost scaling algorithm for the independent assignment problem. *Journal of the Operations Research Society of Japan*, 38(1):124–136, 1995.

[59] H. Gabow. An efficient implementation of edmonds' algorithm for maximum matching on graphs. *J. of ACM*, 23(221-234), 1976.

[60] H. Gabow. A scaling algorithm for weighted matching on general graphs. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 90–100, 1985.

[61] H. Gabow. Data structures for weighted matching and nearest common ancestors with linking. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 434–443, 1990.

[62] H. Gabow and P. Sankowski. Algebraic algorithms for b-matching, shortest undirected paths, and f-factors. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 137–146, 2013.

[63] H. Gabow and R. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18:1013–1036, 1989.

[64] H. Gabow and R. Tarjan. Faster scaling algorithms for network problems. *SIAM Journal on Computing*, 18(5):1013–1036, 1989.

[65] H. Gabow and R. Tarjan. Faster scaling algorithms for general graph-matching problems. *J. of ACM*, 38:815–853, 1991.

[66] H. Gabow and R. Tarjan. Faster scaling algorithms for general graph-matching problems. *Journal of the ACM*, 38(4):815–853, 1991.

[67] H. N. Gabow and M. F. M. Stallmann. Efficient algorithms for graphic matroid intersection and parity (extended abstract). In *Proceedings of the 12th Colloquium on Automata, Languages and Programming,, ICALP*, pages 210–220, 1985.

[68] H. N. Gabow and Y. Xu. Efficient algorithms for independent assignments on graphic and linear matroids. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science, FOCS*, pages 106–111, 1989.

[69] H. N. Gabow and Y. Xu. Efficient theoretic and practical algorithms for linear matroid intersection problems. *Journal of Computer and System Sciences*, 53(1):129–147, 1996.

[70] D. Gale and L. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[71] F. L. Gall. Power of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 296–303, 2014.

[72] F. L. Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC*, pages 296–303, 2014.

[73] T. Gallai. Maximale systeme unabhängiger kanten. *A Magyar Tudományos Akadémia—Matematikai Kutató Intézetének Közleményei*, 9:401–413, 1964.

[74] P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Sciences*, 20:166–173, 1975.

[75] A. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM Journal on Computing*, 24(3):494–504, 1995.

[76] A. Goldberg and A. Karzanov. Maximum skew-symmetric flows. In *Proceedings of the 3rd European Symposium on Algorithms, ESA*, pages 155–170, 1995.

[77] A. Goldberg and R. Tarjan. Solving minimum-cost flow problems by successive approximation. In *Proceedings of the 9th Symposium on Theory of Computing Conference, STOC*, pages 7–18, 1987.

[78] A. Goldberg and R. Tarjan. Solving minimum-cost flow problems by successive approximation. *Mathematics of Operations Research*, 15(3):430–466, 1990.

[79] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[80] F. B. H. Aziz and M. Brill. On the tradeoff between economic efficiency and strategyproofness in randomized social choice. In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems, AAMAS*, pages 455–462, 2013.

[81] F. B. H. Aziz and P. Stursberg. On popular random assignments. In *Proceedings of the 6th International Symposium on Algorithmic Game Theory, SAGT*, pages 183–194, 2013.

[82] F. Hadlock. Finding a maximum cut of a planar graph in polynomial time. *SIAM Journal on Computing*, 4(3):221–225, 1975.

[83] N. J. A. Harvey. An algebraic algorithm for weighted linear matroid intersection. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 444–453, 2007.

[84] N. J. A. Harvey. Algebraic algorithms for matching and matroid problems. *SIAM Journal on Computing*, 39(2):679–702, 2009.

[85] A. Hoffman and R. Oppenheim. Local unimodularity in the matching polytope. *Annals of Discrete Mathematics*, 2:201–209, 1978.

[86] C.-C. Huang, N. Kakimura, and N. Kamiyama. Exact and approximation algorithms for weighted matroid intersection. Mi preprint series, Mathematics for Industry, Kyushu University, 2014.

[87] C.-C. Huang and T. Kavitha. Efficient algorithms for maximum weight matchings in general graphs with small edge weights. In *Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 1400–1412, 2012.

