City University of New York (CUNY)
CUNY Academic Works

All Dissertations, Theses, and Capstone Projects | Dissertations, Theses, and Capstone Projects

9-2017

Approximation Algorithms for Effective Team Formation

George Rabanca
The Graduate Center, City University of New York

How does access to this work benefit you? Let us know!

More information about this work at: https://academicworks.cuny.edu/gc_etds/2353
Discover additional works at: https://academicworks.cuny.edu

This work is made publicly available by the City University of New York (CUNY).
Contact: AcademicWorks@cuny.edu

Approximation Algorithms for Effective Team Formation

by

George Rabanca

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

2017

© 2017

George Rabanca

All Rights Reserved

This manuscript has been read and accepted by the Graduate Faculty in Computers Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

	Amotz Bar Noy
Date	Chair of Examining Committee
	Robert Haralick
Date	Executive Officer

Supervisory Committee

Matthew P. Johnson

Saad Mneimneh

David Peleg

ABSTRACT

Approximation Algorithms for Effective Team Formation

by

George Rabanca

Advisor: Professor Amotz Bar-Noy

This dissertation investigates the problem of creating multiple disjoint teams of maximum efficacy from a fixed set of workers. We identify three parameters which directly correlate to the team effectiveness – team expertise, team cohesion and team size – and propose efficient algorithms for optimizing each in various settings. We show that under standard assumptions the problems we explore are not optimally solvable in polynomial time, and thus we focus on developing efficient algorithms with guaranteed worst case approximation bounds. First, we investigate maximizing team expertise in a setting where each worker has different expertise for each job and each job may be completed only by teams of certain sizes. Second, we consider the problem of maximizing team cohesion when the set of workers form a social network with known pairwise compatibility. Third, we explore the problem from a game theoretic perspective in which multiple teams compete on a fixed number of workers and the true needs of each team are private. We present allocation algorithms that both incentivize teams to state their needs accurately and allocate workers effectively. Finally, we experimentally measure the correlation between team cohesiveness, team expertise and team efficacy on a social network graph of computer science research co-authorship.

to my parents

Acknowledgements

I would like to first thank my advisor Amotz Bar-Noy for his continuous support since my undergraduate years until the completion of this dissertation. He has been of an unmatched help during the most challenging times of my doctoral studies helping me regain my motivation and confidence whenever I needed it. With his joyful interest in all aspects of computer science he has inspired me in finding new problems to work on and directed me toward fulfilling research areas. Moreover, he has helped me connect with senior researchers by generously supporting multiple research visits, conferences and summer school expenses.

I am especially indebted to Bhaskar Krishnamachari for hosting me multiple times at his research group at USC, and closely working with his student, Yi Gai, and I on my first published paper. His mentorship has been invaluable and my numerous visits to USC were some of the most fun and productive times of my PhD.

I would like to thank all the members of my committee for their time and I am particularly grateful to David Peleg and Matthew P. Johnson for their close collaboration on research included in this thesis. Among the senior researchers I had the pleasure to interact with I would especially like to thank Sergei Artemov and Rohit Parikh for introducing me to different subfields of Game Theory; Simon Parsons and Elizabeth Sklar for supporting me though grants during my first graduate year; Magnús M. Halldórsson for hosting me at the University of Reykjavik. I would also like to thank Prithwish Basu, Ben Baumer, Eden Chlamtác, Michael Dinitz, Yi Gai, Christian Konard, Guy Kortsarz, Ou Liu and Yanting Wu for collaborating with me on interesting problems. I would especially like to thank Ivo Vigan who through all these years has been not only a very productive research partner but also a very dear and reliable friend.

My life during these years would not have been the same without a close group of friends I shared almost every day with and I came to consider family. I don't have the words to

express my gratitude to have been around you for so many years both during good and bad times. Thank you Catherine Hall, Kostantinos Pouliasis, Ivo Vigan (again), Raffael Flores-Contreras and Michelle Christina Larsen for your friendship and all the fun we had together during this time. Above all, I would like to thank Krystal Raydo for supporting me during the most stressful moments of the last few years and my brother, Bogdan, for first inspiring me to start my doctoral studies.

Contents

List of Figures

Chapter 1

Introduction

In our increasingly globalized society governments, companies and other organizations need to assemble teams of growing complexity to tackle problems on a larger scale than ever before. While it is clear that some teams are more effective than others, and numerous studies analyze the mutifaceted qualities that correlate with team performance, bringing together such highly competent groups is still as much of an art as a science. This, we argue, is at least in part due to the computational complexity of the optimization problems intrinsic to finding such solutions, and thus in this dissertation we study efficient approximation algorithms for idealized versions of these problems.

Over the last three decades, stemming from the work on group dynamics of Kurt Lewin [1], researchers in psychology, social sciences and management have produced a large body of literature aimed at understanding the factors determining the performance of groups of people working together. One of the major obstacles to this understanding is having a robust definition of the concept of *team success*. While, for example, a team manager might define success as the timely completion of an individual task, a team member might find improved social relations within the team and personal growth just as important or more [2]. Although in this work we only concern ourselves with the correlation between team performance and

the individual qualities of the team members, it is worth noting that team composition is only one of multiple factors that affect team success. For example, Daniel Levi [3] distinguishes three other major factors that determine team efficacy: the task at hand (how well suited is the task for team work), the group process (how well does the internal structure of the team work) and the organizational context (how much support does the team get from the larger organization).

It is unsurprising that, as Bennis and Biederman show [4], good teams are comprised good team members. However, other factors of the team composition have also been found to correlate with team performance. For example heterogenous groups have been found to be more innovative than homogenous groups [5], more cohesive teams have better performance in certain tasks [6], and without a doubt larger teams outperform teams of smaller size in many tasks [7].

In this dissertation we take a computational approach to understand the complexities of efficient team formation. While many of the qualities of teams discussed above are hard or even impossible to objectively and precisely measure, and moreover, it is not well understood how these characteristics offset and complement each other to determine a team's efficacy, we focus our attention to much simplified versions of the real-world problems. We focus on three axis on which team quality can be measured – expertise, cohesion and size – and show that even in this idealized setting some of the natural optimization problems arising from searching for optimally performing teams on any of these axis are computationally intractable. Therefore, we aim to find efficient algorithms that are guaranteed to output solutions with provable approximation guarantees in polynomial time.

In Chapter 2 we investigate a team formation setting where the goal is to maximize cumulative team *expertise* when each worker has different expertise for each *job* and jobs can only be completed by teams of specific sizes. In graph theoretic terms this setting is captured by the SeminarAssignment Problem [8] which is NP-complete in general.

We provide an efficient algorithm that guarantees a constant approximation ratio while identifying conditions under which the problem becomes computationally tractable.

Next, we study a setting where a set of workers must be partitioned into teams in a way that maximizes team *cohesiveness*. We propose two measures for team cohesiveness based on pairwise collaboration scores for each pair of workers which are captured by well studied graph partition problems. This motivates Chapter 3 where we study novel approximation algorithms for two well known graph partition problems: the 2-PathPartition problem [9–12] and the TrianglePacking problem [9, 13].

In Chapter 4 we investigate two game theoretic models of the problem. This setting focuses on the team *size* while individual workers' expertise is ignored. Given that the true needs of each team are private information to each team's *manager* our goal is to find mechanisms that, under common game theoretic assumptions, guarantee efficient worker allocation even when team leaders intend to manipulate the mechanism to be allocated as many workers as possible.

In Chapter 5 we take an experimental approach to the study of team cohesiveness. We analyze a DBLP dataset with 45,191 authors and investigate how author expertise and previous collaboration among a paper's authors correlate with the success (number of citations) of the paper. We observe that in this dataset not all pairwise collaborations are equally important, and that the two measures for team cohesiveness we propose have similar accuracy estimating a team's strength.

Chapter 2

Team Expertise

2.1 Introduction

In this chapter, based on work presented at WAOA 2016 [14], we consider a setting where a set of workers must be allocated to a set of teams in a way that maximizes the total expertise of the workers to the assigned team. More precisely, let B be a set of teams (or bins) and let I be a set of workers (or items), and for each worker i and team b let $p(i, b) \in \mathbb{R}$ represent the expertise (or profit) of the worker i when assigned to team b. Moreover, assume that each team $b \in B$ must be assigned one of several number of workers, and let the set of integers K_b denote the allowable number of workers that can be assigned to team b. A *team assignment* is a function $\mathcal{A} : I \to B$ and we say that the assignment is feasible if $|\mathcal{A}^{-1}(b)| \in K_b$ for all $b \in B$, where $\mathcal{A}^{-1}$ is the pre-image of $\mathcal{A}$. The goal is to find a feasible team assignment $\mathcal{A}$ that maximizes the total expertise:

$$p(\mathcal{A}) = \sum_{i \in I} p(i, \mathcal{A}(i)).$$

The problem has been introduced in a slightly less general form by Krumke et. al. as

the Seminar Assignment problem (SAP) [8]. In their version for each $b \in B$ the set K_b equals to $\{0\} \cup \{l_b, ..., u_b\}$ for some lower and upper bounds $l_b, u_b \in \mathbb{N}$. The more general setting considered in this work can be useful for example when a team doesn't just require a minimum number of workers and has a fixed maximum size, but in addition requires workers to work in pairs and therefore would allow only teams of even size. Moreover, this generalization also simplifies notation and helps simplify the exposition of our algorithm.

SAP is a variant of the classic General Assignment problem (GAP) in which one is given m bins with capacity $B_1, ..., B_m$ and n items. Each item i has size $s(i, b)$ in bin b and yields profit $p(i, b)$. The goal is to find a packing of the items into the bins that maximizes total profit, subject to the constraint that no bin is overfilled. A GAP instance with a single bin is equivalent to the knapsack problem, and a GAP instance with unit profit can be interpreted as a decision version of the bin packing problem: can all items be packed in the m bins? The SAP problem can be seen as a variant of GAP with unit item sizes and more strict capacity constraints.

SAP is also related to the Maximum Coverage problem (MC). In the classic version of the MC problem one is given a collection of sets $\mathcal{S} = \{S_1, ..., S_m\}$ and a budget B. The goal is to select a subcollection $\mathcal{S}' \subseteq \mathcal{S}$ with cardinality less than or equal to B such that $|\cup_{S \in \mathcal{S}'} S|$ is maximized. In Section 2.2 we show how the existing hardness of approximation bounds for MC can be extended to SAP.

The algorithms with the best approximation ratio for both MC and GAP are greedy algorithms and the approximation bounds have been proved with similar techniques. In this chapter we show how to extend these analysis techniques to SAP.

2.1.1 Related Work.

In [8] the authors show that SAP is NP-complete even when $K_b = \{0, 3\}$ for all $b \in B$ and $p(i, b) \in \{0, 1\}$ for any $i \in I$. Moreover, they show that SAP does not admit a PTAS by

providing a gap-preserving reduction from the 3-bounded 3-dimensional matching problem. In [15] the authors investigate the approximability of the problem and provide a randomized algorithm which they claim outputs a solution that in expectation has value at least 1/3.93 of the optimal. In [16] this result is revised and the authors show that for any $c \geq 2$, their randomized algorithm outputs a feasible solution with probability at least $1 - \min\{\frac{1}{c}, \frac{e^{c-1}}{c^c}\}$ and has an approximation ratio of $\frac{e-1}{(2c-1)\cdot e}$.

The GAP is well studied in the literature, with [17] and [18] surveying the existing algorithms and heuristics for multiple variations of the problem. In [19] the authors provide a 2-approximation algorithm for the problem and in [20] it is shown that any α-approximation algorithm to the knapsack problem can be transformed into a $(1 + \alpha)$-approximation algorithm for GAP. In [21] tight bounds for the GAP are given showing that no polynomial time algorithm can guarantee a solution within a factor better than $1 - e^{-1}$, unless $P = NP$, and providing an LP-based approximation which for any $\epsilon > 0$ outputs a solution with profit within a $1 - e^{-1} - \epsilon$ factor of the optimal solution value.

The GAP with minimum quantities, in which a bin cannot be used if it is not packed at least above a certain threshold, is introduced in [8]. Because items have arbitrary size, it is easy to see that when a single bin is given and the lower bound threshold equals the bin capacity, finding a feasible solution with profit greater than zero is equivalent to solving Subset Sum. Therefore, in its most general case the problem cannot be approximated in polynomial time, unless $P = NP$.

In [22] and [23] the authors study the problem of maximizing a non-decreasing submodular function f satisfying $f(\emptyset) = 0$ under a cardinality constraint. They show that a simple greedy algorithm achieves an approximation factor of $1 - e^{-1}$ which is the best possible under standard assumptions. Vohra and Hall note that the classic version of the maximum coverage problem belongs to this class of problems [24]. When each set S_i in the MC problem is associated with a cost $c(S_i)$ the Budgeted Maximum Coverage problem asks to

find a collection of sets $\mathcal{S}'$ covering the maximum number of elements under the (knapsack) constraint that $\sum_{S_i \in \mathcal{S}'} c(S_i) \leq B$ for some budget $B \in \mathbb{R}$. Khuller et. al. [25] show that the greedy algorithm combined with a partial enumeration of all solutions with small cardinality also achieves a $1 - e^{-1}$ approximation guarantee, and provide matching lower bounds which hold even in the setting of the classic MC problem (when all sets have unit cost). In [26] Sviridenko generalizes the algorithm and proof technique to show that maximizing any monotone submodular function under a knapsack constraint can be approximated within $1 - e^{-1}$ as well.

2.1.2 Contributions.

In Section 2.2, by a reduction from the Maximum Coverage problem, we show that there exists no polynomial time algorithm that guarantees an approximation factor larger than $1-e^{-1}$, unless $NP \subseteq DTIME(n^{\log \log n})$. Section 2.3 introduces some of the techniques used in the later sections and shows that a greedy algorithm provides a $1 - e^{-1}$ approximation guarantee when the allowable number of workers that can be assigned to any team b is a set $K_b = \{0, k_b\}$ for some integer k_b. In Section 2.4 we present a greedy algorithm that outputs a solution that has profit at least $\frac{1}{2} \cdot (1 - e^{-1})$ of the optimal solution. The algorithm is based on the observation that when the required number of workers in each team is fixed, the problem is solvable in polynomial time. Finally, in Section 2.5 we show how this algorithm can be improved to guarantee an approximation bound of $1 - e^{-1}$.

2.2 Hardness of Approximation

In this section we show that the problem is hard to approximate within a factor of $(1-e^{-1}+\epsilon)$, $\forall \epsilon > 0$, even for the case when for each $b \in B$ the set K_b equals $\{0, n\}$ for some integer n, and the profit for assigning any worker to any team is either 0 or 1. We prove this result by

showing that such restricted instances of SAP are as hard to approximate as the MAXIMUM COVERAGE problem defined below.

Definition 2.2.1. *Given a collection of sets $\mathcal{S} = \{S_1, ..., S_m\}$ and an integer k, the* MAXIMUM COVERAGE *(MC) problem is to find a collection of sets $\mathcal{S}' \subseteq \mathcal{S}$ such that $|\mathcal{S}'| \leq k$ and the size of the union of the sets in $\mathcal{S}'$ is maximized.*

In [25] it is shown that the MC problem is hard to approximate within a factor of $1 - e^{-1} + \epsilon$, unless $NP \subseteq DTIME(n^{\log \log n})$. We use this result to prove the following:

Theorem 2.2.1. *For any $\epsilon > 0$ the SAP problem is hard to approximate within a factor of $1 - e^{-1} + \epsilon$ unless $NP \subseteq DTIME(n^{\log \log n})$.*

Proof. To prove the theorem we create a SAP instance for any given MC instance and show that from any solution of the SAP instance we can create a solution for the MC instance with at least equal value, and that the optimal solution of the SAP instance has value at least equal to the optimal solution of the MC instance. Therefore, an α-approximation algorithm for SAP can be transformed into an α-approximation algorithm for MC.

Given a MC instance, let $U = \cup_{S \in \mathcal{S}} S$ and $n = |U|$. For each set $S \in \mathcal{S}$ let b_S be a team with the allowable number of workers $K_b = \{0, n\}$, and for each element $e \in U$ let i_e be a worker in I. The profit of a worker i_e assigned to a team b_S is 1 if the element e belongs to the set S and 0 otherwise. In addition, let $d_1, ..., d_{n*(k-1)}$ be dummy workers that have profit 0 for any team.

We first show that any feasible assignment $\mathcal{A}$ corresponds to a valid solution to the given MC instance. Since every team requires exactly n workers and there are exactly $k \cdot n$ workers available, clearly at most k teams can be assigned workers in any feasible assignment. Let $\mathcal{S}' = \{S \in \mathcal{S} : \mathcal{A}(b_S) > 0\}$. It is easy to see that the number of elements in $\cup_{S \in \mathcal{S}'} S$ is at least equal to the profit $p(\mathcal{A})$ since a worker i_e has profit 1 for a team b_S only if the set S covers element e.

It remains to show that for any solution to the MC instance there exists a solution to the corresponding SAP instance with the same value. Fix a collection of sets $\mathcal{S}' \subseteq \mathcal{S}$ with $|\mathcal{S}'| \leq k$. For every $e \in \cup_{S \in \mathcal{S}'} S$ let S_e be a set in $\mathcal{S}'$ that contains e and let $\mathcal{A}(i_e) = b_{S_e}$. Then, assign additional dummy workers to any team with at least one worker to reach the required n workers per team. Clearly, the profit of the assignment $\mathcal{A}$ is equal to the number of elements covered by the collection $\mathcal{S}'$, which proves the theorem. □

2.3 Teams of Fixed Size

In this section we show that when the allowable number of workers that can be assigned to any team b is a set $K_b = \{0, k_b\}$ for some integer k_b, SAP can be approximated within a factor of $1 - e^{-1}$ in polynomial time. This introduces some of the techniques used in the general case in a simpler setting.

For an instance of the SAP, a *team-size selection* is a function $S : B \to \mathbb{N}$ with the property that $S(b) \in K_b$ for any $b \in B$. We say that S is feasible if $\sum_{b \in B} S(b) \leq |I|$. In other words, a team-size selection is a function that maps each team to the number of workers to be assigned to it. A team-size selection S *corresponds* to an assignment $\mathcal{A}$ if for any team b the number of workers assigned by $\mathcal{A}$ to b is $S(b)$. We slightly abuse notation and denote by $p(S)$ the maximum profit over all team assignments corresponding to the team-size selection S; we call $p(S)$ the profit of S. In the remainder of this chapter for a graph $G = (V, E)$ we denote the subgraph induced by the vertices of $X \subseteq V$ by $G[X]$.

Definition 2.3.1. *Given a SAP instance let $V_b = \{v_{b,1}, ..., v_{b,k_b}\}$ for every $b \in B$ and let $V = \bigcup_{b \in B} V_b$. The* bipartite representation *of the instance is the complete bipartite graph $G = (V \cup I, E)$ with edge weights $\omega(v_b, i) = p(i, b)$ for every $v_b \in V_b$. The* bipartite representation *of a team-size selection S is the graph $G[V_S \cup I]$ where $V_S = \bigcup_{b \in B} V_{S,b}$ and $V_{S,b} = \{v_{b,1}, ..., v_{b,S(b)}\}$ for every $b \in B$.*

Lemma 2.3.1. *For any SAP instance and any feasible team-size selection S, $p(S)$ is equal to the value of the maximum weight matching in the bipartite representation of S.*

Proof. Let $G_S = (V_S \cup I, E)$ be the bipartite representation of S. First observe that any matching M of G_S that matches all the vertices of V_S can be interpreted as an assignment $\mathcal{A}_M$ of equal value by setting $\mathcal{A}_M(i) = b$ whenever vertex $i \in I$ is matched by M to a vertex in $V_{S,b}$. Since G_S is complete and has non-negative edge weights, there exists a maximum weight matching that matches all the vertices of V_S.

Similarly, any feasible assignment for the SAP instance can be interpreted as a matching $M_{\mathcal{A}}$ of equal value, which proves the lemma. □

Definition 2.3.2. *For a given finite set A, a set function $f : 2^A \to \mathbb{R}$ is submodular if for any $X, Y \subseteq A$ it holds that:*

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y).$$

Sviridenko shows that certain submodular functions can be maximized under knapsack constraints, which will be useful in proving Theorem 2.3.4:

Theorem 2.3.2 ([26])**.** *Given a finite set A, a submodular, non-decreasing, non-negative, polynomially computable function $f : 2^A \to \mathbb{R}$, a budget $L \geq 0$, and costs $c_a \geq 0$, $\forall a \in A$, the following optimization problem is approximable within a factor of $1 - e^{-1}$ in polynomial time:*

$$\max_{X \subseteq A} \left\{ f(X) : \sum_{x \in X} c_x \leq L \right\}$$

We relate now the value of a maximum weight matching in a bipartite graph to the notion of submodularity.

Definition 2.3.3. *For an edge weighted bipartite graph $G = (A \cup B, E)$, the* partial maximum

weight matching function $f : 2^A \to \mathbb{R}$ *maps any set* $S \subseteq A$ *to the value of the maximum weight matching in* $G[S \cup B]$.

Lemma 2.3.3. *Let* f *be the partial maximum weight matching function for a bipartite graph* $G = (A \cup B, E)$ *with non negative edge weights. Then* f *is submodular.*

Proof. Fix two sets $X, Y \subseteq A$ and let $M_\cap$ and $M_\cup$ be two matchings for the graphs $G[(X \cap Y) \cup B]$ and $G[(X \cup Y) \cup B]$ respectively. To prove the lemma it is enough to show that it is possible to partition the edges in $M_\cap$ and $M_\cup$ into two disjoint matchings M_X and M_Y for the graphs $G[X \cup B]$ and $G[Y \cup B]$ respectively.

The edges of $M_\cap$ and $M_\cup$ form a collection of alternating paths and cycles. Let $\mathcal{C}$ denote this collection and observe that no cycle of $\mathcal{C}$ contains vertices from $X \setminus Y$ or $Y \setminus X$. This holds because $M_\cap$ does not match those vertices.

Let $\mathcal{P}_X$ be the set of paths in $\mathcal{C}$ with at least one vertex in $X \setminus Y$ and let $\mathcal{P}_Y$ be the set of paths in $\mathcal{C}$ with at least one vertex in $Y \setminus X$. Two such paths are depicted in Fig. 2.1.

Claim 1. $\mathcal{P}_X \cap \mathcal{P}_Y = \emptyset$.

Proof of claim: Assume by contradiction that there exists a path $P \in \mathcal{P}_X \cap \mathcal{P}_Y$. Let x be a vertex in $X \setminus Y$ on path P and similarly let y be a vertex in $Y \setminus X$ on path P. Observe that since neither x nor y belong to $X \cap Y$ they do not belong to the matching $M_\cap$ by definition, and therefore they are the endpoints of the path P. Moreover, since both x and y are in A, the path P has even length and since it is an alternating path, either the first or last edge belongs to $M_\cap$. Therefore $M_\cap$ matches either x or y contradicting its definition. □

For a set of paths $\mathcal{P}$ we let $E(\mathcal{P}) = \{e \in P : P \in \mathcal{P}\}$. Moreover, let

$$M_X = (E(\mathcal{P}_X) \cap M_\cup) \cup (E(\mathcal{C} \setminus \mathcal{P}_X) \cap M_\cap)$$

and

$$M_Y = (E(\mathcal{P}_X) \cap M_\cap) \cup (E(\mathcal{C} \setminus \mathcal{P}_X) \cap M_\cup).$$

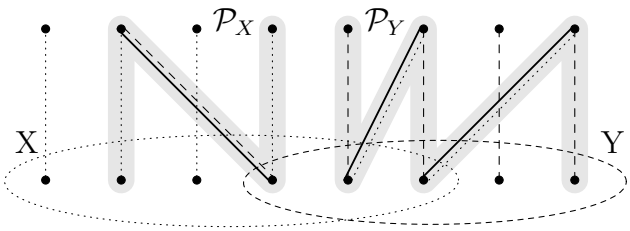


Figure 2.1: $M_{X\cup Y}$ matches each vertex in $X \cup Y$ to the vertex directly above it. $M_{X\cap Y}$ is depicted with contiguous segments, M_X with dotted segments and M_Y with dashed segments. Two alternating paths of $\mathcal{P}$ are shown in light gray.

It is clear that $M_X \cup M_Y = M_\cap \cup M_\cup$ and $M_X \cap M_Y = M_\cap \cap M_\cup$. To prove the theorem it remains to show that M_X and M_Y are valid matchings for $G[X\cup B]$ and $G[Y\cup B]$ respectively. To see that M_X is a valid matching for $G[X \cup B]$ observe first that no vertex of $Y \setminus X$ is matched by M_X since $\mathcal{P}_X$ does not intersect $Y \setminus X$ by Claim 1, and $M_\cap$ does not intersect $Y \setminus X$ by definition. Therefore, M_X only uses vertices of $X \cup B$. Second observe that every vertex $x \in X$ is matched by at most one edge of M_X since otherwise x belongs to either two edges of $M_\cup$ or two edges of $M_\cap$, contradicting the definition. This proves that M_X is a valid matching for $G[X \cup B]$; showing that M_Y is a valid matching for $G[Y \cup B]$ is similar. □

Theorem 2.3.4. *Any instance of SAP in which $|K_b| \leq 2$ for all $b \in B$ can be approximated in polynomial time to a factor of $1 - e^{-1}$.*

Proof. Fix a SAP instance and for any $X \subseteq B$ let S_X be the team-size selection which allocates k_b workers to any team in S and 0 workers to any team in $B \setminus S$. Moreover, let G be the bipartite representation of the SAP instance and f be the partial maximum weight matching function for graph G. Denote by $G[V_X \cup I]$ the bipartite representation of S_X and let $g(X) = f(V_X)$. Since f is submodular by Lemma 2.3.3, it is easy to see that g is submodular as well. Assume by contradiction that there exist sets $X, Y \subseteq B$ such that the

submodularity condition for g doesn't hold:

$$g(X) + g(Y) < g(X \cup Y) + g(X \cap Y). \tag{2.1}$$

Therefore, by definition of g we have

$$f(V_X) + f(V_Y) < f(V_X \cup V_Y) + g(V_X \cap V_Y),$$

contradicting the submodularity of f proven in Lemma 2.3.3.

Clearly g is also monotone, non-negative and polynomially computable. Let $c_b = k_b$, $\forall b \in B$, let $L = |I|$, and observe that S_X is feasible if and only if $\sum_{x \in X} c_x \leq L$. Moreover, by Lemma 2.3.1 and the definition of g, $g(X) = p(S_X)$ whenever the team-size selection S_X is feasible and therefore the proof follows from Theorem 2.3.2. □

2.4 A Constant Factor Greedy Algorithm

The algorithm presented in this section sequentially increments the number of workers allocated to each team in a greedy fashion. It is similar in nature to the greedy algorithm of [25] and [26] but the details of the approximation guarantee proof are different. In the rest of this section we denote by $\mathcal{A}_S$ an optimal assignment for the team-size selection S. Remember that Lemma 2.3.1 shows that given feasible team-size selection S, an optimal team assignment $\mathcal{A}_S$ can be found in polynomial time.

We say that a team-size selection T is greater than a selection S (denoted by $T \succ S$) if $T(b) \geq S(b)$, $\forall b \in B$, and there exists $b \in B$ s.t. $T(b) > S(b)$. The cost of a team-size selection S is denoted by $c(S)$ and equals $\sum_{b \in B} S(b)$. When $T \succ S$ we define the *marginal*

cost of T relative to S as the difference between the cost of T and the cost of S:

$$c_S(T) = c(T) - c(S)$$

Similarly, we define $p_S(T) = p(T) - p(S)$, the *marginal profit* of T relative to S. We say that T is an *incrementing selection* for a team-size selection S if $T \succ S$ and there exists a single team for which the selection T allocates more workers than selection S; more precisely, the cardinality of the set $\{b \in B : T(b) > S(b)\}$ is 1. For a selection S we denote the set of incrementing team-size selections that are feasible by $inc(S)$.

We are now ready to present our algorithm:

GREEDY

1. $S_0 =$ initial team-size selection;
2. $i = 0$;
3. While $inc(S_i) \neq \emptyset$:
   (a) $S_{i+1} \leftarrow \arg\max_{S' \in inc(S_i)} (p(S') - p(S_i))/(c(S') - c(S_i))$;
   (b) $i \leftarrow i + 1$
4. $\mathcal{A}_1 \leftarrow \mathcal{A}_{S_i}$;
5. $\mathcal{A}_2 \leftarrow$ maximum assignment to any single team b for which $S_0(b) = 0$;
6. Return $\arg\max\{p(\mathcal{A}_1), p(\mathcal{A}_2)\}$;

In this section we analyze the algorithm starting from an empty initial team-size selection. In the following section we show that by running the algorithm repeatedly with different initial team-size selections, the approximation guarantee can be improved.

Observe that the cardinality of $inc(S)$ is never greater than $|B|\cdot|I|$ and is therefore polynomial in the size of the input. Thus, using the maximum weight matching reduction from the proof of Lemma 2.3.1, step 3(a) of the algorithm can be performed efficiently.

Definition 2.4.1. *For a team-size selection S and a tuple (b, k_b) with $b \in B$ and $k_b \in \mathbb{N}$, let $S \oplus (b, k_b)$ denote the team-size selection S' with $S'(b) = \max\{k_b, S(b)\}$ and $S'(b') = S(b')$ for any $b' \in B$, $b' \neq b$.*

Lemma 2.4.1. *For any feasible team-size selections S and T, if for every team $b \in B$ the team-size selection $S \oplus (b, T(b))$ is feasible, then it holds that:*

$$\sum_{b \in B} [p(S \oplus (b, T(b))) - p(S)] \geq p(T) - p(S).$$

Proof. For a fixed SAP instance let G be its bipartite representation and let $G[V_S \cup I]$ and $G[V_T \cup I]$ be the bipartite representations of S and T respectively. Moreover, let M_S and M_T be two maximum weight matchings in $G[V_S \cup I]$ and $G[V_T \cup I]$ respectively. Remember that according to Lemma 2.3.1 it holds that $p(S) = \omega(M_S)$ and $p(T) = \omega(M_T)$. To prove the lemma we create matchings $\mathcal{M} = \{M_b\}_{b \in B}$ for the bipartite representations of assignments $p(S \oplus (b, T(b))$, such that each edge of M_T is used in exactly one of the matchings in $\mathcal{M}$ and each edge of M_S is used in exactly $|B|-1$ of the matchings in $\mathcal{M}$.

Let $\mathcal{C}$ be the collection of isolated components formed by the union of the edges of M_S and M_T. Since both M_S and M_T are matchings in G, each element of $\mathcal{C}$ is a path or cycle in G. For every $b \in B$ let $\mathcal{P}_b = \{P \in \mathcal{C} : V(P) \cap V_b \cap (V(M_T) \setminus V(M_S)) \neq \emptyset\}$, where $V(P)$ denotes the vertices of component P (Fig. 2.2).

Claim 2. For any $a \neq b \in B$, $\mathcal{P}_a \cap \mathcal{P}_b = \emptyset$.

Proof: Assume that there exists a path or cycle $P \in \mathcal{P}_a \cap \mathcal{P}_b$ for some $a \neq b \in B$. Then by definition there exist $v_a \in V_a$ and $v_b \in V_b$ such that $v_a, v_b \in V(P)$ and $v_a, v_b \notin V(M_S)$ and therefore v_a and v_b are the endpoints of the alternating path P. Since neither of the

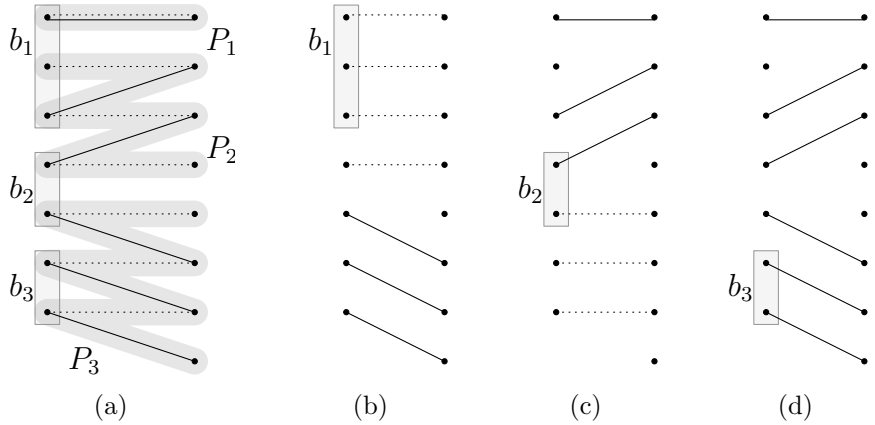


Figure 2.2: An example with 3 teams, b_1, b_2, b_3. (a) Two assignments M_S (dashed edges) and M_T (dotted edges); the three alternating paths formed by $M_S \cup M_T$ (light gray). $q(P_1) = b_1$ because it only intersects vertices from V_{b_1}; $q(P_2) = b_1$ because P_2 contains a vertex $V(M_T) \setminus V(M_S)$ that is in V_{b_1}; $r(P_3) = b_2$. (b), (c) and (d) assignments for team-size selections $S \oplus (b_1, 3)$, $S \oplus (b_2, 2)$ and $S \oplus (b_3, 2)$ combining edges of M_S and M_T.

endpoints of the path belong to M_S, P must have an odd number of edges. However, because both endpoints of P belong to the same partition of the bipartite graph G, the path P must have an even number of edges, hence the claim holds by contradiction. □

Let $q : \mathcal{C} \to B$ be a map of the isolated components to the teams with the following properties:

1. $q(P) \in \{b \in B : V(P) \cap V_b \neq \emptyset\}$;

2. if $P \in \mathcal{P}_b$ for any $b \in B$, $q(P) = b$.

Since $\mathcal{P}_b$ are disjoint by the previous claim and since for any team b it holds by definition that $V(P) \cap V_b \neq \emptyset$ whenever $P \in \mathcal{P}_b$, it is clear that such a mapping q exists.

For every $b \in B$ let M_b be the matching of G that uses all the edges of M_T from the alternating paths $P \in \mathcal{C}$ mapped by q to the team b, and all the edges of M_S from the paths

$P \in \mathcal{C}$ mapped by q to some other team:

$$M_b = [M_T \cap E(q^{-1}(b))] \cup [M_S \cap (E(\mathcal{C}) \setminus E(q^{-1}(b)))].$$

Observe that any edge of M_T belongs to at least one matching M_b for some $b \in B$ and that any edge of M_S belongs to all but one of the matchings M_b. Therefore,

$$\sum_{b \in B} \omega(M_b) \geq \omega(M_T) + (|B|-1) \cdot \omega(M_S).$$

Moreover, observe that for each $b \in B$, M_b is a matching in the bipartite representation of the team-size selection $S \oplus (b, T(b))$. Therefore $p(S \oplus (b, T(b))) = \omega(M_b)$ and the lemma follows. □

Lemma 2.4.2. *Let S and T be two team-size selections such that $S \oplus (b, T(b))$ is feasible for every $b \in B$. Let $S^* = \arg\max_{S' \in inc(S)}(p(S') - p(S))/(c(S') - c(S))$. Then it holds that:*

$$\frac{p(S^*) - p(S)}{c(S^*) - c(S)} \geq \frac{p(T) - p(S)}{c(T)}.$$

Proof. By Lemma 2.4.1 we have that

$$\sum_{b \in B} [p(S \oplus (b, T(b))) - p(S)] \geq p(T) - p(S). \tag{2.2}$$

Since $\sum_{b \in B}[c(S \oplus (b, T(b))) - c(S)] \leq \sum_{b \in B} T(b) = c(T)$, inequality (2.2) implies that

$$\frac{\sum_{b \in B}[p(S \oplus (b, T(b))) - p(S)]}{\sum_{b \in B}[c(S \oplus (b, T(b))) - c(S)]} \geq \frac{p(T) - p(S)}{c(T)}. \tag{2.3}$$

Then, there exists at least one team $b^* \in B$ such that

$$\frac{p(S \oplus (b^*, T(b^*))) - p(S)}{c(S \oplus (b^*, T(b^*))) - c(S)} \geq \frac{p(T) - p(S)}{c(T)}. \tag{2.4}$$

Since $S \oplus (b^*, T(b^*))$ is clearly in $inc(S)$ the lemma follows directly from Eq. (2.4) and the definition of S^*. □

Lemma 2.4.3. *Let T be a feasible team-size selection and let $r \in \mathbb{N}$ be such that $S_i \oplus (b, T(b))$ is feasible for every $i < r$ and $b \in B$. Then for each $i \leq r$ the following holds:*

$$p(S_i) - p(S_0) \geq \left[1 - \prod_{k=0}^{i-1} \left(1 - \frac{c(S_{k+1}) - c(S_k)}{c(T)}\right)\right] \cdot \Big(p(T) - p(S_0)\Big).$$

Proof. We prove the lemma by induction on the iterations i. By the definition of the algorithm, S_1 is the team-size selection with maximum marginal density in $inc(S_0)$, and thus Lemma 2.4.2 shows that the inequality holds for $i = 1$. Suppose that the lemma holds for iterations $1, ..., i$. We show that it also holds for iteration $i + 1$. For ease of exposition, for the remainder of this proof let $\alpha_i = \frac{c(S_{i+1}) - c(S_i)}{c(T)}$.

$$
\begin{aligned}
p(S_{i+1}) - p(S_0) &= p(S_i) - p(S_0) + p(S_{i+1}) - p(S_i) \\
&\geq p(S_i) - p(S_0) + \alpha_i \cdot (p(T) - p(S_i)) \\
&= (1 - \alpha_i)p(S_i) + \alpha_i \cdot p(T) - p(S_0) \\
&\geq (1 - \alpha_i) \cdot \left(1 - \prod_{k=0}^{i-1} (1 - \alpha_k)\right) (p(T) - p(S_0)) \\
&\quad + (1 - \alpha_i) \cdot p(S_0) + \alpha_i \cdot p(T) - p(S_0) \\
&= \left(1 - \alpha_i - \prod_{k=0}^{i} (1 - \alpha_k)\right) (p(T) - p(S_0)) \\
&\quad + \alpha_i \cdot (p(T) - p(S_0)) \\
&= \left(1 - \prod_{k=0}^{i} (1 - \alpha_k)\right) (p(T) - p(S_0)),
\end{aligned}
$$

where the first inequality follows from Lemma 2.4.2 and the second inequality follows from the induction hypothesis. □

Theorem 2.4.4. *When S_0 is the empty assignment the* **Greedy** *algorithm is a $\frac{1}{2} \cdot (1 - e^{-1})$ approximation for SAP.*

Proof. Let OPT be the team-size selection of a fixed optimal assignment solution for the given SAP instance. Let $b^* \in B$ be the team that is allocated the most workers in OPT and let OPT' be the team-size selection for which $OPT'(b^*) = 0$ and $OPT'(b) = OPT(b)$ for any $b \neq b^* \in B$. Let r be the first iteration of the algorithm for which $c(S_r) > c(OPT')$. Clearly, $S_i \oplus (b, OPT(b))$ is feasible for every $i < r$ and $b \in B$. Since $p(S_0) = 0$, by applying Lemma

2.4.3 to iteration r we obtain:

$$p(S_r) \geq \left[1 - \prod_{k=0}^{r-1}\left(1 - \frac{c(S_{k+1}) - c(S_k)}{c(OPT')}\right)\right] \cdot p(OPT')$$
$$\geq \left[1 - \prod_{k=0}^{r-1}\left(1 - \frac{c(S_{k+1}) - c(S_k)}{c(S_r)}\right)\right] \cdot p(OPT'). \tag{2.5}$$

Observe that $c(S_r) = \sum_{k=0}^{r-1} c(S_{k+1}) - c(S_k)$ and that for any real numbers $a_0, ..., a_{r-1}$ with $\sum_{k=0}^{r-1} a_k = A$ it holds that:

$$\prod_{k=0}^{r-1}\left(1 - \frac{a_k}{A}\right) \leq \left(1 - \frac{1}{r}\right)^r < e^{-1}. \tag{2.6}$$

Therefore Eq. (2.5) implies $p(S_r) > (1 - e^{-1}) \cdot p(OPT')$. Since the profit of $\mathcal{A}_2$ is at least $p(b^*, OPT(b^*))$ it holds that

$$p(\mathcal{A}_1) + p(\mathcal{A}_2) > (1 - e^{-1}) \cdot p(OPT') + p(b^*, OPT(b^*))$$
$$\geq (1 - e^{-1}) \cdot p(OPT)$$

and therefore either $p(\mathcal{A}_1)$ or $p(\mathcal{A}_2)$ is greater than or equal to $\frac{1}{2} \cdot (1 - e^{-1})p(OPT)$. □

2.5 Improving the Approximation

In this section we show that the algorithm can be improved by starting the greedy algorithm not from an empty team-size selection, but from a team-size selection that is part of the optimal solution. The improved algorithm is less efficient but achieves the optimal approximation ratio of $1 - e^{-1}$. Let $\mathcal{A}_{opt}$ be an optimal team assignment and for any $b \in B$ let

$p_{opt}(b)$ be the profit obtained in this assignment from team b:

$$p_{opt}(b) = \sum_{i \in \mathcal{A}_{opt}^{-1}(b)} p(i, b).$$

Clearly, the profit of the optimal solution is $\sum_{b \in B} p_{opt}(b)$. W.l.o.g, let b_1, b_2, b_3 be the three teams of the optimal solution with highest profit and let S^* be a team-size selection such that $S^*(b) = OPT(b)$ if $b \in \{b_1, b_2, b_3\}$, and $S^*(b) = 0$ otherwise.

Theorem 2.5.1. *When $S_0 = S^*$ the* **Greedy** *algorithm is a $(1 - e^{-1})$-approximation for SAP.*

Proof. Let OPT be the team-size selection corresponding to $\mathcal{A}_{opt}$. Let b^* be the team that is allocated the most workers in OPT and is not allocated workers in S^*. Moreover, let OPT' be the team-size selection for which $OPT'(b^*) = 0$ and $OPT'(b) = OPT(b)$ for any $b \neq b^* \in B$. Let r be the first iteration of the algorithm for which $c(S_r) > c(OPT')$. Clearly, the team-size selection $S_i \oplus (b, OPT(b))$ is feasible for every $i < r$ and $b \in B$. By applying Lemma 2.4.3 to iteration r we obtain:

$$\begin{aligned} p(S_r) - p(S^*) &\geq \left[1 - \prod_{k=0}^{r-1} \left(1 - \frac{c(S_{k+1}) - c(S_k)}{c(OPT')}\right)\right] \cdot \Big(p(OPT') - p(S^*)\Big) \\ &\geq \left[1 - \prod_{k=0}^{r-1} \left(1 - \frac{c(S_{k+1}) - c(S_k)}{c(S_r)}\right)\right] \cdot \Big(p(OPT') - p(S^*)\Big). \end{aligned}$$

By applying Eq. (2.6) we obtain that

$$p(S_r) - p(S^*) \geq (1 - 1/e) \cdot \Big(p(OPT') - p(S^*)\Big),$$

and therefore

$$\begin{aligned} p(S_r) &\geq (1 - 1/e) \cdot p(OPT') + p(S^*)/e \\ &\geq (1 - 1/e) \cdot p(OPT) - p_{opt}(b^*) + p(S^*)/e. \end{aligned} \tag{2.7}$$

By hypothesis S^* selects the three teams with maximum profit in the optimal assignment and allocates exactly as many workers to each as OPT does. Then, since $p_{opt}(b^*) \leq p_{opt}(b_i)$ for $i = 1, ..., 3$ it holds that $p(S^*) \geq 3 \cdot p_{opt}(b^*) > e \cdot p_{opt}(b^*)$ and the theorem follows. □

Observe that the number of feasible team-size selections assigning workers to at most three teams is polynomial in the size of the input. Therefore, by repeatedly calling the greedy algorithm with all possible such selections our main result follows:

Corollary 2.5.2. *There exists a polynomial time* $(1-e^{-1})$*-approximation algorithm for SAP.*

2.6 Conclusions

In this chapter we presented a $(1 - 1/e)$-approximation algorithm for the Seminar Assignment Problem and provided matching lower bounds. As mentioned in the introduction, our algorithm analysis is similar in nature to the analysis of the greedy algorithms for a few other different problems. To simplify future research it would be very useful to identify the most general set of requirements that make these proof techniques work.