[88] C.-C. Huang and T. Kavitha. Weight-maximal matchings. In *Proceedings of the 2nd International Workshop on Matching under Preferences*, 2012.

[89] C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. *Information and Computation*, 220(180-194), 2013.

[90] C.-C. Huang and T. Kavitha. New algorithms for maximum weight matching and a decomposition theorem. *Mathematics of Operations Research*, 42(2):411–426, 2017.

[91] C.-C. Huang, T. Kavitha, K. Mehlhorn, and D. Michail. Fair matchings and related problems. *Algorithmica*, 74(2):1184–1203, 2016.

[92] M. Iri. A new method of solving transportation-network problems. *Journal of the Operations Research Society of Japan*, (3):27–87, 1960.

[93] M. Iri. Applications of matroid theory. In *Mathematical Programming—the state of the art*, pages 158–201, 1983.

[94] M. Iri and N. Tomizawa. An algorithm for finding an optimal "independent assignment". *Journal of the Operations Research Society of Japan*, 19(1):32–57, 1976.

[95] R. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch. Rank-maximal matchings. *ACM Transactions on Algorithms*, 2(4):602–610, 2006.

[96] R. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *J. of ACM*, 34(3):532–543, 1987.

[97] R. W. Irving. An efficient algorithm for the stable roommates problem. *Journal of Algorithms*, 6:577–595, 1985.

[98] R. W. Irving. Greedy matchings. University of Glasgow, Computing Science Department Research Report, TR-2003-136, 2003.

[99] R. W. Irving, T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch. Rank-maximal matchings. *ACM Transactions on Algorithms*, 2(4):602–610, 2006.

[100] T. A. Jenkyns. The efficacy of the "greedy" algorithm. *Proceedings of the 7th Southeastern International Conference on Combinatorics, Graph Theory, and Computing*, pages 341–350, 1976.

[101] P. M. Jensen and B. Korte. Complexity of matroid property algorithms. *SIAM Journal on Computing*, 11(1):184–190, 1982.

[102] M.-Y. Kao, T. W. Lam, W.-K. Sung, and H.-F. Ting. A decomposition theorem for maximum weight bipartite matchings. *SIAM Journal on Computing*, 31(1):18–26, 2001.

[103] T. Kavitha. A size-popularity tradeoff in the stable marriage problem. *SIAM Journal on Computing*, 43(1):52–71, 2014.

[104] T. Kavitha. Popular half-integral matchings. In *Proceedings of the 43rd Colloquium on Automata, Languages and Programming,, ICALP*, pages 22:1–22:13, 2016.

[105] T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412:2679–2690, 2011.

[106] T. Kavitha and C. D. Shah. Efficient algorithms for weighted rank-maximal matchings and related problems. In *Procceding of 17th International Symposium on Algorithms and Computation, ISAAC*, pages 153–162, 2006.

[107] T. Kavitha and C. D. Shah. Efficient algorithms for weighted rank-maximal matchings and related problems. In *Proceedings of the 17th International Symposium on Algorithms and Computation, ISAAC*, volume 4288 of *Lecture Notes in Computer Science*, pages 153–162, 2006.

[108] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 217–226, 2014.

[109] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 137–146, 2003.

[110] D. Knuth. *Mariages stables et leurs relations avec d'autre problèmes.* Les Presses de l'université de Montréal, 1976.

[111] B. Korte and D. Hausmann. An analysis of the greedy heuristic for independence systems. In P. H. B. Alspach and D. Miller, editors, *Algorithmic Aspects of Combinatorics*, volume 2 of *Annals of Discrete Mathematics*, pages 65–74. Elsevier, 1978.

[112] A. Krause, A. P. Singh, and C. Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9:235–284, 2008.

[113] G. Kreweras. Aggregation of preference orderings. *Mathematics and Social Science I: Proceedings of the seminar of Menthon-Saint-Bernad, France and of Gösing, Austria*, (73-79), 1966.

[114] M.-K. Kuan. Graphic programming using odd or even points. *Chinese Mathematics*, 1(273-277), 1962.

[115] H. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–9, 1955.

[116] A. Kulik, H. Shachnai, and T. Tamir. Maximizing submodular set functions subject to multiple linear constraints. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 545–554, 2013.