Chapter 3

Team Cohesion

3.1 Introduction

In this chapter, based on work published at ESA 2015 [27], we consider a setting where a social network of workers must be partitioned into small teams with the goal of maximizing team cohesion. More precisely, assume that a social network of workers is given as a complete weighted graph G, where each node of G represents a worker and the weight of the edge between two workers represents the compatibility between the two workers. We define the cohesion of a team with a leader as the sum of the compatibility of each worker with the team leader, and the cohesion of a flat team (with no team leader) as the sum of the compatibility of every pair of workers within the team. In this chapter we present some algorithms for partitioning the social network in small teams with and without leader, with the goal of maximizing the sum of the teams' cohesiveness, as defined above.

Observe that if one desires to partition the network into teams of size 2, there is no distinction between teams with leader and teams without leader. Moreover, the problem is equivalent with the well studied Maximum Weight Matching problem which can be solved optimally in polynomial time [28, 29]. Therefore we focus our attention to teams of size

3. Observe that in this case allocating workers to teams with a leader is equivalent to partitioning the soial network graph into paths of length 2, and allocating the workers to leader-less teams of size 3 is equivalent to partitioning the graph into triangles. Both problems are NP-complete and have been studied before from an approximation perspective [9, 11–13].

3.1.1 Problem definition

For some integer $k > 0$ let G be a complete graph on $3k$ vertices having non-negative edge weights. The Maximum Weight 2-path Partition (M2PP) problem asks to compute a set of k vertex disjoint paths of length 2 (referred to as 2-paths) such that the sum of the weights of the paths is maximized. The Maximum Weight Triangle Partition (MTP) problem asks to compute a set of k vertex disjoint cycles of length 3 (referred to as triangles) such that the sum of the edge weights of the k cycles is maximized. For the remainder of this work G will refer to a complete graph on $3k$ vertices having non-negative edge weights.

Our main contribution is a 7/12-approximation algorithm for the M2PP problem. We also investigate $\{0, 1\}$-edge-weighted graphs – in which the weights of the edges of the input graph are either 0 or 1 – and we present a .75-approximation algorithm for the M2PP problem and a .6-approximation algorithm for the MTP problem in this setting.

Both the M2PP and MTP problems have been studied before, mostly under the names Maximum 2-Path Packing and Maximum Triangle Packing respectively. Unfortunately, these names have also been used for the related but different problems defined below. Consequently, we use separate terminology to make a clear distinction between the packing and partitioning settings.

Given an unweighted graph H, a 2-Path Packing of H is a collection of vertex disjoint 2-paths and a Triangle Packing of H is a collection of vertex disjoint triangles. Such a collection is called *perfect* if it uses all the vertices of H. The Maximum 2-Path

Packing problem asks to find a 2-Path Packing of maximum cardinality. The Maximum Triangle Packing problem is defined similarly.

3.1.2 Related work.

In their classic book Garey and Johnson [30] show that deciding whether a graph admits a perfect Triangle Packing or a perfect 2-Path Packing is NP-complete (p. 68 and 76 respectively). More general results on the NP-completeness of packing families of graphs into a given graph are shown by Hell and Kirkpatrick [31], and Lonc [32]. Both the Maximum 2-Path Packing and Maximum Triangle Packing problems are special cases of the unweighted 3-Set Packing problem for which Hurkens and Schrijver [33] (also see Halldórsson [34]) presents a local search algorithm that achieves a $\frac{2}{3} - \epsilon$ approximation (with $\epsilon > 0$).

Kann [35] shows that the Maximum Triangle Packing is APX-hard even in graphs of maximum degree 4 and Chlebík and Chlebíková [36] show that Maximum Triangle Packing is NP-hard to approximate within a factor of .9929. Moreover, Guruswami et. al. [37] show that the problem remains NP-complete even when restricted to the families of chordal, planar, line or total graphs. A .833-approximation algorithm for graphs with maximum degree 4 is presented by Manić and Wakabayashi [38].

For Maximum 2-Path Packing problem van Bevern et. al. [39] study its computational complexity on multiple classes of special graphs. The authors provide a quasilinear-time algorithm for the problem on interval graphs and polynomial time algorithms for the more general Star Partition problem on cographs and on bipartite permutation graphs. Moreover, the authors show that the problem is NP-hard even on grid graphs with maximum degree three. Prieto and Sloper [40] present a fixed parameter tractable algorithm and Babenko and Gusakov [41] present an approximation algorithm for an edge-weighted version of the problem.

The M2PP and MTP problems studied in this chapter are special cases of the weighted 3-Set Packing problem for which Arkin and Hassin [42] present a $\frac{1}{2} - \epsilon$ approximation algorithm. M2PP and MTP are special cases because all sets of size three exist and have non-negative weights, and the weight of each set is determined by the weights of the included subsets of size two. Hassin et al. [9] observe that there exists a simple reduction from M2PP (respectively, MTP) to the problem of deciding whether a graph has a perfect 2-Path Packing (resp., Triangle Packing), implying that the two problems are NP-complete. Moreover, they present a randomized $\frac{35}{67} - \epsilon \approx .5222$ approximation algorithm for M2PP and Tanahashi and Chen [10] refine and derandomize this algorithm, leading to an improved approximation ratio of $.5265-\epsilon$; van Zuylen [11] presents a simpler analysis of this algorithm. For the MTP problem, Hassin and Shlomi Rubinstein [9] (see also Erratum [43]) presents a $\frac{43}{83} - \epsilon \approx .518$ approximation algorithm and Chen et. al. [13] (see also Erratum [44]) present a randomized approximation algorithm which achieves a ratio of .5257. For $\{0, 1\}$-edge-weighted graphs, Hassin and Schneider [12] present a local search based .55-approximation algorithm for M2PP which runs in time $O(|V|^{10})$. Hassin and Rubinstein [45] also study the problem of partitioning a complete weighted graph into paths of length 3, and present a .75-approximation algorithm.

3.1.3 Contribution.

In this chapter we present a simple, matching based $7/12 \approx .583$-approximation algorithm for the M2PP problem on graphs with general non-negative weights, improving upon the $(.5265 - \epsilon)$-approximation algorithm of Tanahashi and Chen [10]. Besides improving the approximation ratio, our algorithm is significantly less computationally intensive since the constant factor for the algorithm of Tanahashi and Chen is exponential in $1/\epsilon$. Moreover, for $\{0, 1\}$-edge-weighted graphs we provide a .75-approximation algorithm for the M2PP problem improving upon the .55-approximation algorithm of Hassin and Schneider [12]. The

core idea of our algorithms is adapted from Hassin and Rubinstein [45], where the authors show how to partition a complete weighted graph into paths of length 3. For a complete graph on $n = 3k$ vertices we prove the following two theorems in Section 3.3 and Section 3.4 respectively.

Theorem 3.1.1. *There exists a* $7/12$*-approximation algorithm for the* M2PP *problem running in time* $O(n^{2.5})$*.*

Theorem 3.1.2. *For* $\{0, 1\}$*-edge-weighted graphs there exists a* $3/4$*-approximation algorithm for the* M2PP *problem running in time* $O(n^{2.5})$*.*

In Section 3.5 we show how an approximation algorithm for M2PP can be combined with a 3-Set Packing approximation algorithm to obtain an approximate solution for the MTP on $\{0, 1\}$-edge-weighted graphs. We are not aware of any previous results for the MTP problem restricted to this case.

Theorem 3.1.3. *For* $\{0, 1\}$*-edge-weighted graphs there exists a* $5/8$*-approximation algorithm for the* MTP *problem running in time* $O(n^{2.5})$*.*

3.1.4 Motivation.

Besides being interesting variants of the Maximum 2-Path Packing and Maximum 2-Path Packing problems, M2PP and MTP are natural special cases of the Team Formation problem. Given a social network of experts, the Team Formation problem asks to find the most cohesive team. Some authors [46–49] measure the cohesiveness of a team as the number of connections within a team, while others [39, 47] consider only connections between a team leader and the other team members. Experimental evidence presented by Baumer et. al. [50] suggests that indeed, not all ties between team members are of equal importance, and maximizing the connection between the team leader and the rest of the group

suffices. If we are interested in forming multiple teams, the TEAM FORMATION problem for teams of size 3 can be cast as either M2PP or MTP, depending on whether we are forming teams with or without a leader respectively.

3.2 Preliminaries

Some of the graphs we argue about in this chapter refer to non-simple graphs that have parallel edges and self-loops. To avoid confusion we begin by formally defining the notion of a graph as used in the remaining of this work. A *graph* $G = (V, E, \gamma)$ is a set of vertices V together with a set of edges E and a function $\gamma : E \to \{\{u, v\} : u, v \in V\}$. For an edge $e \in E$ with $\gamma(e) = \{u, v\}$ we call the vertices u and v the *endpoints* of the edge e. Two edges $e_1, e_2 \in E$ are called *parallel* if $\gamma(e_1) = \gamma(e_2)$, and an edge $e \in E$ is called a *loop* if $\gamma(e) = \{v\}$ contains a single element. For clarity, we sometimes use $V(G)$ and $E(G)$ to denote the vertex and edge set of the graph G respectively. We say that the graph G is *complete* if for any vertices $u \neq v$ in $V(G)$ there exists an edge $e \in E(G)$ such that $\gamma(e) = \{u, v\}$, and we say that the graph G is *simple* if it contains no loops or parallel edges. In an edge-weighted graph each edge $e \in E$ is associated with a weight $\omega(e)$. We slightly abuse notation by using $\omega(A)$ to denote the sum of the weights of the edges in an edge set A, and by using $V(A)$ to denote the set of endpoints of edges in A. Moreover, we use $\omega(G)$ to denote the sum of the weights of the edges in $E(G)$. For a set of 2-paths Π we use $E(\Pi)$ to denote the set of edges used by the 2-paths of Π and $\omega(\Pi)$ to denote the sum of the weights of the edges in $E(\Pi)$. For an edge $e \in E$, let V_e, E_e and γ_e be defined as follows:

- $V_e = (V \setminus \gamma(e)) \cup \{v_e\}$ for some new vertex $v_e \notin V$. We say that e is the edge of G *corresponding* to v_e, and vice versa.
- $E_e = E \setminus \{e\}$

- for any edge $f \in E_e$, $\gamma_e(f)$ is defined as:

$$\gamma_e(f) = \begin{cases} \gamma(f) & \text{if } \gamma(f) \cap \gamma(e) = \emptyset, \\ (\gamma(f) \setminus \gamma(e)) \cup \{v_e\} & \text{otherwise.} \end{cases}$$

We define $G/e = (V_e, E_e, \gamma_e)$, the graph resulting from *contracting* the edge e. For a set of edges A we denote by G/A the graph obtained from G by sequentially contracting all edges of A.

3.3 M2PP in graphs with non negative edge-weights

In this section we present and analyze the **WeightedDoubleMatching** approximation algorithm for the Maximum Weight 2-path Partition (M2PP) problem in graphs with general non-negative weights. We first restrict our attention to the case when k is even. In section 3.3.7 we show how to extend the algorithm to graphs with odd k.

The **WeightedDoubleMatching** algorithm (Fig. 3.1) takes as input a weighted complete simple graph $G = (V, E, \omega_G, \gamma_G)$, with $|V| = 3k$, and in the first step it computes a perfect maximum weight matching M_1 of G. If needed, M_1 may use edges of weight 0 so that all vertices are matched. Therefore, the size of M_1 is $3k/2$. In the second step, it contracts the edges of M_1 to obtain a graph H and assigns to each edge e in H the weight

$$\omega_H(e) \equiv \omega_G(e) - \min\{\omega_G(a), \omega_G(b)\}, \tag{3.1}$$

where a and b are the edges in M_1 corresponding to the endpoints of e. Observe that that H is graph with $|V(H)| = 3k/2$ and since every vertex in H corresponds to two vertices of G, there are four parallel edges between any two vertices of H. Intuitively, the choice of $\omega_H(e)$ internalizes the loss of weight in the final solution due to the fact that for each edge e in

M_2, one of the two edges of M_1 adjacent to e is not being used in the final solution. This intuition is made precise in Lemma 3.3.1.

In the next step, a maximum weight matching M_2 of size exactly $k/2$ is computed in H based on the edge weight function ω_H. Observe that some of the edges in M_2 may have negative weights in H. The matching M_2 determines how the edges of M_1 are combined into the output 2-paths:

- for each edge $e \in M_2$, let $a, b \in M_1$ be the edges corresponding to the endpoints of e, with $\omega_G(a) \geq \omega_G(b)$; create a 2-path $\{a, e\}$, and call the vertex of b not incident to e a *residual vertex*.
- for each edge $a \in M_1$ corresponding to a vertex of H not matched by M_2, create a 2-path from a and an arbitrary residual vertex.

We denote by $\mathcal{A}_1$ the set of 2-paths created that contain an edge of M_2 and the remaining 2-paths by $\mathcal{A}_2$. Notice that the size of $\mathcal{A}_1$ is $k/2$ since the size of M_2 is $k/2$ and each 2-path in this set uses an edge of M_2. Moreover, the size of $\mathcal{A}_2$ is also $k/2$ since there are exactly $k/2$ vertices of H not matched by M_2, and each 2-path in $\mathcal{A}_2$ corresponds to one such vertex. The algorithm outputs the set of 2-paths $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$. A formal description of the algorithm is presented in Fig. 3.1.

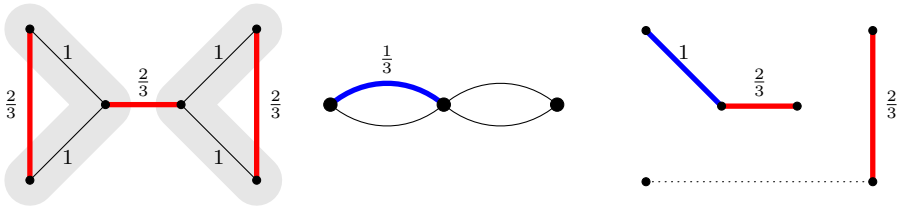
3.3.1 Example.

In Fig. 3.2 we illustrate the main steps of the **WeightedDoubleMatching** algorithm on a simple complete graph on six vertices (therefore $k = 2$). Fig. 3.2(a) shows the complete graph G with the zero weight edges omitted, and for each non-zero edge its corresponding weight. The optimal solution of weight 4 is drawn in light gray and a possible maximum weight matching M_1 is displayed in bold (red). Fig. 3.2(b) shows the positive weight edges of the graph H. All the illustrated edges have weight 1 in the graph G and have weight

Input: complete simple weighted graph $G = (V, E, \omega_G, \gamma_G)$.

1. $M_1 \leftarrow$ maximum weight perfect matching of G of size $3k/2$.
2. $H \leftarrow G/M_1$; H is a complete graph with $3k/2$ vertices and four parallel edges between any two vertices.
3. Let $\omega_H : E(H) \rightarrow \mathbb{R}$ be an edge weight function defined as follows:
$$\omega_H(e) \equiv \omega_G(e) - \min\{\omega_G(a), \omega_G(b)\},$$
where a and b are the edges of M_1 corresponding to the two endpoints of e.
4. $M_2 \leftarrow$ maximum weight matching of H (based on ω_H) with $|M_2| = k/2$.
5. Let Q be the set of $k/2$ vertices of H not matched by M_2.
6. Create two sets of 2-paths $\mathcal{A}_1$ and $\mathcal{A}_2$ as follows:
   - $\mathcal{A}_1$: $k/2$ many 2-paths, each formed from an edge $e \in M_2$ and one of the edges $a, b \in M_1$ corresponding to the endpoints of e. If $\omega_G(a) \geq \omega_G(b)$, create a 2-path from a and e and call the unused vertex of b a *residual* vertex. Denote by R be the set of $k/2$ residual vertices.
   - $\mathcal{A}_2$: $k/2$ many 2-paths, each formed from an edge of G corresponding to a vertex of Q (not matched by M_2), and an arbitrary residual vertex from R.
7. Return $\mathcal{A} \leftarrow \mathcal{A}_1 \cup \mathcal{A}_2$.

Figure 3.1: The **WeightedDoubleMatching** Algorithm



(a) Matching M (b) H and matching N (c) Algorithm output

Figure 3.2: Illustration of a tight example (see Section 3.3.1).

$1 - 2/3$ in H. A maximum matching M_2 of size $k/2 = 1$ is shown in bold (blue). Finally, Fig. 3.2(c) shows a possible output of the algorithm, with the dotted segment denoting an edge of zero weight. The weight of the output solution is $7/3$ which is a $7/12$ fraction of the weight of the optimal solution.

3.3.2 Algorithm analysis overview.

In the remainder of Section 3.3 unless otherwise specified G refers to the complete simple weighted graph given as input to the algorithm, M_1 and M_2 refer to the matchings found by the **WeightedDoubleMatching** in step 1 and 4 respectively, H denotes the graph G/M_1 resulting from contracting the edges of M_1 in G and ω_H is an edge weight function on $E(H)$ as defined in step 3 of the algorithm. Moreover, we denote by Π the set of 2-paths in an arbitrary fixed 2-path partition.

We show that the algorithm outputs a solution of weight at least $7/12 \cdot \omega_G(\Pi)$. In Lemma 3.3.1 we show that the weight of the output solution is at least the weight of the matching M_1 in G plus the weight of the matching M_2 in H, i.e., by denoting the weight of the algorithm solution by $\omega_G(\mathcal{A})$, we show that

$$\omega_G(\mathcal{A}) \geq \omega_G(M_1) + \omega_H(M_2). \tag{3.2}$$

In Lemma 3.3.5 we lower bound the weight of the edges of Π in the graph H

$$\omega_H(E(\Pi) \setminus M_1) \geq \omega_G(\Pi) - 4/3 \cdot \omega_G(M_1), \tag{3.3}$$

and in Lemma 3.3.8 we lower bound the weight of the matching M_2 in H:

$$\omega_H(M_2) \ \geq \ \omega_H(E(\Pi) \setminus M_1)/4. \tag{3.4}$$

Combining inequalities (3.3) and (3.4) it follows that:

$$\omega_H(M_2) \ \geq \ \omega_G(\Pi)/4 - \omega_G(M_1)/3. \tag{3.5}$$

Furthermore, since one can form a matching by selecting the heaviest edge of each 2-path in Π and arbitrarily match the remaining vertices, the following holds:

$$\omega_G(M_1) \ \geq \ \omega_G(\Pi)/2. \tag{3.6}$$

Theorem 3.1.1. WeightedDoubleMatching *is a* $7/12$*-approximation algorithm for the* M2PP *problem and runs in time* $O(k^{2.5})$.

Proof. Observe that the most time consuming part of the WeightedDoubleMatching algorithm is computing the matchings M_1 and M_2, which can be computed in time $O(k^{2.5})$ [29]. The approximation guarantee follows from the following:

$$\begin{aligned} \omega_G(\mathcal{A}) \ &\geq \ \omega_G(M_1) + \omega_H(M_2) \\ &\geq \ 2/3 \cdot \omega_G(M_1) + \omega_G(\Pi)/4 \\ &\geq \ 7/12 \cdot \omega_G(\Pi), \end{aligned}$$

where first inequality is due to Ineq. (3.2), the second inequality is due to Ineq. (3.5) and the last inequality is due to Ineq. 3.6. □

3.3.3 Bounding the algorithm solution weight

Remember that the DoubleMatchingAlgorithm uses only k edges out of the $3k/2$ edges of matching M_1. As stated above, the edge-weight function ω_H is crafted specifically to incorporate this loss in the weight of the matching M_2. The following lemma proves that this is the case.

Lemma 3.3.1. $\omega_G(\mathcal{A}) \geq \omega_G(M_1) + \omega_H(M_2)$.

Proof. There is a one-to-one correspondence between the $k/2$ edges of M_2 and the $k/2$ many 2-paths of $\mathcal{A}_1$. For an edge $e \in M_2$ let $a, b \in M_1$ be the edges of G corresponding to the endpoints of e, and w.l.o.g assume that $\omega_G(a) \geq \omega_G(b)$. If $\pi \in \mathcal{A}_1$ is the 2-path formed from edge e, then by construction $\omega_G(\pi) = \omega_G(a) + \omega_G(e)$. Therefore, since $\omega_G(b) = \min\{\omega_G(a), \omega_G(b)\}$ we have:

$$\begin{aligned} \omega_G(\pi) &= \omega_G(a) + \omega_G(b) + \omega_G(e) - \min\{\omega_G(a), \omega_G(b)\} \\ &= \omega_G(a) + \omega_G(b) + \omega_H(e), \end{aligned}$$

where the second equality follows directly from the definition of ω_H. Summing over all 2-paths of $\mathcal{A}_1$ and denoting by a the edge of M_1 corresponding to a vertex $v_a \in V(M_2)$ we obtain

$$\omega_G(\mathcal{A}_1) = \sum_{v_a \in V(M_2)} \omega_G(a) + \sum_{e \in M_2} \omega_H(e). \tag{3.7}$$

There is also a one-to-one correspondence between the $k/2$ vertices of Q (the vertices of H not matched by M_2) and the $k/2$ many 2-paths of $\mathcal{A}_2$. For a vertex $v_c \in Q$, let $\pi_c \in \mathcal{A}_2$

be the corresponding 2-path and let $c \in M_1$ be the edge corresponding to v_c. Then clearly $\omega_G(\pi_c) \geq \omega_G(c)$ since any 2-path containing edge c has weight at least $\omega_G(c)$. Summing over all 2-paths of $\mathcal{A}_2$ we obtain

$$\omega_G(\mathcal{A}_2) \;\geq \sum_{v_a \in V(H) \setminus V(M_2)} \omega_G(a). \tag{3.8}$$

Combining Eq. (3.7) and (3.8) yields:

$$\omega_G(\mathcal{A}_1) + \omega_G(\mathcal{A}_2) \;\geq \sum_{v_a \in V(H)} \omega_G(a) + \sum_{e \in M_2} \omega_H(e) \;=\; \omega_G(M_1) + \omega_H(M_2)$$

and thus proving the lemma. □

3.3.4 Intermezzo: uniformly c-sparse graphs and fair supply functions.

Our overarching goal is to bound the difference of the weight of the edges of Π in G and the weight of the edges of Π in H. To this end, in this section we extend Courcelle's notion of *uniform c-sparseness* ([51]) to edge-weighted graphs and prove some general facts about edge-weighted uniform c-sparse graphs. The main result of this section is Lemma 3.3.4 which, given a uniformly c-sparse graph G and a node function $\chi : V(G) \to \mathbb{R}$, bounds the following expression:

$$\sum_{e \in E} \min\{\chi(v) : v \text{ endpoint of } e \text{ in } G\}. \tag{3.9}$$

Remember that by definition $\omega_H(e) \equiv \omega_G(e) - \min\{\omega_G(a), \omega_G(b)\}$ and observe that when the set of edges E is set to $E(\Pi)$, expression (3.9) relates to $\omega_G(E(\Pi)) - \omega_H(E(\Pi))$.

In the next section we show that the edges of Π in H form in fact a uniformly 4/3-sparse

subgraph and use Lemma 3.3.4 to upper bound $\omega_H(E(\Pi))$. For the rest of this section, for a graph $G = (V, E, \omega, \gamma)$ and $S \subseteq V$, we denote the graph induced by the vertices of S by $G[S]$.

Definition 3.3.1. *An edge-weighted graph $G = (V, E, \omega, \gamma)$ is called* uniformly c-sparse*, if for any $S \subseteq V$ it holds that $\omega(G[S]) \leq c \cdot |S|$. An unweighted graph G is called* uniformly c-sparse *if for any $S \subseteq V$ it holds that $|E(G[S])| \leq c \cdot |S|$.*

Lemma 3.3.2. *Let $G = (V, E, \omega, \gamma)$ be a weighted uniformly c-sparse graph and let $S, T \subseteq V$ be such that $\omega(G[S]) = c \cdot |S|$ and $\omega(G[T]) = c \cdot |T|$. Then it holds that $\omega(G[S \cup T]) = c \cdot |S \cup T|$.*

Proof. Since G is c-sparse it holds by definition that

$$\omega(G[S \cup T]) \leq c \cdot |S \cup T|. \tag{3.10}$$

Moreover, $\omega(G[S \cup T]) \geq \omega(G[S]) + \omega(G[T]) - \omega(G[S \cap T])$ and since $\omega(G[S]) = c \cdot |S|$ and $\omega(G[T]) = c \cdot |T|$ it holds that

$$\omega(G[S \cup T]) \geq c \cdot |S| + c \cdot |T| - \omega(G[S \cap T]). \tag{3.11}$$

By definition of c-sparseness it holds that $\omega(G[S \cap T]) \leq c \cdot |S \cap T|$, which combined with (3.11) implies

$$\omega(G[S \cup T]) \geq c \cdot |S| + c \cdot |T| - c \cdot |S \cap T| = c \cdot |S \cup T|. \tag{3.12}$$

Inequalities (3.10) and (3.12) prove the lemma. □

We introduce now the notion of a *node supply function* which captures a setting in which the edges of a graph require resources that are provided by each edge's incident nodes. Each

edge's weight represents the amount of resource it needs and a node supply function defines the amount of resource each node provides to each incident edge.

Definition 3.3.2. *For an edge-weighted graph $G = (V, E, \omega, \gamma)$, we define a* node supply function *to be a function $f : V \times E \to \mathbb{R}$ satisfying the following properties:*

1. *for any loop $l \in E$ with $\gamma(l) = \{v\}$, $f(v, l) = \omega(l)$.*
2. *for any edge $e \in E$ with $\gamma(e) = \{u, v\}$ and $u \neq v$, $f(u, e) + f(v, e) = \omega(e)$.*
3. *for any edge e and vertex $v \notin \gamma(e)$, $f(v, e) = 0$.*

In the following lemma we show that for a c-sparse graph there exists a node supply function that distributes the load on the nodes fairly, in the sense that each node supplies at most c units of resource. In Lemma 3.3.4 we use such a fair node supply function in a counting argument to bound Exp. (3.9) which is the main goal of this section.

Lemma 3.3.3. *For any uniformly c-sparse weighted graph $G = (V, E, \omega, \gamma)$ there exists a node-supply function $f : V \times E \to \mathbb{R}$, with*

$$\sum_{e \in E} f(v, e) \leq c, \quad \forall v \in V. \tag{3.13}$$

We call such a function a fair node-supply function*.*

Proof. We prove the lemma by induction over the number of vertices in the graph G. If the graph G contains a single vertex v, all edges of G are loops and therefore the graph is uniformly c-sparse only if $\sum_{e \in E} \omega(e) \leq c$. By choosing $f(v, e) = \omega(e)$ for any $e \in E$, f is a valid node supply function that satisfies Equation (3.13).

Assuming that a supply function satisfying Ineq. (3.13) exists for any graph on $n - 1$ vertices, we show that such a function exists for any graph on n vertices. Let $G_n =$

$(V_n, E_n, \omega_n, \gamma_n)$ be an uniformly c-sparse graph with $|V_n| = n$, and for a vertex $v \in V_n$ denote by $I_n(v)$ the set of edges incident to v in G_n, not including loops.

We create a graph $G_{n-1} = (V_{n-1}, E_{n-1}, \omega_{n-1}, \gamma_{n-1})$ with $n-1$ vertices by removing an arbitrary vertex u and any loops of u from G_n i.e., $V_{n-1} = V_n \setminus \{u\}$ and $E_{n-1} = E_n \setminus \{l : \gamma_n(l) = \{u\}\}$. For any edge $e \in E_{n-1}$ we let $\gamma_{n-1}(e) = \gamma_n(e) \setminus \{u\}$, meaning that all edges in $I_n(u)$ are loops in G_{n-1}.

Observe that if for any edge $e \notin I_n(u)$ we let $\omega_{n-1}(e) = \omega_n(e)$ and for all edges $e \in I_n(u)$ we let $\omega_{n-1}(e) = 0$, the graph G_{n-1} is c-sparse. This is true because for any $S \subseteq V_{n-1}$ it holds that $\omega_{n-1}(G[S]) = \omega_n(G[S]) \leq c \cdot |S|$.

Let $\omega_{n-1}(e) = \omega_n(e)$ for any $e \notin I_n(u)$ and let $\omega_{n-1}(e) \leq \omega_n(e)$ be the maximum value that leaves the graph G_{n-1} c-sparse. Therefore, $\omega_{n-1}(e) < \omega_n(e)$ only if there exists a subset S_e of V_{n-1} such that $\gamma_{n-1}(e) \subseteq S_e$ and $\omega_{n-1}(G_{n-1}[S_e]) = c \cdot |S_e|$.

Let f_{n-1} be a node supply function for the graph G_{n-1} satisfying Equation (3.13). We create a supply function f_n for G_n by letting each node of G_{n-1} supply resources as defined by function f_{n-1}, and letting node u supply any additional resource. To complete the proof we show that the newly created supply function is fair.

For any $v \neq u$ let $f_n(v, e) = f_{n-1}(v, e)$ for all $e \in E_{n-1}$, and $f_n(v, e) = 0$ for any other edge $e \in E_n \setminus E_{n-1}$ (the loops of u). Moreover, let $f_n(u, \cdot)$ be defined as follows:

(a) $f_n(u, e) = 0$ for any $e \in E_n$ with $u \notin \gamma_n(e)$;

(b) $f_n(u, e) = \omega_n(e) - f_n(v, e)$ for any $e \in E_n$ with $\gamma_n(e) = \{u, v\}$ and $u \neq v$;

(c) $f_n(u, e) = \omega_n(e)$ for any $e \in E_n$ with $\gamma_n(e) = \{u\}$.

Claim: f_n is a fair node supply function for the graph G_n.
Fix a vertex $v \in V_n$, $v \neq u$. By definition of f_n it holds that $f_n(v, e) = f_{n-1}(v, e)$ for any edge $e \in E_{n-1}$ and that $f_n(v, e) = 0$ for any edge $e \in E_n \setminus E_{n-1}$. Therefore, $\sum_{e \in E_n} f_n(v, e) = \sum_{e \in E_{n-1}} f_{n-1}(v, e)$ and Equation (3.13) holds by induction hypothesis.

It remains to prove the lemma for vertex u. Remember that $I_n(u)$ denotes the set of non-loop edges incident to u in G_n, and let $\mathcal{E} \subseteq I_n(u)$ be the set of non-loop edges for which $f_n(u,e) > 0$. Therefore, for every $e \in \mathcal{E}$, $\gamma_n(e) = \{u, v\}$, there exists a vertex set $S_e \subseteq V_{n-1}$ with $v \in S_e$, such that $\omega_{n-1}(G_{n-1}[S_e]) = c \cdot |S_e|$. Let $S = \bigcup_{e\in\mathcal{E}} S_e$, then by Lemma 3.3.2 we have that

$$\omega_{n-1}(G_{n-1}[S]) = c \cdot |S|. \tag{3.14}$$

Partition the set of edges of $G_n[S \cup \{u\}]$ into three sets:

- $A := E(G_n[S])$;
- $B := \{e : \gamma(e) \cap S \neq 0; u \in \gamma(e)\}$;
- $C := \{e : \gamma(e) = \{u\}\}$.

Thus, A is the set of edges not incident to u, B is the set of non-loop edges incident to u and C is the set of loops of u. Observe that by definition of ω_{n-1} it holds that

$$\omega_n(A) = \omega_{n-1}(A). \tag{3.15}$$

Observe that by definition of G_{n-1} all edges $e \in B$ with $\gamma_n(e) = \{u, v\}$ are loops of v in G_{n-1}, and remember that by the definition of a node supply function $f_{n-1}(e) = \omega_{n-1}(v)$ for any loop e of v. Therefore, item (b) of the definition of f_n implies that $\omega_n(e) = \omega_{n-1}(e) + f_n(u,e)$ for every $e \in B$ and summing over all edges of B it holds that

$$\omega_n(B) = \omega_{n-1}(B) + \sum_{e\in B} f_n(u,e) \tag{3.16}$$

Moreover, item (c) of the definition of f_n implies $\omega_n(e) = f_n(u,e)$ for every $e \in C$ and thus,

summing over all edges of C it holds that:

$$\omega_n(C) = \sum_{e \in C} f_n(u, e) \tag{3.17}$$

Remember that sets A, B and C form a partition of the edges of $G_n[S]$) and therefore, by summing over Eq. (3.15)-(3.17), it holds that:

$$\begin{aligned}
\omega_n(G_n[S \cup \{u\}]) &= \omega_n(A) + \omega_n(B) + \omega_n(C) \\
&= \omega_{n-1}(A) + \omega_{n-1}(B) + \sum_{e \in B} f_n(u, e) + \sum_{e \in C} f_n(u, e) \\
&= \omega_{n-1}(G_{n-1}[S]) + \sum_{e \in E_n} f_n(u, e),
\end{aligned} \tag{3.18}$$

where the last equality hods because by the definition of the graph G_{n-1} the set of edges of $G_{n-1}[S]$ is the set $A \cup B$ and because the definition of f_n implies that $f_n(u, e) = 0$ for any $e \notin B \cup C$. By using Eq. (3.14) in Eq. (3.18) we obtain

$$\omega_n(G_n[S \cup \{u\}]) = c \cdot |S| + \sum_{e \in E_n} f_n(u, e),$$

and since G_n is a c-sparse graph it holds that $\omega_n(G_n[S \cup \{u\}]) \leq c \cdot (|S|+1)$. Therefore $\sum_{e \in E_n} f_n(u, e) \leq c$, which proves the claim and the lemma. □

We are now ready to prove the main lemma of this section which uses a fair node-supply function to bound Exp. (3.9).

Lemma 3.3.4. *Let $G = (V, E, \gamma, \omega)$ be a uniformly c-sparse graph with $\omega(e) = 1, \forall e \in E$, and let $\chi : V \to \mathbb{R}$ be a vertex weight function. Then,*

$$\sum_{e \in E} \min\{\chi(v) : v \text{ endpoint of } e \text{ in } G\} \leq c \cdot \sum_{v \in V} \chi(v).$$

Proof. Let f be a fair node-supply function for the graph G. Then for every edge $e \in E$ it holds that

$$\begin{aligned} \min\{\chi(v) : v \in \gamma(e)\} &= \sum_{v \in \gamma(e)} f(v,e) \cdot \min\{\chi(v) : v \in \gamma(e)\} \\ &\leq \sum_{v \in \gamma(e)} f(v,e) \cdot \chi(v), \end{aligned} \tag{3.19}$$

where the equality follows from the fact that $\sum_{v \in \gamma(e)} f(v,e) = 1$ by the definition of a node-supply function and the inequality follows from the fact that $\min\{\chi(v) : v \in \gamma(e)\} \leq \chi(v)$ for any $e \in E$. By summing over all $e \in E$ we obtain

$$\begin{aligned} \sum_{e \in E} \min\{\chi(v) : v \in \gamma(e)\} &\leq \sum_{e \in E} \sum_{v \in \gamma(e)} f(v,e) \cdot \chi(v) \\ &= \sum_{v \in V} \chi(v) \cdot \sum_{e : v \in \gamma(e)} f(v,e) \\ &\leq \sum_{v \in V} \chi(v) \cdot c, \end{aligned}$$

where the first inequality follows from Equation (3.19), the equality follows by rearranging terms, and the second inequality is due to the fact that the function f satisfies Equation (3.13). □

3.3.5 Bounding the weight of $E(\Pi)$ in H.

We are now ready to bound the weight of the edges of the 2-partition Π in the graph H derived from the input graph G by contracting the edges of the maximum weight matching M_1 and modifying the weight of the remaining edges according to Equation (3.1). Remember that H has no self-loops, has four parallel edges between any two vertices, and may have negative weight edges. The idea of the proof is to show that the edges of Π form a 4/3-

sparse graph after the *endpoint identification* of the edges of M_1 in G, and use Lemma 3.3.4 to bound the difference $\omega_G(E(\Pi)) - \omega_H(E(\Pi) \setminus M_1)$. The endpoint identification operation defined below is similar to the edge contraction, with the difference that the contracted edge is not removed but instead it becomes a loop in the new graph.

Definition 3.3.3. *(Endpoint Identification) Given a graph $G = (V, E, \gamma)$ the* endpoint identification *of an edge $e \in E$ results in graph $G'_e = (V'_e, E'_e, \gamma'_e)$ with*

- *vertex set $V'_e = (V(G) \setminus \gamma(e)) \cup \{v_e\}$ for some vertex $v_e \notin V(G)$,*
- *edge set $E'_e = E$ and*
- *γ'_e is defined as:*

$$\gamma'_e(f) = \begin{cases} \gamma_e(f) & \text{if } \gamma(f) \cap \gamma(e) = \emptyset, \\ (\gamma(f) \setminus \gamma(e)) \cup \{v_e\} & \text{otherwise.} \end{cases}$$

We say that v_e corresponds to e and vice versa. Observe that the edge e is a loop of v_e in G'_e.

Lemma 3.3.5. $\omega_H(E(\Pi) \setminus M_1) \geq \omega_G(E(\Pi)) - 4/3 \cdot \omega_G(M_1)$.

Proof. Let F_Π be the unweighted graph resulting from the input graph G after the endpoint identification of all edges in M_1, and after removing any edges that are not in $E(\Pi)$. Observe that all edges of $E(\Pi) \cap M_1$ are loops in F_Π and all edges of $E(\Pi) \setminus M_1$ are non-loops in F_Π.
Claim: F_Π is a 4/3-sparse graph.
Assume for contradiction that there exists a set of edges $S \subseteq E(F_\Pi)$ such that

$$|S| > 4/3 \cdot |V_F(S)|,$$

where $V_F(S)$ is the set of endpoints of the edges in S in the graph F_Π. Let $V_G(S)$ be the set of endpoints of the edges in S in the graph G. Since any vertex of F_Π corresponds to an edge in G, it holds that $|V_G(S)| \leq 2 \cdot |V_F(S)|$. Therefore, it holds that

$$|S| > 2/3 \cdot |V_G(S)|.$$

Let $S_1, ..., S_p$ be the set of independent components of S in G. Clearly, $|S| = |S_1| + ... + |S_p|$ and $|V_G(S)| = |V_G(S_1)| + ... + |V_G(S_p)|$. Therefore there exist at least one $S^* \in \{S_1, ..., S_p\}$ such that

$$|S^*| > 2/3 \cdot |V_G(S^*)|. \tag{3.20}$$

However, S^* is a subset of the edges of Π in G and therefore S^* is either a 2-path or a single edge. If S^* is a 2-path then $|S^*| = 2$ and $|V_G(S^*)| = 3$, otherwise $|S^*| = 1$ and $|V_G(S^*)| = 2$, thus contradicting Inequality (3.20) and proving the claim. ■

Let $\chi : V(F_\Pi) \to \mathbb{R}$ be a node weight function with $\chi(v_a) \equiv \omega_G(a)$ for all $v_a \in V(F_\Pi)$, where a is the edge of M_1 corresponding to the vertex v_a. Applying Lemma 3.3.4 to F_Π, it holds that

$$\sum_{e \in E} \min\{\chi(v) : v \text{ endpoint of } e \text{ in } F_\Pi\} \;\leq\; \frac{4}{3} \cdot \sum_{v \in V} \chi(v).$$

By the choice of χ we have $\sum_{v \in V} \chi(v) = \omega_G(M_1)$ and therefore from the inequation above we obtain:

$$\sum_{e \in E} \min\{\chi(v) : v \text{ endpoint of } e \text{ in } G\} \;\leq\; \frac{4}{3} \cdot \omega_G(M_1). \tag{3.21}$$

To simplify the following argument, let $\mu(e) \equiv \min\{\chi(v) : v \text{ endpoint of } e \text{ in } F_\Pi\}$ for any

$e \in F_\Pi$. Therefore Eq. (3.21) becomes:

$$\sum_{e \in E} \mu(e) \;\leq\; \frac{4}{3} \cdot \omega_G(M_1). \tag{3.22}$$

We split the left term of Ineq. (3.21) according to whether the edge e is a loop or not. As observed above, the loops in F_Π are exactly the edges of $E(\Pi) \cap M_1$, and therefore it holds that $\mu(e) = \omega_G(e)$ for all $e \in E(\Pi) \cap M_1$. Thus the following holds:

$$\sum_{e \in E(\Pi) \cap M_1} \mu(e) \;=\; \sum_{e \in E(\Pi) \cap M_1} \omega_G(e). \tag{3.23}$$

Let e be an edge of F_Π with endpoints v_a and v_b, corresponding to edges a and b of M_1 respectively. Remember that $\chi(v_a) = \omega_G(a)$ and $\chi(v_b) = \omega_G(b)$ by definition of χ and therefore $\mu(e) = \min\{\omega_G(a), \omega_G(b)\}$. The definition of ω_H (Eq. 3.1), can therefore be rewritten as

$$\omega_H(e) = \omega_G(e) - \mu(e).$$

By summing the above equation over all edges in $E(\Pi) \setminus M_1$, we obtain

$$\sum_{e \in E(\Pi) \setminus M_1} \omega_H(e) \;=\; \sum_{e \in E(\Pi) \setminus M_1} \omega_G(e) - \sum_{e \in E(\Pi) \setminus M_1} \mu(e)$$

or equivalently,

$$\sum_{e \in E(\Pi) \setminus M_1} \mu(e) \;=\; \sum_{e \in E(\Pi) \setminus M_1} \omega_G(e) - \sum_{e \in E(\Pi) \setminus M_1} \omega_H(e). \tag{3.24}$$

Adding the left and right sides of Eqs. (3.23) and (3.24) we obtain

$$\begin{aligned}\sum_{e\in E(\Pi)} \mu(e) &= \sum_{e\in E(\Pi)} \omega_G(e) - \sum_{e\in E(\Pi)\setminus M_1} \omega_H(e) \\ &= \omega_G(E(\Pi)) - \omega_H(E(\Pi)\setminus M_1).\end{aligned}$$

By using Eq. (3.22) to bound the left term it holds that

$$\frac{4}{3}\cdot\omega_G(M_1) \geq \omega_G(E(\Pi)) - \omega_H(E(\Pi)\setminus M_1),$$

and therefore

$$\omega_H(E(\Pi)\setminus M_1) \geq \omega_G(E(\Pi)) - \frac{4}{3}\cdot\omega_G(M_1),$$

which completes the proof. □

3.3.6 Bounding the weight of M_2 in H.

The main result of the previous section lower bounds the weight of the edges of $E(\Pi)\setminus M_1$ in H. In this section we use this bound to show that the weight of the matching M_2 of H is at least $\omega_H(E(\Pi)\setminus M_1)/4$. We do this in Lemma 3.3.8 by showing that there exist a matching in H using only the edges in $E(\Pi)\setminus M_1$ that satisfies the desired bound.

We prove this result in two steps. Lemma 3.3.7 shows that for any matching M there exists a subset A of k edges of $E(\Pi)\setminus M$ such that $A\cup M$ forms a collection of paths in H, A contains an edge from every 2-path of Π and that A has a "large enough" weight. Lemma 3.3.8 then shows that any such set of edges can be partitioned into two matchings of H, at least one of them satisfying the desired bound.

Definition 3.3.4. *For a set of edges X and a 2-path partition Π, let Π_X denote the set of 2-paths of Π that use at least one edge of X.*

Definition 3.3.5. *Let $X \subseteq E(G)$, M be matching of G, and ω be a weight function on $E(G)$. Then X is called* ω-dominant *with respect to* Π *and M if*

$$\omega(E(\Pi) \cap X) \geq \frac{1}{2} \cdot \omega(\Pi_X \setminus \Pi_M) + \omega(\Pi_X \cap \Pi_M). \tag{3.25}$$

Moreover, X is called strongly ω-dominant *w.r.t. Π and M if every connected component of X is ω-dominant w.r.t. Π and M.*

We simply refer to a set of edges X as ω-dominant when Π and M are implicit.