[117] E. L. Lawler. Optimal matroid intersections. In R. Guy, H. Hanani, N. Sauer, and J. Schönheim, editors, *Combinatorial Structures and Their Applications*, pages 233–234. Gordon and Breach, 1970.

[118] E. L. Lawler. Matroid intersection algorithms. *Mathematical Programming*, 9(1):31–56, 1975.

[119] E. L. Lawler. *Combinatorial Optimization: Networks and Matroids*. Dover Publications, 1976.

[120] J. Lee. *Maximum Entropy Sampling*, volume 3 of *Encyclopedia of Environmetrics*, pages 1229–1234. John Wiley & Sons, Ltd., 2006.

[121] J. Lee, M. Sviridenko, and J. Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

[122] Y. Lee and A. Sidford. Path finding ii : An $\tilde{O}(m\sqrt(n))$ algorithm for the minimum cost flow problem. ArXiv: 1312.6713v2, 2015.

[123] Y. Lee, A. Sidford, and S. C.-W. Wong. A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proceedings of the 56th Annual Symposium on Foundations of Computer Science, FOCS*, 2015.

[124] Y. T. Lee, S. Rao, and N. Srivastava. A new approach to computing maximum flows using electrical flows. In *Proceedings of the 45th Symposium on Theory of Computing Conference, STOC*, pages 755–764, 2013.

[125] H. Lin and J. Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 912–920, 2010.

[126] H. Lin and J. Bilmes. A class of submodular functions for document summarization. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT)*, pages 510–520, 2011.

[127] L. Lovász and M. Plummer. *Matching theory*. American Mathematical Society, 2009.

[128] D. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific Publishing Company Incorporated, 2013.

[129] S. T. McCormick. *Handbook on Discrete Optimization*, chapter 7, pages 321–391. Elsevier, 2006.

[130] K. Mehlhorn and D. Michail. Network problems with non-polynomial weights and applications. Available at www.mpi-sb.mpg.de/~mehlhorn/ftp/HugeWeights.ps.

[131] D. Michail. Reducing rank-maximal to maximum weight matching. *Theoretical Computer Science*, 389(1-2):125–132, 2007.

[132] M. Mucha and P. Sankowski. Maximum matchings via gaussian elimination. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 248–255, 2004.

[133] K. Murota. *Matrices and matroids for systems analysis*. Springer, 2nd edition, 2000.

[134] K. Murota, M. Iri, and M. Nakamura. Combinatorial canonical form of layered mixed matrices and its applications to block-triangularization of systems of linear/nonlinear equations. *SIAM Journal on Algebraic and Discrete Methods*, 8:123–149, 1987.

[135] J. Orlin. A faster strongly polynomial minimum cost flow algorithm. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 377–387, 1988.

[136] J. Orlin. Parallel algorithms for the assignment and minimum-cost flow problems. *Operations Research*, 42(2):338–350, 1993.

[137] J. Orlin. Max flows in $o(nm)$ time, or better. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing, STOC*, pages 765–774, 2013.

[138] J. B. Orlin and R. K. Ahuja. New scaling algorithms for the assignment and minimum mean cycle problems. *Mathematical Programming*, 54(1):41–5, 1992.

[139] R. I. P. Biró and D. Manlove. Popular matchings in the marriage and roommates problems. In *Proceedings of the 7th International Conference on Algorithms and Complexity, CIAC*, pages 97–108, 2010.

[140] I. M. P. Eirinakis, D. Magos and P. Miliotis. Polyhedral aspects of stable marriage. *Mathematics of Operation Research*, 39(3):656–671, 2014.

[141] K. E. Paluch. Capacitated rank-maximal matchings. In *Proceedings of the 8th International Conference on Algorithms and Complexity, CIAC*, volume 7878 of *Lecture Notes in Computer Science*, pages 324–335, 2013.

[142] G. Pap. A matroid intersection algorithm. Technical Report TR-2008-10, Egerváry Research Group on Combinatorial Optimization, 2008.

[143] J. Petersen. Die theorie der regulären graphs. *Acta Mathematica*, 15:193–220, 1891.

[144] S. Pettie. A simple reduction from maximum weight matching to maximum cardinality matching. *Information Processing Letters*, 112(23):893–898, 2012.