Lemma 3.3.6. *Let $X, Y \subseteq E(G)$ be two ω-dominant sets w.r.t. a 2-path partition Π and a matching M. If $X \cap Y = \emptyset$ then $X \cup Y$ is ω-dominant w.r.t. Π and M.*

Proof. Observe that since X and Y are disjoint

$$\omega(E(\Pi) \cap (X \cup Y)) = \omega(E(\Pi) \cap X) + \omega(E(\Pi) \cap Y)$$

and since both X and Y are ω-dominant it holds that

$$\begin{aligned}
\omega(E(\Pi) \cap (X \cup Y)) \geq\ & \frac{1}{2} \cdot (\omega(\Pi_X \setminus \Pi_M) + \omega(\Pi_Y \setminus \Pi_M)) \\
& + \omega(\Pi_X \cap \Pi_M) + \omega(\Pi_Y \cap \Pi_M) \\
\geq\ & \frac{1}{2} \cdot \omega((\Pi_X \cup \Pi_Y) \setminus \Pi_M) + \omega((\Pi_X \cup \Pi_Y) \cap \Pi_M) \\
=\ & \frac{1}{2} \cdot \omega(\Pi_{X \cup Y} \setminus \Pi_M) + \omega(\Pi_{X \cup Y} \cap \Pi_M),
\end{aligned}$$

where the first inequality follows from the definition of ω-dominance applied to both X and Y, the second inequality follows from the inclusion-exclusion principle and the equality is direct from Definition 3.3.4. □

Lemma 3.3.7. *Let Π be a 2-path partition of a graph G, M be an arbitrary matching of G and ω be a weight function on $E(G)$. Then there exists a matching A of G such that:*

(a) $|A| = k$,

(b) $A \subseteq (E(\Pi) \setminus M)$,

(c) $A \cup M$ *is strongly* ω*-dominant w.r.t.* Π *and* M*, and*

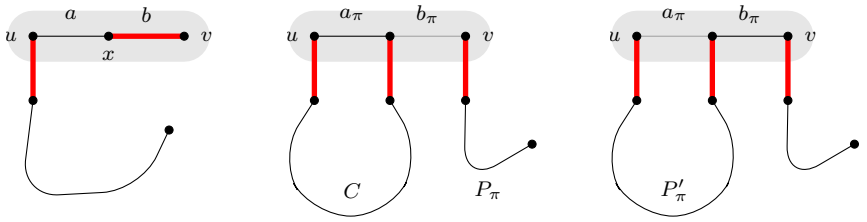
(d) $A \cup M$ *contains no cycles.*

Proof. We first show that there exists matching in G which satisfies conditions (a)-(c). For this, let A^* be the set of edges obtained by taking all the edges of $E(\Pi_M) \setminus M$, and the edge with higher weight from every 2-path in $\Pi \setminus \Pi_M$. Since the 2-paths of Π are vertex disjoint by definition, A^* is a valid matching. The matching A^* contains one edge from every 2-path of Π and thus the size of A^* is k, satisfying condition (a) of the lemma. Moreover, it is easy to see that A^* satisfies condition (b) of the lemma by construction.

Let $X \subseteq A^* \cup M$ be a connected component. Then for every $\pi \in \Pi_X \cap \Pi_M$ one edge of π is in M by definition of Π_M and the other edge of π is in $A^* \cap X$ by definition of A^* and therefore $\pi \subseteq X$. Moreover, for every $\pi \in \Pi_X \setminus \Pi_M$ the edge of π with higher weight is in X, and therefore X is ω-dominant. Thus, since any connected component $X \subseteq A^* \cup M$ is ω-dominant, it holds that $A^* \cup M$ is strongly ω-dominant w.r.t. Π and M, and A^* satisfies condition (c) of the lemma.

Among all matchings of G that satisfy conditions (a)-(c), let A be such that $A \cup M$ contains the fewest number of cycles and assume for contradiction that there exists a cycle $C \subseteq A \cup M$. Observe that since any vertex of G is incident to at most an edge of A and an edge of M, the set of edges $A \cup M$ is a collection of disjoint paths and cycles of G. We construct a new matching A' that satisfies conditions (a)-(c) and has one less cycle, thus reaching a contradiction.

Let Π_C and Π_M denote the set of 2-paths of Π having an edge in common with the cycle C and the matching M respectively, like in Definition 3.3.4. We first show that these two sets of paths are disjoint:



(a) Path. (b) Cycle. (c) Cycle modified to path.

Figure 3.3: Edges of the matching M are depicted in bold (red) and 2-paths of Π in light gray. (a) An example of a 2-path in Π containing an edge $a \in M$; the edge a can be incident to only one other edge $b \in M \cup A^*$ and therefore cannot be part of a cycle in $M \cup A^*$. (b) An example of a cycle C in $M \cup A^*$. (c) The path P' formed from the edges of cycle C when edge a^* is replaced by b^*.

Claim 1: $\Pi_C \cap \Pi_M = \emptyset$.

Assume for contradiction that $\pi \in \Pi_C \cap \Pi_M$ and let $\pi = \{a, b\}$, with $a \in E(\Pi) \setminus M$ and $b \in M$ (Fig. 3.3(a)). Moreover, let $\gamma(a) = \{u, x\}$ and $\gamma(b) = \{v, x\}$. Since vertex v is incident to b, v is not incident to any other edge in M. Moreover, since v is a vertex of π, v is not incident to any other edge in Π. Since $A \subset E(\Pi)$ it holds that v is not incident to any edge in A. Therefore v is the end of a path in $A \cup M$ which implies $\pi \notin \Pi_C$, contradicting the assumption. ■

The following three claims show that there are multiple ways to modify A to a matching that satisfies conditions (a) and (b) of the lemma and forms fewer cycles with the edges of M. Then we show that one of these modified matchings also satisfies condition (c), therefore contradicting the fact that A is the matching with fewest cycles satisfying conditions (a)-(c).

Let π be an arbitrary 2-path in Π_C. Observe that since A is a matching of size k and $A \subseteq E(\Pi)$, A contains an edge from every 2-path of Π. Let $\pi = \{a_\pi, b_\pi\}$ with $a_\pi \in A$ and $b_\pi \in \pi \setminus A$, and let $A_\pi = A \cup \{b_\pi\} \setminus \{a_\pi\}$.

Claim 2: A_π is a matching of G.

Since $A \subseteq E(\Pi)$ and the 2-paths of Π are vertex disjoint, edge b_π is adjacent only to edge

a_π in A. Therefore $A \cup \{b_\pi\} \setminus \{a_\pi\}$ is a matching of G. ■

Claim 3: $|A_\pi| = k$.

The cardinality of A_π is the same as the cardinality of A and $|A| = k$ by definition. ■

Claim 4: $A_\pi \subseteq (E(\Pi) \setminus M)$.

Clearly $A_\pi \subseteq E(\Pi)$ since by definition $A \subseteq E(\Pi)$ and $b_\pi \in E(\Pi)$. Since $A \cap M = \emptyset$ to prove the claim we just need to show that $b_\pi \notin M$. Observe that if $b_\pi \in M$ it holds that $\pi \in \Pi_M$ and since by definition $\pi \in \Pi_C$, it holds that $\pi \in \Pi_M \cap \Pi_C$ which contradicts Claim 1. ■

Claim 5: $A_\pi \cup M$ has one less cycle than $A \cup M$.

Let v be the vertex incident to b_π but not incident to a_π like in Fig. 3.3(b). Since $b_\pi \notin A$ and $A \subseteq E(\Pi)$, the vertex v is not being matched by A and therefore v has degree at most 1 in $A \cup M$ and thus is a path-end for some (possibly trivial) path P_π in $A \cup M$. Let $P'_\pi = C \cup P_\pi \cup \{b_\pi\} \setminus \{a_\pi\}$ (Fig. 3.3(c)) and observe that P'_π is a connected component of $A_\pi \cup M$.

Let u be the vertex incident to a_π but not incident to b_π. Observe that u is a vertex of P'_π and has degree 1, implying P'_π is not a cycle. Since A_π is a matching, all connected components of $A_\pi \cup M$ are either paths or cycles and therefore P'_π is a path. Since all other connected components of $M \cup A_\pi$ are also connected components of $M \cup A$, it holds that $M \cup A_\pi$ has one less cycle than $M \cup A$. ■

To prove the lemma it remains to show that there exists $\pi^* \in \Pi_C$ such that A_{π^*} satisfies condition (c) of the lemma, i.e. that $A^* \cup M$ is strongly ω-dominant w.r.t. Π and M. Let $\pi^* = \{a^*, b^*\}$ be a 2-path in Π_C such that

$$\omega(a_{\pi^*}) - \omega(b_{\pi^*}) \leq \omega(a_\pi) - \omega(b_\pi), \quad \forall\ \pi \in \Pi_C. \tag{3.26}$$

Claim 6: A_{π^*} is strongly ω-dominant w.r.t. Π and M.

Let $P'_{\pi^*} = C \cup P_{\pi^*} \cup \{b^*_\pi\} \setminus \{a^*_\pi\}$. In the proof for Claim 5 we showed that P'_{π^*} is a path and therefore P'_{π^*} is a connected component. Since all other connected components of $A_{\pi^*} \cup M$ are also connected components of $A \cup M$, to prove the claim it is enough to show that P'_{π^*} is ω-dominant.

Let $C' = C \cup \{b^*_\pi\} \setminus \{a^*_\pi\}$ and observe that since C' and P_{π^*} are disjoint, to show that P'_{π^*} is ω-dominant by Lemma 3.3.6 it is enough to show that both C' and P_{π^*} are ω-dominant. Since P_{π^*} is a connected component of $A \cup M$ which is strongly ω-dominant, P_{π^*} is ω-dominant. It remains to show that C' is ω-dominant. Observe that

$$\begin{aligned}
\omega\left(E(\Pi) \cap C\right) &= \sum_{\pi \in \Pi_C} \omega(a_\pi) \\
&= \frac{1}{2}\left(\sum_{\pi \in \Pi_C} (\omega(a_\pi) + \omega(b_\pi)) + \sum_{\pi \in \Pi_C} (\omega(a_\pi) - \omega(b_\pi))\right) \\
&\geq \frac{1}{2}\left(\sum_{\pi \in \Pi_C} (\omega(a_\pi) + \omega(b_\pi)) + \sum_{\pi \in \Pi_C} (\omega(a_{\pi^*}) - \omega(b_{\pi^*}))\right) \\
&\geq \frac{1}{2}\left(\sum_{\pi \in \Pi_C} (\omega(a_\pi) + \omega(b_\pi))\right) + \omega(a_{\pi^*}) - \omega(b_{\pi^*}) \\
&= \frac{1}{2}\omega(\Pi_C) + \omega(a_{\pi^*}) - \omega(b_{\pi^*}), \qquad (3.27)
\end{aligned}$$

where the first equality is by definition of a_π and the second equality is by rearranging terms. The first inequality follows from the choice of $\pi^* = \{a_{\pi^*}, b_{\pi^*}\}$ as the 2-path that minimizes $\omega(a_\pi) - \omega(b_\pi)$ over all $\pi \in \Pi_C$ (Ineq. (3.26)) and the second inequality is due to the fact that there are at least two edges of A in a cycle of $A \cup M$, and therefore $|\Pi_C| \geq 2$. By moving terms, Ineq. (3.27) can be rewritten as

$$\omega(E(\Pi) \cap C) - \omega(a_{\pi^*}) + \omega(b_{\pi^*}) \;\geq\; \frac{1}{2}\omega(\Pi_C). \qquad (3.28)$$

Since by definition $a_{\pi^*} \in E(\Pi) \cap C$ and $b_{\pi^*} \notin E(\Pi) \cap C$, it holds that

$$\omega(E(\Pi) \cap C') \;=\; \omega(E(\Pi) \cap C) - \omega(a_{\pi^*}) + \omega(b_{\pi^*})$$

and therefore Ineq. (3.28) implies

$$\omega(E(\Pi) \cap C') \;\geq\; \frac{1}{2}\omega(\Pi_C). \tag{3.29}$$

Since $\Pi_C = \Pi'_C$ and by Claim 1 it holds that $\Pi_C \cap \Pi_M = \emptyset$, Ineq. (3.29) is equivalent to

$$\omega(E(\Pi) \cap C') \;\geq\; \frac{1}{2}\omega(\Pi_{C'} \setminus \Pi_M) + \omega(\Pi_{C'} \cap \Pi_M) \tag{3.30}$$

and therefore C' is ω-dominant, thus proving the claim. ■

Putting it all together, we first showed that there exists a matching satisfying conditions (a)-(c) of the lemma and we assumed that A is a matching of G satisfying conditions (a)-(c) such that the number of cycles in $A \cup M$ is minimized. Assuming that there exists a cycle C in $A \cup M$ we constructed a set of edges A_{π^*} that is a matching by Claim 2, satisfies conditions (a) by Claim 3, (b) by Claim 4 and (c) by Claim 6, and for which $A_{\pi^*} \cup M$ has one less cycle than $A \cup M$ (Claim 5), thus reaching a contradiction. □

We are now ready to prove the main result of this section which lower bounds the weight of the matching M_2 in graph H. To do so we show that there exists a matching of H satisfying the desired bound and uses only edges of Π.

Lemma 3.3.8. *Let Π and M be a 2-path partition and a matching of a simple complete graph G respectively. If $H = G/M$ then there exists a matching A^* of H with $A^* \subseteq E(\Pi) \setminus M$, $|A^*| = k/2$ and $\omega_H(A^*) \geq \omega_H(E(\Pi) \setminus M)/4$.*

Proof. Let ω be an edge weight function on $E(G)$ with $\omega(e) = 0$ for any $e \in M$ and

$\omega(e) = \omega_H(e)$ for any edge $e \in E(G) \setminus M$. Let $A \subseteq E(\Pi) \setminus M$ be a matching of size k such that $A \cup M$ is ω-dominant w.r.t. Π and M and $A \cup M$ is a collection of disjoint paths, as guaranteed to exist by Lemma 3.3.7.

Since A is a matching in G and $A \subseteq E(\Pi)$, every 2-path of Π has an edge in A and therefore $\Pi_A = \Pi$, thus also implying $\Pi_{A \cup M} = \Pi$. Then, from the ω-dominance of $A \cup M$ it holds that

$$\omega(E(\Pi) \cap (A \cup M)) \;\geq\; \frac{1}{2} \cdot \omega(\Pi \setminus \Pi_M) + \omega(\Pi \cap \Pi_M). \tag{3.31}$$

Since A and M are disjoint, Ineq. (3.31) implies

$$\omega(E(\Pi) \cap A) + \omega(E(\Pi) \cap M)) \;\geq\; \frac{1}{2} \cdot \omega(\Pi \setminus \Pi_M) + \omega(\Pi_M),$$

and therefore

$$\begin{aligned}
\omega(E(\Pi) \cap A) \;&\geq\; \frac{1}{2} \cdot \omega(\Pi \setminus \Pi_M) + \omega(\Pi_M) - \omega(E(\Pi) \cap M)) \\
&=\; \frac{1}{2} \cdot \omega(\Pi \setminus \Pi_M) + \omega(\Pi_M) - \omega(E(\Pi_M) \cap M)) \\
&=\; \frac{1}{2} \cdot \omega(\Pi \setminus \Pi_M) + \omega(E(\Pi_M) \setminus M) \\
&\geq\; \frac{1}{2} \cdot \omega(E(\Pi) \setminus M),
\end{aligned}$$

where the first equality is direct from the definition of Π_M as the set of 2-paths of Π intersecting M.

Since $A \cup M$ forms a collection of disjoint paths in G, the edges of A also form a collection of disjoint paths in $H = G/M$. We partition these edges into two matchings A_1 and A_2 of H, each with $k/2$ edges. One of those matchings has at least half the weight of A, thus proving the lemma. □

3.3.7 Graphs with odd number of vertices.

Remember that since the **WeightedDoubleMatching** algorithm presented in Fig. 3.1 assumes a perfect matching exists for the input graph, it must take as input a graph of size $3k$ for some even k. In this section we show how to modify the algorithm to deal with graphs of odd size. The high level idea is to modify the input graph by adding three additional dummy vertices that are incident only to edges of weight 0 and to choose matchings M_1 and M_2 with some additional properties in such a way that the resulting 2-path partition has a 2-path containing the three dummy vertices. By discarding this 2-path the algorithm outputs a partition of the original graph. The challenge is to show that the new properties enforced on the matchings M_1 and M_2 don't have a significant effect on the weight of the resulting 2-path partition.

Let G' be the graph formed from G by adding three dummy vertices d_1, d_2 and d_3 incident only to edges of weight 0. Let e_1 be the edge of G' with endpoints d_1 and d_2 and let e_2 be the edge of G' with endpoints d_2 and d_3. Let OPT be a fixed optimal 2-path partition of G and let π_d be the 2-path $\{e_1, e_2\}$. Moreover, let $\Pi = OPT \cup \{\pi_d\}$ be a 2-path partition of G', not necessarily optimal. Clearly, the weight of OPT in G equals the weight of Π in G'. We run the WeightedDoubleMatching algorithm on graph G' with some minor modifications, as explained below.

We modify Step 1 of the algorithm to choose a maximum weight perfect matching M_1 with the property that $e_1 \in M_1$. It is easy to see that such a perfect maximum weight matching of G' exists and can be computed efficiently.

Let e_3 be the edge of M_1 matching vertex d_3 and let v_{e_1} and v_{e_3} be the vertices of $H \equiv G'/M_1$ corresponding to edges e_1 and e_3 of G' respectively. We modify Step 4 of the WeightedDoubleMatching algorithm to set M_2 to a matching of maximum weight in H among those matchings M having the following properties:

- $|M| = (k+1)/2$, and
- M matches vertex v_{e_1} to vertex v_{e_3} or M does not match v_{e_1}.

Such a matching can be computed efficiently by removing all edges incident to v_{e_1} in H, except for edge e_2.

Observe that M_2 is may not be a maximum weight matching of size $(k+1)/2$ in H. However, Lemma 3.3.8 guarantees that there exists a matching of weight $\omega_H(A^*) \geq \omega_H(E(\Pi) \setminus M_1)/4$ using only edges of Π. Since e_2 is the only edge of Π incident to v_{e_1} in H it holds that

$$\omega_H(M_2) \geq \omega_H(E(\Pi) \setminus M_1)/4. \tag{3.32}$$

Next, we modify Step 6 of the algorithm to guarantee that edge e_1 is part of a 2-path in the output of the algorithm. If v_{e_1} is not matched by M_2 then the original algorithm already guarantees that since $v_{e_1} \in Q$ and therefore e_1 belongs to a 2-path in $\mathcal{A}_2$. Otherwise, matching M_2 matches v_{e_1} to v_{e_3}. Since by definition both e_1 and e_3 have weight 0 in G', by breaking ties appropriately we can insure that π_d, the 2-path formed by the three dummy vertices, is a path in $\mathcal{A}_1$.

Observe that the modifications to Step 1 and Step 6 of the algorithm only specify how to break ties and therefore do not alter the properties of the algorithm. The change in Step 4 may result in a suboptimal choice for M_2 but as explained above, Ineq. 3.32 lower bounding the weight of M_2 in H still holds. Therefore, after these modifications of the algorithm $\mathcal{A}$ is a 2-path partition of G' of weight at least $3/4 \cdot \omega_G(\Pi)$. Moreover, vertices d_1 and d_2 belonging to some 2-path $\pi_{1,2} \in \mathcal{A}$.

Let $u_3 \in V(G')$ be the vertex in $V(\pi_{1,2}) \setminus \{d_1, d_2\}$ and let π_3 be the 2-path of $\mathcal{A}$ containing vertex d_3, with $u_1, u_2 \in V(\pi_3) \setminus \{d_3\}$. Finally, let π_u be the 2-path of maximum weight formed from vertices $\{u_1, u_2, u_3\}$ and output $\mathcal{A}' = \mathcal{A} \cup \{\pi_u\} \setminus \{\pi_{1,2}, \pi_3\}$. Clearly $\mathcal{A}'$ is a

2-path partition of G and since all edges incident to the dummy vertices have weight 0 it holds that $\omega_G(\mathcal{A}') = \omega_{G'}(\mathcal{A})$, thus proving Theorem 1 for odd values of k.

3.4 M2PP on $\{0,1\}$-edge-weighted Graphs

In this section we describe and analyze the **DoubleMatching** approximation algorithm for the Maximum Weight 2-Path Partition problem (M2PP) in graphs in which the weights of the edges are either 0 or 1. The main difference to the algorithm described in Section 3.3 is that the first matching M_1 only uses edges of weight 1 and may not be perfect. In the graph H resulting from contracting the edges of M_1 we call the vertices corresponding to edges of M_1 *new* vertices, and the vertices from the original graph *old* vertices. The second matching M_2 aims to match as many old vertices with new vertices as possible, and have just as many new-new edges as not matched new vertices. A formal description of the algorithm is presented in Fig. 3.4.

3.4.1 Examples.

Fig. 3.5 illustrates an example for the case where the matching M_1 has at most k edges. Fig. 3.5(a) shows a complete $\{0,1\}$-edge-weighted graph on 12 vertices ($k = 4$) and matching number 3, with its zero weight edges omitted. A maximum weight matching, $M_1 = \{a, b, c\}$, is shown in bold (red). Fig. 3.5(b) shows the graph H obtained from contracting the edges of M_1, with the matching M_2 shown in bold (red). Vertices v_a, v_b and v_c correspond to the edges a, b and c respectively. Fig. 3.5 (c) shows the resulting 2-paths with zero weight edges drawn as dotted segments. The algorithm outputs the sets $\mathcal{A}_1 = \{\pi_1, \pi_1', \pi_1''\}$ and $\mathcal{A}_4 = \{\pi_4\}$. Note that when G has a small matching number ($\leq k$), both $\mathcal{A}_2$ and $\mathcal{A}_3$ are empty since all new vertices are matched by M_2 to old vertices.

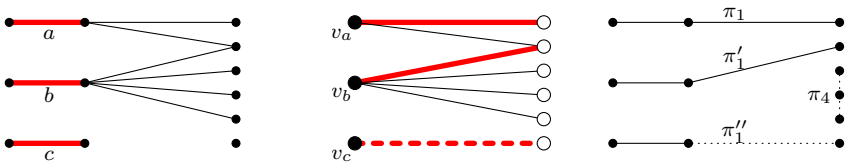
Fig. 3.6(a) shows a complete $\{0,1\}$-edge-weighted graph G on 12 vertices with its zero

Input: complete $\{0,1\}$-edge-weighted graph $G = (V, E, \omega_G, \gamma_G)$.

1. $M_1 \leftarrow$ maximum weight matching of G with no weight 0 edges
2. $H \leftarrow G/M_1$; let $V_{old} = V(H) \cap V(G)$ and $V_{new} = V(H) \setminus V(G)$.
   We call an edge $e \in E(H)$:
   - a *new-new* edge if $\gamma(e) \subseteq V_{new}$,
   - an *old-old* edge if $\gamma(e) \subseteq V_{old}$,
   - an *old-new* edge otherwise.
3. $M_2 \leftarrow$ maximum weight matching of H (based on ω_H) satisfying the following constraints:
   a. M_2 has $\min\{|V_{new}|, |V_{old}|\}$ old-new edges.
   b. if $|V_{new}| > |V_{old}|$, then M_2 has $(|V_{new}| - |V_{old}|)/3$ new-new edges.
   c. M_2 doesn't have any old-old edges.
4. Call any old vertex not matched by M_2 a residual vertex and denote by R the set of residual vertices in H.
5. Create four sets of 2-paths $\mathcal{A}_1, ..., \mathcal{A}_4$ as follows:
   - $\mathcal{A}_1$: one 2-path for each old-new edge $e \in M_2$. Let v_a be the new endpoint of e in H and let $a \in E(G)$ be the edge corresponding to v_a. Create a 2-path from edges a and e.
   - $\mathcal{A}_2$: one 2-path for each new-new edge $e \in M_2$. Let v_a and v_b be the endpoints of e in H and let $a, b \in E(G)$ be the edges corresponding to v_a and v_b respectively. Assume that $\omega_G(a) \geq \omega_G(b)$. Create a 2-path from edges a and e and add the unused vertex of b to the residual vertex set R.
   - $\mathcal{A}_3$: one 2-path for each new vertex v_a of H not matched by M_2. Let $a \in E(G)$ be the edge corresponding to v_a. Create a 2-path from the edge a and an arbitrary residual vertex in R.
   - $\mathcal{A}_4$: create 2-paths arbitrarily from the unused residual vertices of R.

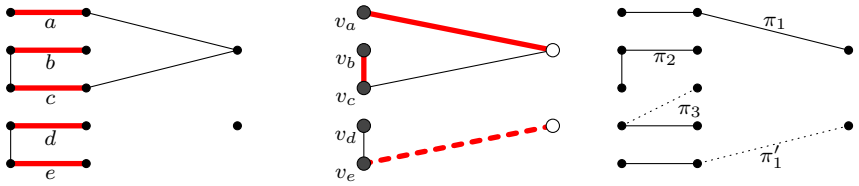
Return: $\mathcal{A} \leftarrow \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$.

Figure 3.4: The **DoubleMatching** algorithm for $\{0,1\}$-edge-weighted graphs



(a) Matching M_1 (b) H and matching M_2 (c) Output 2-paths

Figure 3.5: **DoubleMatching** on a graph with small matching number.



(a) Matching M_1 (b) H and matching M_2 (c) Output 2-paths

Figure 3.6: **DoubleMatching** on a graph with large matching number.

weight edges omitted. A maximum matching $M_1 = \{a, b, c, d, e\}$ is shown in bold (red). Fig. 3.6(b) shows the graph H obtained from G after the edges of M_1 are contracted. The vertices v_a-v_e correspond to edges a-e respectively. The matching M_2 is shown in bold (red). Fig. 3.6(c) shows the resulting 2-paths with zero weight edges drawn as dotted segments. The algorithm produces the sets $\mathcal{A}_1 = \{\pi_1, \pi_1'\}$, $\mathcal{A}_2 = \{\pi_2\}$ and $\mathcal{A}_3 = \{\pi_3\}$. Note that when G has a large matching number ($\geq k$), the set $\mathcal{A}_4$ is empty because all old vertices are matched in M_2 and the number of new-new edges in M_2 equals the number of new vertices not matched my M_2 and therefore all residual vertices are used in $\mathcal{A}_3$.

3.4.2 Algorithm Analysis.

The main result of this section is the following theorem which is a corollary of Proposition 3.4.6 and Proposition 3.4.10.

Theorem 3.1.2. DoubleMatching *is a* 3/4*-approximation algorithm for the* M2PP *on* $\{0, 1\}$*-edge-weighted graphs and runs in time* $O(n^{2.5})$.

To intuitively understand why the DoubleMatching algorithm performs better than WeightedDoubleMatching, consider it's different characteristics depending on whether $|V_{new}|$ is larger than $|V_{old}|$ or not (or equivalently, on whether matching M_1 is larger than k or not). When the size of M_1 is smaller than k, it holds that $|V_{old}|$ is smaller than $|V_{new}|$, and thus all of the edges of the matching M_1 are used in the output solution. This contrasts with WeightedDoubleMatching where only 2/3 of the edges of M_1 are used in the final 2-path partition. Conversely, when M_1 is larger than k, we can assert that $\omega(M_1)$ is up to 3/4 of the weight of the optimal solution. This contrasts to the weighted case where $\omega(M_1)$ may be only half the weight of the optimal solution.

For the rest of this section let Π be the set of 2-paths of a fixed solution. Moreover, the following notations are used throughout the remainder of this section.

Definition 3.4.1. *We use the following notations:*

- Z*: the number of edges of weight* 0 *used by* Π*;*
- $\mathcal{O}_1 \subseteq V_{old}$*: the set of old vertices incident to at least one edge of* Π *of weight* 1*;*
- $\mathcal{O}_0 = V_{old} \setminus \mathcal{O}_1$.

We begin our analysis with a few simple claims that are being used in the later sections. The first observation formalizes the intuition that the smaller the size of the matching M_1, the fewer of its edges we have to break in Step 5 of the algorithm. The second observation emphasizes the fact that no value is lost from not matching old vertices to each other. Finally, the lemma shows that only a limited number of edges of weight 1 exist between old and new vertices. When combined, the three claims can be used to argue that the weight of the algorithm solution is relatively large even when the cardinality of the first matching is small.

Observation 3.4.1. $|V_{old}| = 3k - 2|M_1|$.

Observation 3.4.2. $\omega(e) = 0$*, for any old-old edge* $e \in E(H)$.

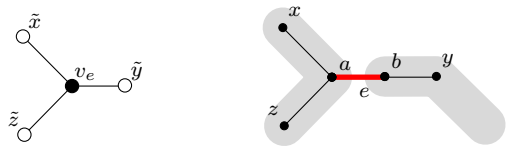


Figure 3.7: Illustration of the proof of Lemma 3.4.3. (a) The new vertex v_e in H having 3 new neighbors connected to by edges of weight 1 in $E(\Pi)$. (b) The edge e with endpoints $\{a, b\}$ in the matching M_1, the 2-paths of OPT using vertices x, y and z in light gray, and the alternating path (x, a, b, y) in G.

Lemma 3.4.3. *Every new vertex has at most two incident old-new edges of weight* 1 *in* $E(\Pi)$.

Proof. Assume that a new vertex $v_e \in V_{new}$ has three incident old-new edges of weight 1 in $E(\Pi)$. Let $\gamma(e) = \{a, b\}$ and let $x, y, z \in E(\Pi)$ be the three old-new edges of weight 1 incident to v_e. Observe that, since all the edges belong to the 2-partition Π, at most two of the three edges are incident to either a or b in G as shown in Fig. 3.7. W.l.o.g., let x be incident to a and let y be incident to b. Remember that by definition an old-new edge is incident to exactly one old vertex and one new vertex and therefore only one of its endpoints in G is matched by M_1. Therefore, the set of edges $M_1 \cup \{x,\ y\} \setminus \{e\}$ is a matching in G of larger weight, contradicting the maximality of M_1. □

Lemma 3.4.4. $Z \geq \frac{2}{3}|\mathcal{O}_0|$.

Proof. As $\mathcal{O}_0 \subseteq V(G)$, each of its vertices is covered by a 2-path of Π. For $j = 1, 2, 3$, let ℓ_j denote the number of 2-paths in Π containing exactly j vertices of $\mathcal{O}_0$. Note that $|\mathcal{O}_0| = 3\ell_3 + 2\ell_2 + \ell_1$.

Let π be a 2-path of Π. If π contains one vertex of $\mathcal{O}_0$, then π contains at least an edge of weight 0 by definition of $\mathcal{O}_0$. Moreover, if two or three vertices of π are in $\mathcal{O}_0$, then both edges of π have weight 0. By summing the number of edges of weight 0 over all 2-paths in Π it follows that $Z \geq 2\ell_3 + 2\ell_2 + \ell_1$, and therefore $Z \geq \frac{2}{3}(3\ell_3 + 2\ell_2 + \ell_1) = \frac{2}{3}|\mathcal{O}_0|$. □

3.4.3 Small matching number.

In this section we show that the algorithm provides a good approximation when the matching number of the graph is less than k. The following lemma bounds the weight of the second matching M_2 in this case.

Lemma 3.4.5. *If* $|M_1| \leq k$, *then* $\omega(M_2) \geq |\mathcal{O}_1|/2$.

Proof. To prove the lemma we show that there exist a matching of weight at least $|\mathcal{O}_1|/2$ in a restricted subgraph of H. Let H_Π be the subgraph of H using only the old-new edges of Π of weight 1. Observe that H_Π is a bipartite graph and let X be a matching of maximum weight in H_Π.

Assume for contradiction that $\omega(X) < |\mathcal{O}_1|/2$. Let N_X be the set of new vertices matched by X and let O_X be the neighborhood of N_X in the graph H_Π. We first show that for any $v \in \mathcal{O}_1 \setminus O_X$ there exist an edge $e \in E(H_\Pi)$ incident to v such that $\{e\} \cup X$ is a matching of H_Π of higher weight than X. To see this, we first show that an there exists an edge incident to v in H_Π and then show that any such edge is not incident to any other edge of X.

By definition of $\mathcal{O}_1$ (Def. 3.4.1), for any vertex $v \in \mathcal{O}_1$ there exists an edge $e \in E(\Pi)$ incident to v such that $\omega(e) = 1$. Since by Obs. 3.4.2 all old-old edges of H have weight zero, e is an old-new edge of H and therefore $e \in E(H_\Pi)$. To see that such an edge e is not incident to any other edges of X remember that by definition of O_X it holds that v is not in the neighborhood of N_X.

It remains to show that the set $\mathcal{O}_1 \setminus O_X$ is not empty. To see this observe that since by assumption $\omega(X) < |\mathcal{O}_1|/2$, it also holds that $|N_X| < |\mathcal{O}_1|/2$. Moreover, since by Lemma 3.4.3 a new vertex can be incident to at most 2 old-new edges of weight 1 in H, it holds that $|W_X| < |\mathcal{O}_1|$. Therefore, $|\mathcal{O}_1| < |O_X|$ implying that $\mathcal{O}_1 \setminus O_X$ is not empty and thus completing the proof. □

Using the above lemma we are ready to prove the approximation guarantee for graphs

with small matching number.

Proposition 3.4.6. *When $|M_1| \leq k$ the DoubleMatching algorithm is a 7/12-approximation for the* M2PP *problem in $\{0,1\}$-graphs.*

Proof. Observe that since the size of M_1 is smaller than k, the number of old vertices is larger than the number of new vertices in H and therefore M_2 has exactly $|V_{new}|$ old-new edges and therefore, M_2 matches all the new vertices with old vertices. Thus, the sets of 2-paths $\mathcal{A}_2$ and $\mathcal{A}_3$ are empty since every element of these sets corresponds to a new-new edge of M_2 or an edge not matched by M_2 respectively.

The weight of $\mathcal{A}_1$ equals the sum of the weights of the edges of the matching M_1 and the weights of the matching M_2 and therefore the lemma follows:

$$\begin{aligned}
\omega(\mathcal{A}) &\geq \omega(M_1) + \omega(M_2) \\
&\geq \omega(M_1) + \frac{1}{2}|\mathcal{O}_1| \\
&= \frac{1}{2}(3k - |\mathcal{O}_0|) \\
&\geq \frac{3}{4}(2k - Z) \\
&= \frac{3}{4}\omega(\Pi).
\end{aligned}$$

The second inequality follows from Lemma 3.4.5; the first equality follows from Obs. 3.4.1 (stating that $2|M_1|+|V_{old}|= 3k$) and the fact that $|\mathcal{O}_0|+|\mathcal{O}_1|= |V_{old}|$ since $\mathcal{O}_0$ and $\mathcal{O}_1$ partition V_{old}; the third inequality follows from Lemma 3.4.4 (stating that $Z \geq \frac{2}{3}|\mathcal{O}_0|$); and the last equality follows from the definition of Z (denoting the number of zero weight edges in Π). □

3.4.4 Large matching number.

The case where the original graph has a matching number larger than or equal to k is more involved. In this case the number of old vertices is less than the number of new vertices

and the second matching has exactly V_{old} old-new edges. Lemma 3.4.7 below shows that satisfying part of constraint on M_2 is not detrimental to its weight. As before, we only consider matchings of H whose edges are in $E(\Pi)$.

Lemma 3.4.7. *If $|M_1|> k$ then for any matching X of H with $X \subseteq E(\Pi)$ and $\omega(e) = 1$ for all $e \in X$, there exists a matching X^* of H such that:*

(1) X^* contains at least $|\mathcal{O}_1|/2$ old-new edges

(2) $X^* \subseteq E(\Pi)$, and

(3) $\omega(X^*) = \omega(X)$.

Proof. Observe that X satisfies conditions (2) and (3) of the lemma and let X^* be a matching of H with the largest number of old-new edges among those matchings of H for which conditions (2) and (3) of the lemma hold. Assume for contradiction that condition (1) is violated, namely, X^* contains $\alpha < |\mathcal{O}_1|/2$ old-new edges and contains at least one new-new edge.

Let N_{X^*}, $|N_{X^*}|= \alpha$, be the set of new vertices that are endpoints of an old-new edge in X^*. Let O_{X^*} be the set of old vertices that are connected to a vertex in the set N_{X^*} by an edge of weight 1 in $E(\Pi)$:

$$O_{X^*} = \{v \in V_{old} : \exists e \in E(\Pi) \ s.t. \ v \in \gamma(e), \ \omega(e) = 1, \ \gamma(e) \cap N_{X^*} \neq \emptyset\}.$$

Since by Lemma 3.4.3 a new vertex cannot be incident to more than 2 old-new edges of weight 1 in $E(\Pi)$ it follows that $|O_{X^*}|\leq 2|N_{X^*}|= 2\alpha < |\mathcal{O}_1|$ and therefore $\mathcal{O}_1 \setminus O_{X^*}$ is not empty.

Let u be an old vertex in $\mathcal{O}_1 \setminus O_{X^*}$ and $e \in E(\Pi)$ be an old-new edge of weight 1 such that $u \in \gamma(e)$. Observe that such an edge exists since u is a vertex in $\mathcal{O}_1$ and moreover

observe that e is not incident to an old-new edge of X^* since otherwise u would be a vertex in O_{X^*}.

If e is incident to some new-new edge of X^*, let e' be that edge; otherwise let e' be an arbitrary new-new edge of X^*. The set $X^* \cup \{e\} \setminus \{e'\}$ is a matching satisfying conditions (2) and (4) of the lemma and having more old-new edges than X^*, and therefore contradicting the choice of X^*. □

The following lemma is the equivalent of Lemma 3.3.8 in the weighted case.

Lemma 3.4.8. *The graph $H = G/M_1$ has a matching $A^* \subseteq E(\Pi)$ with weight $(k - Z)/2$.*

Proof. Let $A \subseteq E(\Pi)$ be a matching of G, with $|A| = k$, such that $A \cup M_1$ is a collection of paths. Remember that Lemma 3.3.7 in Section 3.3.6 shows that such a matching exists. Because $A \subseteq E(\Pi)$ and Π has exactly Z edges of weight 0, it holds that $\omega(A) \geq k - Z$. Moreover, since $A \cup M_1$ is a collection of paths in G, it holds that A is a collection of paths in the graph H. By partitioning the edges of A into two matchings A_1 and A_2 of H, we obtain at least one matching with at least half the weight of A, thus proving the lemma. □

The following lemma bounds the weight of the second matching M_2 in the case when the size of the first matching is large. The idea is to start from a matching of weight $(k - Z)/2$ as guaranteed to exist by Lemma 3.4.8 and transform it into a matching of the same weight that uses some old-new edges by using Lemma 3.4.7. Then, we show that the resulting matching can be completed to a matching with $|V_{old}|$ number of old-new edges.

Lemma 3.4.9. *If $|M_1| > k$, then $\omega(M_2) \geq (2k - 3Z)/4$.*

Proof. Let $A \subseteq E(\Pi)$ be a matching of H with $\omega(A) \geq (k - Z)/2$ as guaranteed to exist by Lemma 3.4.8. Let X be a set of $(2k - 3Z)/4$ edges of weight 1 from A. Then, by Lemma 3.4.7, there exist a matching X^* of cardinality and weight equal to $(2k - 3Z)/4$ that either has at least $|\mathcal{O}_1|/2$ old-new edges, or has no new-new edges.

We show that X^* can be completed to a matching of H that satisfies the requirements of the algorithm, namely a matching that has exactly $|V_{old}|$ old-new edges (step 3.a. of the algorithm) and contains exactly $(|V_{new}|-|V_{old}|)/3$ new-new edges (step 3.b. of the algorithm). For this it is enough to show that X^* does not have too many edges of any type.

Since X^* has only edges of weight 1 and all old-old edges have weight 0, no old-old edges are in X^*. Moreover, observe that no matching can have more than $|V_{old}|$ old-new edges. Therefore, to complete the proof it is enough to show that X^* has at most $(|V_{new}|-|V_{old}|)/3$ new-new edges. Since the number of old-new edges in X^* is at least $|\mathcal{O}_1|/2$, we prove this by bounding the size of the set X^*:

$$\begin{aligned} |X^*| &= \frac{2k-3Z}{4} \\ &= \frac{3k-2|M_1|}{2} - \frac{3Z}{4} + |M_1|-k \\ &\leq \frac{|V_{old}|}{2} - \frac{|\mathcal{O}_0|}{2} + |M_1|-k \\ &= \frac{|\mathcal{O}_1|}{2} + |M_1|-k. \end{aligned} \tag{3.33}$$

The first equality follows from the definition of X^*, the second equality follows by rearranging terms, the first inequality is due to the fact that $|V_{old}|= 3k-2|M_1|$ and that according to Lemma 3.4.4 it holds that $Z \geq \frac{2}{3}|\mathcal{O}_0|$; the last equality follows from the fact that $\mathcal{O}_0$ and $\mathcal{O}_1$ partition V_{old}.

Since X^* has at least $|\mathcal{O}_1|/2$ old-new edges by definition, Ineq. (3.33) implies that $|X^*|$ has at most $|M_1|-k$ new-new edges and therefore it remains to show that $|M_1|-k \leq (|V_{new}|-|V_{old}|)/3$. It holds that

$$\begin{aligned} \frac{|V_{new}|-|V_{old}|}{3} &= \frac{|M_1|-(3k-2\cdot|M_1|)}{3} \\ &= |M_1|-k, \end{aligned}$$

where the first equality is due to the fact that $|V_{new}|= |M_1|$ from the definition of V_{new} and the fact that $|V_{old}|= 3k - 2 \cdot |M_1|$ by Obs. 3.4.1. Thus we proved that X^* does not have more than $(|V_{new}|-|V_{old}|)/3$ new-new edges.

Since X^* has weight $(2k-3Z)/4$ and does not have too many edges of any type, arbitrary edges of each type can be added to complete X^* to a matching of H of weight at least $(2k - 3Z)/4$ satisfying conditions 3.a and 3.b of the DoubleMatching algorithm, namely a matching that has exactly $|V_{old}|$ old-new edges and exactly $(|V_{new}|-|V_{old}|)/3$ new-new edges, thus completing the proof. □

Proposition 3.4.10. *When $|M_1|> k$ the* DoubleMatching *algorithm is a $7/12$-approximation for the* M2PP *problem in $\{0, 1\}$-graphs.*

Proof. Recall that $\omega(\Pi) = 2k - Z$. Since by construction M_2 has $(|V_{new}|-|V_{old}|)/3$ many new-new edges, and there is a 2-path in $\mathcal{A}_2$ for every new-new edge in M_2, it holds that

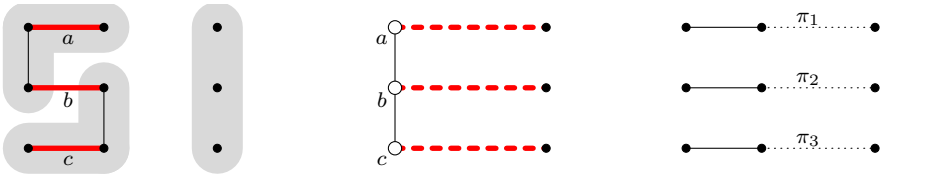
$$|\mathcal{A}_2|= \frac{|V_{new}|-|V_{old}|}{3} = \frac{|M_1|-(3k - 2|M_1|)}{3} = |M_1|-k,$$

where the second equality holds by the fact that $|M_1|= |V_{new}|$ and Obs 3.4.1.
The solution $\mathcal{A}$ provided by the algorithm satisfies

$$\omega(\mathcal{A}) \ \geq \ \omega(M_1) + \omega(M_2) - (|M_1|-k)$$

since the solution uses all the edges of M_2 and all but $|\mathcal{A}_2|= |M_1|-k$ edges of M_1 (as one edge of M_1 is unused for every new-new edge of M_2). By definition $\omega(M_1) = |M_1|$, and therefore the following holds: we have

$$\omega(\mathcal{A}) \ \geq \ |M_1|+\frac{2k - 3Z}{4} - (|M_1|-k) \ = \ \frac{3}{4}(2k - Z) \ = \ \frac{3}{4}\omega(\Pi).$$



(a) Graph G (b) Graph $H = G/M_1$ (c) DoubleMatching partition

Figure 3.8: The analysis of the DoubleMatching approximation ratio is tight

where Lemma 3.4.9, □

3.4.5 Tightness of the analysis.

We illustrate in Fig. 3.8 an example for which our analysis is tight. Fig. 3.8(a) depicts a complete graph on nine vertices ($k = 3$), omitting the zero-weight edges. The 2-paths of the optimal solution are shown in light gray, with a total weight of 4 (two 2-paths of weight 2 and one 2-path of weight 0). The edges of the first matching M_1 of size $k = 3$ are shown in bold (red). Fig. 3.8(b) depicts the graph H obtained after contracting the edges of M_1 in G. The new vertices resulting from contracting the edges of M_1 are depicted in white and the second matching M_2 of zero weight is depicted with dotted edges. Remember that the algorithm requires the number of old-new edges in the matching M_2 to be equal to 3, the number of old vertices in H. Fig. 3.8(c) shows the final 2-partition of the algorithm resulting from the union of the edges of the two matchings. It's weight is 3 showing that the DoubleMatching does not guarantee a better than 3/4 approximation.

3.4.6 Time Complexity

The following lemma shows that the matching M_2 in H satisfying the three properties specified in step 3 of the DoubleMatching algorithm can be computed efficiently.