[145] A. Recski. Terminal solvability and n-port interconnection problem. In *IEEE International Symposium on Circuits and Systems*, pages 988–991, 1979.

[146] A. Recski. *Matroid theory and its applications in electric network theory and in statics*. Springer, 1989.

[147] A. Roth and M. Sotomayor. *Two-sided matching: a study in game-theoretic modeling and analysis*. Cambridge university press, 1992.

[148] A. E. Roth. New physicians: A natural experiment in market organization. *Science*, 250:1524–1528, 1990.

[149] A. E. Roth. A natural experiment in the organization of entry level labor markets: Regional markets for new physicians and surgeons in the u.k. *American Economic Review*, 81:415–440, 1991.

[150] A. E. Roth, U. G. Rothblum, and J. H. V. Vate. Stable matchings, optimal assignments, and linear programming. *Mathematics of Operations Research*, 18(4):803–828, 1993.

[151] U. Rothblum. Characterization of stable matchings as extreme points of a polytope. *Mathematical Programming*, 54(57-67), 1992.

[152] P. Sankowski. Shortest paths in matrix multiplication time. In *Proceedings of the 13th European Symposium on Algorithms, ESA*, pages 770–778, 2005.

[153] P. Sankowski. Maximum weight bipartite matching in matrix multiplication time. *Theoretical Computer Science*, 410(44):4480–4488, 2009.

[154] P. Sankowski. Maximum weight bipartite matching in matrix multiplication time. *Theoretical Computer Science*, 410:4480–4488, 2009.

[155] A. Schrijver. *Theory of Integer and Linear Programming*. Wiley-Blackwell, 1998.

[156] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[157] A. Schrijver. *Combinatorial Optimization - Polyhedra and Efficiency*. Springer-Verlag Berlin and Heidelberg GmbH, 2003.

[158] A. Schrijver. *Combinatorial optimization: Polyhedra and efficiency*. Springer, 2003.

[159] J. Sherman. Nearly maximum flows in nearly linear time. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science, FOCS*, 2013.

[160] M. Shigeno and S. Iwata. A dual approximation approach to weighted matroid intersection. *Operations Research Letters*, 18(3):153–156, 1995.

[161] C. Sng. *Efficient Algorithms for bipartite matching problems with preferences*. PhD thesis, University of Glasgow, 2008.

[162] T. Soma, N. Kakimura, K. Inaba, and K. Kawarabayashi. Optimal budget allocation: Theoretical guarantee and efficient algorithm. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 351–359, 2014.

[163] A. Storjohann. High-order lifting and integrality certification. *Journal of Symbolic Computation*, 36:613–648, 2003.

[164] M. Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Operations Research Letters*, 32(1):41–43, 2004.

[165] C.-P. Teo and J. Sethuraman. The geometry of fractional stable matchings and its applications. *Mathematics of Operations Research*, 23(4):874–891, 1998.

[166] M. Thorup and U. Zwick. Approximate distance oracles. *Journal of the ACM*, 52(1):1–24, 2005.

[167] W. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 22, 1947.

[168] S. Ueno, Y. Kajitani, and S. Gotoh. On the nonseparating independent set problem and feedback set problem for graphs with no vertex degree exceeding three. *Discrete Mathematics*, 72(1-3):355–360, 1988.

[169] J. H. V. Vate. Linear programming brings marital bliss. *Operations Research Letters*, 8(3):147–153, 1989.

[170] W. R. P. W. J. Cook, W.H. Cunningham and A. Schrijver. *Combinatorial Optimization*. Wiley-Interscience, 1997.

[171] V. V. Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th Annual ACM Symposium on Theory of Computing, STOC*, pages 887–898, 2012.

[172] L. Wolsey. Maximising real-valued submodular functions: primal and dual heuristics for location problems. *Mathematics of Operations Research*, 1982.

[173] Q. Yu, E. L. Xu, and S. Cui. Streaming algorithms for news and scientific literature recommendation: Submodular maximization with a d-knapsack constraint. *IEEE Global Conference on Signal and Information Processing*, 2016.

[174] R. Yuster and U. Zwick. Answering distance queries in directed graphs using fast matrix multiplication. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 90–100, 2005.