Lemma 3.4.11. *Let $H = (V_{old} \cup V_{new}, E, \omega)$ be a complete $\{0,1\}$-edge-weighted graph with*

$|V_{old}|+2|V_{new}|= 3k$ for some integer k. There exists a $O(k^{2.5})$ time algorithm that computes the maximum weight matching M of H among those matchings having:

a. $\min\{|V_{new}|, |V_{old}|\}$ *old-new edges.*

b. $\max\{0, (|V_{new}|-|V_{old}|)/3\}$ *new-new edges.*

c. no old-old edges.

Proof. If $|V_{new}|\leq |V_{old}|$, then it is enough to find the maximum weight matching of size $|V_{new}|$ in the bipartite graph composed of the old-new edges of H.

If $|V_{new}|> |V_{old}|$, it is sufficient to find the maximum weight matching of size $2k - |V_{new}|$ in a new graph $\tilde{H}$ obtained from H by changing the weights of the edges as follows:

- for every pair of two old vertices $u, v \in V_{old}$ set $\omega(\{u, v\}) = -\infty$.
- for every old-new edge e, increase the original weight $\omega(e)$ by a constant $T \geq 3k$.

Clearly, a maximum weight matching in $\tilde{H}$ will not contain any edges between two old vertices. Moreover, every matching having $|V_{old}|$ many old-new edges has larger weight than a matching with fewer old-new edges. Therefore, every maximum weight matching in $\tilde{H}$ will have $|V_{old}|$ old-new edges and since the size of the matching is restricted to $|V_{old}|+(|V_{new}|-|V_{old}|)/3$, the number of new-new edges equals $(|V_{new}|-|V_{old}|)/3$.

Every matching in $\tilde{H}$ with no old-old edges and $|V_{old}|$ many old-new edges has weight exactly $T \cdot |V_{old}|$ more than its weight in H. Therefore, a maximum weight matching in $\tilde{H}$ corresponds to (has the same edges as) a maximum weight matching in H satisfying properties (a)-(c) of the lemma.

In [29] an $O(k^{2.5})$ time algorithm for finding the maximum weight matching is presented and can be easily adapted to find a maximum weight matching of a specific size. To find a maximum weight matching of fixed size s, one can add to the graph $n-2s$ dummy nodes that

are connected to all the nodes of the original graph by edges of large weight C and calculate the maximum weight matching on the new graph. The s edges with both endpoints among the vertices of the original graph form a maximum weight matching of cardinality s. □

3.5 MTP on {0, 1}-edge-weighted Graphs

In this section we direct our attention to the Maximum Triangle Partitioning (MTP) problem on $\{0, 1\}$-edge-weighted graphs and show how an approximation algorithm for the M2PP problem can be used in combination with a .5-approximation algorithm for the Triangle Packing problem to obtain an approximation algorithm for the MTP problem on $\{0, 1\}$-edge-weighted graphs.

Our algorithm relies on the existence of an α-approximation algorithm for the M2PP problem on complete $\{0, 1\}$-edge-weighted graphs, with $\alpha \geq .5$ (e.g., our .75-approximation algorithm from the previous section). Moreover, the algorithm relies on the existence of a .5-approximation algorithm for the Triangle Packing problem. Interestingly, an algorithm with a better approximation ratio for the Triangle Packing problem does not help improve the approximation ratio guarantee of our algorithm for the MTP problem. Since the 3-Set Packing problem is more general than the Triangle Packing problem, the depth 2 local search algorithm of Halldórsson [34] paraphrased below can be used for this step.

Theorem 3.5.1 (Theorem 3.4 of [34])**.** 2-*locally optimal solutions for the* 3-Set Packing *problem attain a* .5 *performance ratio.*

Informally, the TriPart algorithm (Fig. 3.9) computes two triangle partitions starting from both a 2-path partition and a triangle packing, and choses the triangle partition with higher weight. A 2-path partition can be completed to a triangle partition of at least equal weight by adding the missing edge to each individual 2-path. To complete a triangle packing

Input - G: complete $\{0,1\}$-edge-weighted graph; **2PP:** 2-path partition of G.

1. Let $G_1 = (V, E_1)$ be the unweighted graph formed by taking $E_1 = \{e \in E \mid \omega(e) = 1\}$.
2. Find a maximal triangle packing T in G_1 such that $|T|$ is at least half of the maximum triangle packing in G_1.
3. Let $G_{\overline{T}} = (V_{\overline{T}}, E_{\overline{T}}, \omega)$ be the subgraph of G, induced by the vertices of $V \setminus V(T)$.
4. Find a maximum weight matching M of size $|V_{\overline{T}}|/3$ in $G_{\overline{T}}$.
5. Complete M to a triangle partitioning T' in $G_{\overline{T}}$ by arbitrarily matching each edge $\{u, v\}$ in M to an unmatched vertex w in $V_{\overline{T}}$ and creating a triangle $\{u, v, w\}$.
6. Let $\mathcal{A}_1 = T \cup T'$.
7. Let $\mathcal{A}_2$ be the collection of triangles obtained by adding the missing edge to each 2-path of **2PP**.
8. Return $\mathcal{A}$, the solution of larger weight among $\mathcal{A}_1$ and $\mathcal{A}_2$.

Figure 3.9: The **TriPart** algorithm.

to a triangle partition the algorithm first removes the vertices of the triangles in the packing from the original graph and then finds a maximum weight matching that matches exactly 2/3 of the remaining vertices. Finally, the edges of the matching are paired with unmatched vertices to create additional triangles.

3.5.1 Algorithm analysis overview.

Let $G = (V, E, \gamma, \omega)$ be a complete $\{0,1\}$-edge-weighted graph. The following notation is used throughout this section:

- OPT - a fixed optimal solution for MTP on G;
- A - the set of triangles of OPT of weight 3; $a = |A|$;
- B - the set of triangles of OPT of weight 2; $b = |B|$;

- C - the set of triangles of OPT of weight 1; $c = |C|$.

By the definition of a, b and c, clearly $\omega(OPT) = 3a + 2b + c$. Lemma 3.5.2 shows that the weight of the triangle partition $\mathcal{A}_1$ is at least $2a+b+c$ while Lemma 3.5.3 shows that the weight of the triangle partition $\mathcal{A}_2$ is at least $\alpha(2a + 2b + c)$. In Proposition 3.5.4 we show that $\mathcal{A}_1$ has a better guaranteed approximation when OPT has mostly triangles of weight 3 and 1, and $\mathcal{A}_2$ has a better guaranteed approximation otherwise.

3.5.2 Bounding individual triangle partitions.

In this section we lower bound the weight of the individual algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ used in the TriPart algorithm. Observe that algorithm $\mathcal{A}_1$ performs better when the optimal solution contains mostly triangles of weight three, while algorithm $\mathcal{A}_2$ performs well when most of the weight of the optimal solution comes from triangles of weight one or two. In the next section we show how to combine these two results to obtain a lower bound for the TriPart algorithm.

Lemma 3.5.2. $\omega(\mathcal{A}_1) \geq 2a + b + c$.

Proof. We use $V(T)$ to denote the set of vertices of the triangles in T, and for every triangle t we use $V(t)$ to denote the set of 3 vertices of G used by the triangle t.
Claim: There exists a matching in $G_{\overline{T}}$ of size $|V_{\overline{T}}|/3$ of weight at least $2a + b + c - 3|T|$.
Let $X \subseteq V(T)$, $|X| = a$, be a set of vertices such that X contains exactly one vertex of each weight 3 triangle of OPT, i.e., for every triangle $t \in A$, $|V(t) \cap X| = 1$. Observe that the maximality of the triangle packing T ensures such a set exists, since otherwise there exists a triangle $t \in A$ such that $V(T) \cap V(t) = \emptyset$ and therefore t could be added to T to form a larger triangle packing.

Let $G_{\overline{X}} = G[V \setminus X]$ be the $\{0, 1\}$-edge-weighted graph obtained from G by removing all vertices in X. Observe that $G_{\overline{X}}$ has one edge of weight 1 from each triangle in A, and

contains all the edges of the triangles in B and C. Therefore we can create a matching $M_{\overline{X}}$ of $G_{\overline{X}}$ by choosing an edge $e \in E(G_{\overline{X}})$, $\omega(e) = 1$, from each of the triangles in A, B and C. Clearly, $|M_{\overline{X}}| = \omega(M_{\overline{X}}) = a + b + c$.

Since $M_{\overline{X}}$ is a matching in $G_{\overline{X}}$ it uses at most $|V(T) \setminus X|$ vertices from $V(T)$ and since $X \subseteq V(T)$ by definition, $|V(T) \setminus X| = |V(T)| - |X|$. Moreover, since every vertex belongs to at most one edge of $M_{\overline{X}}$, there are at most $|V(T)| - |X|$ edges of $M_{\overline{X}}$ using vertices from $V(T)$. Let $M_{\overline{T}}$ be the set of edges of $M_{\overline{X}}$ not using vertices from $V(T)$. Then $M_{\overline{T}}$ is a matching in $G_{\overline{T}}$, and $\omega(M_{\overline{T}}) = |M_{\overline{T}}| = |M_{\overline{X}}| - |V(T) \setminus X| \geq a + b + c - (|V(T)| - |X|)$. Since $|X| = a$ and $|V(T)| = 3|T|$, $\omega(M_{\overline{T}}) \geq 2a + b + c - 3|T|$.

It remains to show that $2a + b + c - 3|T| \leq |V_{\overline{T}}|/3$. To see this, observe that $|V| = 3(a+b+c)$. Thus, $|V_{\overline{T}}| = 3(a + b + c - |T|)$. Since the cardinality of the triangle packing T is at least half of the cardinality of the maximum triangle packing in G_1 and since OPT packs exactly a triangles in G_1 by definition of a, it holds that $2|T| \geq a$. Therefore $2a + b + c - 3|T| \leq a + b + c - |T| = |V_{\overline{T}}|/3$ thus completing the proof of the claim. ■

Clearly, $\omega(\mathcal{A}_1) \geq \omega(T) + \omega(M_1)$ and therefore, by the claim above it holds that

$$\omega(\mathcal{A}_1) \geq 3|T| + 2a + b + c - 3|T| = 2a + b + c,$$

thus proving the lemma. □

Observe that the bound on $\omega(\mathcal{A}_1)$ cannot be improved by simply using a better approximation algorithm for the Triangle Packing problem. The .5-approximation guarantee is used in Lemma 3.5.2 only to ensure that the size of the constructed matching is at most $|V_{\overline{T}}|/3$, as required by the step 4 of the algorithm.

Lemma 3.5.3. $\omega(\mathcal{A}_2) \geq \alpha(2a + 2b + c)$.

Proof. A 2-path partition of weight $2a + 2b + c$ can be constructed from the optimal triangle

partition OPT: create a 2-path of weight 2 from each of the triangles in A and B, and a 2-path of weight 1 from the triangles in C. Since $\mathcal{A}_2$ is an α-approximation to the M2PP problem, the lemma follows. □

3.5.3 Bounding the algorithm solution.

In this section we show that taking best triangle partition among $\mathcal{A}_1$ and $\mathcal{A}_2$ results in a solution with better approximation guarantee.

Proposition 3.5.4. $\max\{\omega(\mathcal{A}_1), \omega(\mathcal{A}_2)\} \geq \frac{2\alpha}{2\alpha+1} \cdot \omega(OPT)$.

Proof. We consider two cases independently and we use Lemma 3.5.2 or Lemma 3.5.3 according to the relative value of b.

Case 1:

$$b < \frac{(1-\alpha)(2a+c)}{(2\alpha-1)}. \tag{3.34}$$

By Lemma 3.5.2, $\omega(\mathcal{A}_1) \geq 2a + b + c$ and therefore

$$\begin{aligned}(2\alpha+1) \cdot \omega(\mathcal{A}_1) &\geq (4\alpha+2)a + (2\alpha+1)b + (2\alpha+1)c \\ &= 2\alpha \cdot \omega(OPT) + \underbrace{2(1-\alpha)a + (1-2\alpha)b + c}_{\Delta}. \end{aligned} \tag{3.35}$$

It remains to show that the term denoted by Δ in Eq. 3.35 is positive:

$$\begin{aligned}\Delta = &\ 2a \cdot (1-\alpha) - (2\alpha-1) \cdot b + c \\ \geq &\ 2a \cdot (1-\alpha) - (2\alpha-1) \cdot \frac{(1-\alpha)(2a+c)}{2\alpha-1} + c \\ = &\ c\alpha \ \geq \ 0\ , \end{aligned} \tag{3.36}$$

where the first inequality follows from the fact that $\alpha \geq .5$ and Ineq. (3.34). Therefore, from Eqs. (3.35) and (3.36) it holds that

$$\omega(OPT) \;\leq\; \frac{2\alpha+1}{2\alpha} \cdot \omega(\mathcal{A}_2), \tag{3.37}$$

thus proving the lemma for Case 1.

Case 2:

$$b \geq \frac{(1-\alpha)(2a+c)}{(2\alpha-1)}.$$

Adding a to both sides we obtain

$$\begin{aligned} a+b \;&\geq\; \left(1+\frac{2(1-\alpha)}{2\alpha-1}\right)a + \frac{(1-\alpha)}{2\alpha-1}\cdot c \\ &\geq\; \left(1+\frac{2(1-\alpha)}{2\alpha-1}\right)a \\ &=\; \frac{a}{2\alpha-1}\,, \end{aligned} \tag{3.38}$$

where the second inequality is due to the fact that $1 \geq \alpha \geq .5$, and therefore the coefficient of c in the second term is positive. Remember that by Lemma 3.5.3 it holds that $\omega(\mathcal{A}_2) \geq \alpha(2a+2b+c)$ and thus

$$\begin{aligned} \frac{2\alpha+1}{2\alpha}\cdot\omega(\mathcal{A}_2) \geq\; & \frac{2\alpha+1}{2}(2a+2b+c) \\ =\; & 2a\alpha + a + 2b\alpha + b + c\alpha + c/2 \\ \geq\; & 2a+2b+c+(a+b)(2\alpha-1) \\ \geq\; & 3a+2b+c, \end{aligned} \tag{3.39}$$

where the second inequality is due to the fact that $\alpha \geq .5$ and the third inequality is due to

Ineq. (3.38). Since by definition $\omega(OPT) = 3a + 2b + c$, Eq. (3.39) implies that

$$\omega(OPT) \;\leq\; \frac{2\alpha + 1}{2\alpha} \cdot \omega(\mathcal{A}_2), \tag{3.40}$$

thus proving the lemma for Case 2. □

Taking the output of the DoubleMatching algorithm as the 2-path partition solution in the TriPart algorithm guarantees $\alpha \geq 3/4$ and therefore Proposition 3.5.4 yields the following theorem as a corollary.

Theorem 3.1.3. TriPart *is a* $5/8$*-approximation algorithm for the* MTP *problem on* $\{0,1\}$*-edge-weighted graphs and runs in time* $O(n^{2.5})$.

3.6 Conclusions

We presented two matching based algorithms that significantly improve the approximation factor for the M2PP problem, providing a 7/12-approximation algorithm for graphs with general non-negative weights and a 3/4-approximation algorithm for $\{0,1\}$-edge-weighted graphs. Moreover, we showed that any approximation algorithm for M2PP in $\{0,1\}$-edge-weighted graphs can be transformed into a MTP approximation algorithm and in particular that the DoubleMatching algorithm for M2PP can be used to obtain a 5/8-approximation algorithm for MTP in $\{0,1\}$-edge-weighted graphs.

Although in this work we significantly improved the approximation ratio for the M2PP problem in both general and $\{0,1\}$-edge-weighted graphs, a large gap between our approximation guarantees and the best known lower bounds still exists. Moreover, even though the improved approximation guarantee for the the M2PP problem in $\{0,1\}$-edge-weighted graphs translates into a better approximation for the MTP problem in those same graphs, the better approximation for the M2PP problem in graphs with non-negative weights does not

unfortunately improve the best known approximation ratio for the MTP problem in general graphs.

Chapter 4

Team Size

4.1 Introduction

Efficiently allocating resources among multiple potential recipients is just as relevant a problem in microeconomics as in computer science. In this chapter, based on work published in the conference on Web and Internet Economics [52], we analyze the team formation problem from a game theoretic perspective where the goal of each team is to acquire as many resources (workers) as possible, while the goal of the organization at large is to allocate the workers in a way that maximizes the *social welfare*, i.e. the total efficacy of all the teams.

In the mechanism design literature it is customary to use the power of currency exchange to provide incentives for the players to be truthful. However, it has been pointed out that assuming the existence of currency in the model is not always justified ([53]). In the present work we initiate the study of mechanisms with verification, first introduced by Nisan and Ronen in [54] for the job scheduling problem, for resource allocation problems in a setting without currency. This reveals an unexplored middle ground area between the settings of the multiple choice knapsack problem and that of multi-item auctions, which has some obvious practical applications.

The knapsack problem and its variations model the setting where the supplier knows precisely what value the players are getting from any number of items. This can be thought of as a perfect verification mechanism, and selfishness does not play a role. At the other extreme, work in algorithmic game theory has generally considered the case where the supplier knows nothing about the agents' valuation and must provide incentives, typically by imposing payments, for the players to be truthful.

In this work we analyze the Funding Game, in which the supplier is a corporation that desires to improve its efficiency by hiring m additional employees and distributing them among n teams, each of whom can improve its productivity with each new team member. In this chapter we assume that the new employees are identical, so that only the number of new team members is relevant to each team's increased productivity. The productivity increase for each team is a private valuation function known only to the team's manager, and not known directly to the upper management.

We assume a simplified worker allocation process in which each team manager requests a number of new employees x_i, and specifies its value (expected increase in productivity) $\tilde{v}_i(x_i)$ for these workers, which might be less, but not more, that its real value $v_i(x_i)$. The upper management can verify that the valuations are not exaggerated, and uses a publicly known algorithm to allocate the new hires to the teams. The allocation algorithm has an impact on the requests made by players, and in effect, on the instance of the allocation problem that it will have to solve. Therefore we desire a mechanism that encourages teams to be relatively abstemious, or *not too greedy* in choosing their requests, and thereby produces an allocation yielding near-optimal social welfare.

4.1.1 Related work

We show how related literature fits in our setting, categorizing it along two orthogonal dimensions: the power of the verification mechanism and communication complexity, or

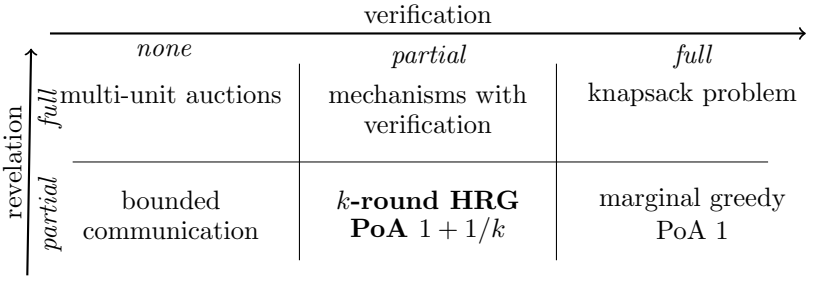


Figure 4.1: Problem settings.

metaphorically, soundness and completeness. Fig. 4.1 classifies existing work within these dimensions.

No verification, full revelation. This is the most common assumption in the algorithmic mechanism design literature. Multi-unit auctions model the situation where a verification mechanism does not exist and thus players must be assumed dishonest. Truthfulness can be achieved through VCG payments, but doing so depends on solving the allocation problem optimally, which may be intractable. Starting with the work of Nisan and Ronen [54], the field of algorithmic mechanism design has sought to reconcile selfishness with computational complexity. Multi-unit auctions have been studied extensively in this context, including truthful mechanisms for single-minded bidders [55, 56], and k-minded bidders [57–59].

More recently Procaccia and Tennenholtz ([53]), initiated the study of strategy proof mechanisms without money, which was followed by the adaptation of many previously studied mechanism design problems to the non-monetary setting ([60], [61], [62], [63], [64], [65]).

No verification, partial revelation. The multi item allocation problem has also been studied in the setting where dishonest players only partially reveal their valuation functions. The main question in this setting concerns the extent to which limiting communication complexity affects mechanism efficiency. In [66, 67], for example, bid sizes in a single-item auction are restricted to real numbers expressed by k bits. In [68], player valuation functions are only partially revealed because full revelation would require exponential space in the number of items.

Partial verification, full revelation. Mechanisms with verification have been introduced in [54] for selfish settings of the task scheduling problem. The authors show that truthful mechanisms exist for this problem when the mechanism can detect some of the lies, which is very natural in this setting. More recently, this results were generalized to mechanisms that are collusion resistant ([69]), and to more general optimization functions ([70], [71], [72]), as well as multi parameter players [73].

Full verification, full revelation. If the verification mechanism has full power to ensure agents' honesty and players must report their full valuation functions, the supplier has complete information and selfishness on the part of the recipients is irrelevant. This setting can be modeled as a multiple-choice knapsack problem solvable by FPTAS [74].

4.1.2 Contributions

This chapter extends the study of mechanisms with partial verification to multi unit resource allocation. Although polynomial time truthful mechanisms exist for multi unit auctions, these mechanisms require both full revelation of the player type, which may be hard to compute and communicate, and currency transfer, which may be impractical in some scenarios. Our work takes advantage of the additional power of verification to provide an efficient approximation mechanism for scenarios where currency transfer cannot be modeled.

We propose the highest-ratio greedy (HRG) mechanism for the Funding Game, which provides a Bayesian PoA of 2 under the assumption that valuation functions give diminishing marginal returns (Theorem 4.3.2). We also provide an algorithm that computes the Nash equilibrium strategy profile in $O(n^2 \log^2 m)$ time and a best response protocol that converges to a Nash equilibrium profile. We show that an extension of HRG to multiple rounds can arbitrarily strengthen the pure PoA. In this extension, the supplier partitions the m new hires into k carefully-sized subsets, and allocates them successively over k consecutive rounds. We show that this mechanism has a pure PoA of $1 + \frac{1}{k}$, yielding a graceful tradeoff between

communication complexity and the PoA (Theorem 4.4.3).

The remainder of the chapter is organized as follows. Section 4.2 formally introduces our model for the single round Funding Game and section 4.3 presents our results for this setting. Sections 4.4 and 4.5 deal with the multiple-round Funding Games.

4.2 Preliminaries

A single-round Funding Game is specified by a set of teams or *players* $\{1, ..., n\}$, a set of m identical new hires or *items*, and for each player i a valuation function $v_i : \{0, ..., m\} \to \mathbb{R}_0^+$ denoting the value i derives from receiving different numbers of items. We assume all valuation functions satisfy $v_i(0) = 0$, are nondecreasing, and exhibit diminishing marginal returns:

$$v_i(x) - v_i(x-1) \geq v_i(x+1) - v_i(x)$$

A strategy or *request* of player i is a pair $s_i(x_i) = (x_i, \tilde{v}_i(x_i))$ specifying the number x_i of items requested, and its valuation for these items. A request is valid if and only if $\tilde{v}_i(x) \leq v_i(x)$.

A *strategy profile* is an n-tuple of strategies $\mathbf{s} = (s_1(x_1), ..., s_n(x_n))$. We denote by $\mathcal{S}_i$ the set of valid strategies for player i, and by $\mathcal{S} = \mathcal{S}_1 \times ... \times \mathcal{S}_n$ the set of valid strategy profiles. We denote by $X = (X_1, ..., X_n)$ an allocation of the items to the players where X_i is the number of items allocated to player i. Let $\mathcal{X}$ be the set of all valid allocations, i.e. all X such that $\sum_{i \in [n]} X_i \leq m$. A *mechanism* $M : \mathcal{S} \to \mathcal{X}$ is an allocation algorithm that takes as input a strategy profile $\mathbf{s}$ and outputs an allocation X of the items to the players. We will denote by $X^M(\mathbf{s}) = (X_1^M(\mathbf{s}), ..., X_n^M(\mathbf{s}))$ the output of mechanism M for strategy profile $\mathbf{s}$. The *payoff* of player i with valuation v_i is its valuation for the number of items it has been allocated: $u_i^M(v_i; \mathbf{s}) = v_i(X_i^M(\mathbf{s}))$. If $\mathbf{v} = (v_1, ..., v_n)$ is a valuation function profile we denote by $OPT^{\mathbf{v}}$ an *optimal allocation*, by $sw(OPT^{\mathbf{v}})$ the social welfare of the optimal allocation,

and by $sw^M(\mathbf{v};\mathbf{s}) = \sum_{i\in[n]} u_i^M(v_i;\mathbf{s})$ the social welfare of strategy profile $\mathbf{s}$. We use $(s_i', \mathbf{s}_{-i})$ to denote the strategy profile $\mathbf{s}$ in which player i^{th} strategy has been replaced by s_i'.

A strategy profile $\mathbf{s}$ is a *Nash equilibrium* for a Funding Game with valuation functions $\mathbf{v}$ if for any i and any $s_i' \in \mathcal{S}_i$, $u_i^M(v_i;\mathbf{s}) \geq u_i^M(v_i; s_i', \mathbf{s}_{-i})$. The *Price of Anarchy* (PoA) bounds the ratio of the optimal social welfare and the social welfare of the worst Nash equilibrium in any Funding Game:

$$PoA^M = \sup_{\mathbf{v},\ \text{NE } \mathbf{s}} \frac{sw(OPT^{\mathbf{v}})}{sw^M(\mathbf{v};\mathbf{s})}$$

In incomplete information games we assume that player i's valuation function v_i is drawn from a set V_i of possible valuation functions, according to some distribution D_i. We denote by $D = D_1 \times ... \times D_n$ the product distribution of all players' valuation functions. A strategy σ_i in an incomplete information game is a mapping $\sigma_i : V_i \to S_i$ from the set of the possible valuation functions to the set of valid requests. Assuming that the distribution D is commonly known, the *Bayesian Nash equilibrium* is a tuple of strategies $\sigma = (\sigma_1, ..., \sigma_n)$ such that, for any player i, any valuation function $v_i \in V_i$ and any alternate pure strategy s_i':

$$\mathbb{E}_{v_{-i}\sim D_{-i}}[u_i^M(v_i;\sigma_i(v_i),\sigma_{-i}(\mathbf{v}_{-i})] \geq \mathbb{E}_{v_{-i}\sim D_{-i}}[u_i^M(v_i;s_i',\sigma_{-i}(\mathbf{v}_{-i})]$$

The *Bayesian Price of Anarchy* is defined as the ratio between the expected optimal social welfare and that of the worst bayesian Nash equilibrium:

$$BPoA = \sup_{D,\ \text{BNE } \sigma} \frac{\mathbb{E}_{\mathbf{v}\sim D}[sw^M(\mathbf{v};OPT^{\mathbf{v}}]}{\mathbb{E}_{\mathbf{v}\sim D}[sw^M(\mathbf{v};\sigma(\mathbf{v})]}$$

4.3 Single-round games

We first observe that the mechanism that solves the induced integer knapsack problem optimally has an unbounded PoA. This can be shown by the following simple example.

Assume that n items are to be allocated to n players with valuation functions $v_i(x) = 1 + x * \epsilon$ for all i and $x > 0$. A Nash equilibrium of this game is when all players request all items. The mechanism allocates all items to one player resulting in a social welfare of $1 + n \cdot \epsilon$. The optimal allocation will allocate one item to each player for a social welfare of n.

For the remainder of this section we analyze the performance of a simple greedy mechanism in a single shot game. The *Highest Ratio Greedy (HRG)* mechanism grants the requests in descending order according to the ratio $v_i(x_i)/x_i$, breaking ties in the favor of the player with lower index. If there are not enough items available to satisfy a request completely, the request is satisfied partially. This is exactly the greedy algorithm for the fractional knapsack problem. In this section we show that both the pure and Bayesian PoA are 2. An interesting open problem is whether a mechanism exist for the single round game that improves this PoA. We make use of the notion of smooth games ([75]) which we review below, cast to the Funding Games studied here. Since we are only considering the Highest Ratio Greedy mechanism we will omit the superscript M from all notations in this section.

Definition 4.3.1 (Smooth game [75])**.** *A Funding Game is (λ, μ)-smooth with respect to a choice function $c^* : V_1 \times ... \times V_n \to \mathcal{S}$ and the social welfare objective if, for any valuation function profiles $\mathbf{v}$ and $\mathbf{w}$ and any strategy profile $\mathbf{s}$ that is valid with respect to both $\mathbf{v}$ and $\mathbf{w}$, we have:*

$$\sum_{i=1}^{n} u_i(v_i; c_i^*(\mathbf{v}), \mathbf{s}_{-i}) \geq \lambda \cdot sw(\mathbf{v}; c^*(\mathbf{v})) - \mu \cdot sw(\mathbf{w}; \mathbf{s})$$

The choice function can be thought of as the optimal strategy profile, in our case the strategy profile in which each player requests the number of items received in an optimal allocation, when the valuation function profile is v.

Lemma 4.3.1. *Let $OPT^{\mathbf{v}} = (o_1^{\mathbf{v}}, ..., o_n^{\mathbf{v}})$ be an optimal allocation for valuation profile $\mathbf{v}$ and $O : V_1 \times ... \times V_n \to \mathcal{S}$ be the optimal strategy choice function, with $O(\mathbf{v}) = ((o_i^{\mathbf{v}}, v_i(o_i^{\mathbf{v}}))_{i \in [n]})$. The Funding Games are $(1, 1)$-smooth with respect to O and the social welfare objective.*

Proof. We will use o_i instead of either request $(o_i^{\mathbf{v}}, v_i(o_i^{\mathbf{v}}))$ or integer $o_i^{\mathbf{v}}$. It will be clear from context whether o_i stands for a request or an integer.

Fix valuation function profiles $\mathbf{v}$ and $\mathbf{w}$. For a strategy profile $\mathbf{s}$ valid with respect to both $\mathbf{v}$ and $\mathbf{w}$ we show that $\sum_{i=1}^{n} u_i(v_i; o_i, \mathbf{s}_{-i}) \geq sw(\mathbf{v}; O(\mathbf{v})) - sw(\mathbf{w}; \mathbf{s})$.

Let $A = \{i : u_i(v_i; o_i, \mathbf{s}_{-i}) < u_i(v_i; O(\mathbf{v}))\}$ be the set of players that are allocated more items in the optimal allocation than in profile $(o_i, \mathbf{s}_{-i})$. It is enough to show that $\sum_{i \in A} u_i(v_i; o_i, \mathbf{s}_{-i}) + sw(\mathbf{w}; \mathbf{s}) \geq \sum_{i \in A} u_i(v_i; O(\mathbf{v}))$.

For each player $i \in A$, the value per allocated item at profile $(o_i, \mathbf{s}_{-i})$ is at least $\frac{v_i(o_i)}{o_i}$ since by definition i is being allocated less than o_i items, and the valuation functions are concave. Then, $u_i(v_i; o_i, \mathbf{s}_{-i}) \geq \frac{v_i(x_i^*)}{x_i^*} \cdot X_i(o_i, \mathbf{s}_{-i})$. By definition, each player $i \in A$ would be allocated fewer items than o_i.

Therefore the requests in $\mathbf{s}_{-i}$ that have a better value per item ratio sum up to $m - X_i(c_i^*(\mathbf{v}), \mathbf{s}_{-i})$ items. Since the strategy profile $\mathbf{s}$ is assumed to be valid with respect to valuation function profile $\mathbf{w}$, the valuations expressed in $\mathbf{s}$ are at most equal to the valuations $\mathbf{w}$. We can conclude that for any $i \in A$

$$sw(\mathbf{w}; \mathbf{s}) \geq (m - X_i(o_i, \mathbf{s}_{-i})) \cdot \frac{v_i(o_i)}{o_i}$$

Then for any $i \in A$, $u_i(v_i; o_i, \mathbf{s}_{-i}) + sw(\mathbf{w}; \mathbf{s}) \geq m \cdot \frac{v_i(o_i)}{o_i}$. This is true in particular for player $j \in A$ with the highest value per item ratio $\frac{v_j(o_j)}{o_j}$. Therefore

$$\begin{aligned} \sum_{i \in A} u_i(v_i; o_i, \mathbf{s}_{-i}) + sw(\mathbf{w}; \mathbf{s}) &\geq u_j(v_j; o_j, \mathbf{s}_{-j}) + sw(\mathbf{w}; \mathbf{s}) \\ &\geq m \cdot \frac{v_j(o_j)}{o_j} \\ &\geq \sum_{i \in A} u_i(v_i; O(\mathbf{v})) \end{aligned}$$

which completes the proof. □ □

Theorem 4.3.2. *Both the pure and Bayesian Price of Anarchy for the Funding Games are equal to 2.*

Proof. Since the Funding Games are (1, 1)-smooth with respect to an optimal allocation, the extension theorem in [75] guarantees that the BPoA is bounded by 2. We now show that the pure PoA is arbitrarily close to 2. Consider the Funding Game with m items and two players with valuation functions $v_1(x) = m$ and $v_2(x) = x \ \forall x > 0$. One possible Nash equilibrium strategy is for both players to request all items. Since the value per item ratios are equal, only the first player will be allocated, for a social welfare of m. The optimal solution allocates one item to the first player and $m - 1$ items to the second player for a social welfare of $2m-1$. Taking m large enough leads to a PoA arbitrarily close to 2. □ □

4.3.1 Complexity of computing the Nash equilibrium

We now present an algorithm that finds the Nash equilibrium in the full information setting in $O(n^2 \log^2 m)$ time. For each player i we use binary search to find the largest request $(\alpha_i, v_i(\alpha_i))$ that passes the isSatisfiable test. The isSatisfiable function below assures that regardless of the other players requests, there will be at least α_i items available when the request of player i is considered by the greedy algorithm. It is easy to see that for the resulting strategy profile each player receives exactly as many items as requested and that all items are allocated. We need to show that if player i increases its request then it will not receive more items. By the construction of α_j, for any player $j \neq i$, player j will receive at least α_j items regardless of the requests of the other players. Therefore player i cannot receive more than $\alpha_i = m - \sum_{j \neq i} \alpha_j$ by changing its request.

Algorithm 1 isSatisfiable (i, x_i)

for all $j < i$ **do**
  $x_j \leftarrow \max\{x \in [m] : \frac{v_j(x)}{x} \geq \frac{v_i(\alpha_i)}{\alpha_i}\}$
end for
for all $j > i$ **do**
  $x_j \leftarrow \max\{x \in [m] : \frac{v_j(x)}{x} > \frac{v_i(\alpha_i)}{\alpha_i}\}$
end for
return true if $\sum_{j \neq i} x_j \leq m - \alpha_i$ **else false**

4.4 Multiple-round games

In this section we present our main algorithmic result. We extend the Funding Game introduced in the previous section to multiple rounds, and we show that the PoA of a k-round Funding Game is $1 + \frac{1}{k}$, yielding a graceful tradeoff between mechanism complexity and the social welfare. In a k-round Funding Game, the supplier partitions the m items into k bundles, which are distributed among the n players in k successive Funding Games or *rounds*. We assume that the supplier does not reveal the total number of available items m, nor the number of rounds k a priori. In our analysis we assume that the players play the Nash equilibrium strategy myopically, in each individual round. This assumption is in line with the maximin principle which states that rational players will choose a strategy that maximizes their minimum payoff. If players never know whether any additional items are going to be awarded in future rounds, they will try to maximize the utility in the current round. In the Funding Game, this is equivalent to playing the Nash equilibrium strategy.

As above, we use subscripts to indicate player index; we now use superscripts to indicate round index. Let $m^1, ..., m^k$ be the sizes of the bundles awarded in rounds $1, ..., k$ respectively, with $\sum_{t=1}^{k} m^t = m$. As before, the players have valuation functions $v_i : \{0, ..., m\} \rightarrow \mathbb{R}_0^+$, which are normalized ($v_i(0) = 0$), are nondecreasing, and exhibit diminishing marginal returns.

Let x_i^t be the number of items requested by player i in game t and let X^t be the allocation

vector for round t. Let $\alpha_i^t = \sum_{j=1,..,t} X_i^j$ be the cumulative number of items allocated to player i in the first t games, with $\alpha_i^0 = 0$ for all i. In round t, player i's valuation function v_i^t is its *marginal valuation* given the number of items received in the earlier rounds:

$$v_i^t(x) = v_i(x + \alpha_i^{t-1}) - v_i(\alpha_i^{t-1})$$

Observe that these marginal valuations functions v_i^t are normalized, are nondecreasing and have diminishing marginal returns, just like the full valuation functions v_i. G^t will denote the Funding Game played at round t with m^t items and valuation functions v_i^t. Observe that these individual Funding Games players are playing at each round depend on how items have been allocated in previous rounds, and indirectly, on players' strategies in previous rounds.

A strategy or request for player i is a k-tuple $s_i(x_i^1, ..., x_i^k) = (s_i^1(x_i^1), ..., s_i^k(x_i^k))$ where $s_i^t(x_i^t) = (x_i^t, v_i^t(x_i^t))$ is the request of player i in game t. We use s_i as a shorthand to denote the strategy of player i in G, and s_i^t to denote the strategy of player i in game t. A *strategy profile* for a k-round Funding Game will refer to an n-tuple of strategies $\mathbf{s} = (s_1, ..., s_n)$ and a strategy profile for game G^t will refer to the n-tuple of requests of players in round t, $\mathbf{s}^t = (s_1^t, ..., s_n^t)$. For a strategy profile $\mathbf{s}$, we will write $sw(\mathbf{s}) = \sum_{i=1}^n v_i(\alpha_i^k)$ for the social welfare of $\mathbf{s}$. Let $sw(\mathbf{s}^t)$ be the social welfare of $\mathbf{s}^t$. Let $\Delta^t = \max_i v_i^t(1)$ be the highest marginal value for one item for any player in round t. Observe that Δ^t is a nonincreasing function of t.

Definition 4.4.1. *Strategy profile* $\mathbf{s}$ *is a myopic equilibrium for the* k*-round Funding Game if for each* t, $\mathbf{s}^t$ *is a Nash equilibrium of round* t*. The myopic Price of Anarchy (PoA) bounds the ratio of the optimal social welfare and the social welfare of the worst myopic equilibrium in any* k*-round Funding Game:*

$$PoA = \sup_{\mathbf{v},\ myopic\ NE\ \mathbf{s}} \frac{sw(OPT^{\mathbf{v}})}{sw(\mathbf{s})}$$

Our goal is to analyze how a supplier should partition the m items into bundles in order to obtain as good a PoA as possible. Theorem 4.4.3 in this section shows how the PoA relates to the choices of bundle ratios, while in the next section we find the bundle ratios that give the best PoA guarantees.

Lemma 4.4.1. *For any myopic Nash equilibrium strategy profile* $\mathbf{s}$ *for a* k*-round game, we have* $\Delta^t \geq \frac{sw(\mathbf{s}^t)}{m^t} \geq \Delta^{t+1}$ *for each* t*.*

Proof: The first inequality follows from the definition of Δ^t and the diminishing returns assumption.

For the second inequality, suppose $\Delta^{t+1} > \frac{sw(\mathbf{s}^t)}{m^t}$. This would imply that either some items are not allocated at $\mathbf{s}^t$ (impossible since $\mathbf{s}^t$ is Nash equilibrium and by assumption $\Delta^{t+1} > 0$) or that some winning player i has valuation-per-item ratio $\frac{v_i^t(x_i^t)}{x_i^t} < \Delta^{t+1} = v_j^{t+1}(1)$, for some player j. But then j could have successfully requested another item in game G^t, meaning $\mathbf{s}^t$ is not Nash equilibrium, and so contradiction. □

Lemma 4.4.2. *For any myopic equilibrium* $\mathbf{s}$ *of a* k*-round Funding Game, we have:*

$$sw(OPT) \leq sw(\mathbf{s}) + \Delta^{k+1} \cdot \sum_{t=1}^{k} \left(m^t - \frac{sw(\mathbf{s}^t)}{\Delta^t} \right)$$

Theorem 4.4.3. *Let* $y_t = m^t/m^1$*. The PoA of the* k*-round Funding Game with bundle sizes* m^t *is bounded by:*

$$1 + \sup_{x_1,\ldots,x_n : x_i \geq 1} \frac{\sum_{t=1}^{k} y_t(1 - \frac{1}{x_t})}{\sum_{t=1}^{k} y_t \prod_{i=t+1}^{k} x_i} \tag{4.1}$$

Proof. Let $\mathbf{s}$ be a myopic equilibrium for a k-round game. We will show that there exist $x_1, ..., x_k$, $x_i \geq 1$, such that:

$$\frac{sw(OPT)}{sw(\mathbf{s})} \leq \frac{\sum_{t=1}^{k} y_t(1 - \frac{1}{x_t})}{\sum_{t=1}^{k} y_t \prod_{i=t+1}^{k} x_i}$$

From Lemma 4.4.2, we have:

$$sw(OPT) \leq sw(\mathbf{s}) + \Delta^{k+1} \cdot \sum_{t=1}^{k} \left(m^t - \frac{sw(\mathbf{s}^t)}{\Delta^t} \right)$$

Let $x_t = \frac{\Delta^t}{\Delta^{t+1}}$, which is at least 1 for each t. Since $\mathbf{s}^t$ is a Nash equilibrium for round t, $\Delta^{t+1} \leq \frac{sw(\mathbf{s}^t)}{m^t}$ for each t.

Then we have:

$$\begin{aligned} sw(OPT) - sw(\mathbf{s}) &\leq \Delta^{k+1} \cdot \sum_{t=1}^{k} \left(m^t - \frac{sw(\mathbf{s}^t)}{\Delta^t} \right) \\ &\leq \Delta^{k+1} \cdot \sum_{t=1}^{k} m^t \left(1 - \frac{\Delta^{t+1}}{\Delta^t} \right) \\ &\leq m^1 \Delta^{k+1} \cdot \sum_{t=1}^{k} y_t \left(1 - \frac{1}{x_t} \right) \end{aligned} \tag{4.2}$$

Observe that $\Delta^t = \Delta^{k+1} \prod_{i=t}^{k} x_i$. Therefore:

$$sw(\mathbf{s}) = \sum_{t=1}^{k} sw(\mathbf{s}^t) \geq \sum_{t=1}^{k} m^t \Delta^{t+1} \geq m^1 \Delta^{k+1} \cdot \sum_{t=1}^{k} y_t \cdot \prod_{i=t+1}^{k} x_i \tag{4.3}$$

From (4.2) and (4.3) it follows that for any k-round game with bundle sizes m^t, there exist $x_1, ..., x_k$ such that:

$$PoA = 1 + \sup \frac{sw(OPT) - sw(\mathbf{s})}{sw(\mathbf{s})} \leq 1 + \sup_{x_t \geq 1} \frac{\sum_{t=1}^{k} y_t \left(1 - \frac{1}{x_t} \right)}{\sum_{t=1}^{k} y_t \cdot \prod_{i=t+1}^{k} x_i}$$

□ □

4.5 Evaluating the PoA

In this section we present two results analyzing the expression (4.1) above. Theorem 4.5.1 shows that supremum of this expression taken over all valid choices of x_t but fixing $y_t = t$ is $1/k$. This corresponds to bundle sizes $m_1, 2 \cdot m_1, ..., k \cdot m_1$ for some m_1, indicating that the PoA *for such bundle sizes* equals $1 + 1/k$.

Second, we show that the min-sup of this expression, now also taken over choices of y_i, which corresponds to considering all possible choices of bundle sizes, equals the same value $1/k$, indicating that there is no better partition of the items.

Theorem 4.5.1. *Let*

$$F(x_1, ..., x_k) = \frac{\sum_{i=1}^{k} i(1 - \frac{1}{x_i})}{\sum_{i=1}^{k} i \prod_{j=i+1}^{k} x_j} \quad x_i \geq 1,\ i = 1, ..., k$$

Then $\sup_{\mathbf{x}} F(\mathbf{x}) = \frac{1}{k}$.

Proof. First observe that:

$$F(\mathbf{x}) = \frac{1 - \frac{1}{x_1} + \sum_{i=2}^{k} i(1 - \frac{1}{x_i})}{\sum_{i=1}^{k} i \prod_{j=i+1}^{k} x_j} < \lim_{x_1 \to \infty} F(\mathbf{x})$$

If we set $x_i = \frac{i}{i-1}$, $i = 2, ..., k$, we have $\lim_{x_1 \to \infty} F(\mathbf{x}) = \frac{1}{k}$. It remains to show that $\lim_{x_1 \to \infty} F(\mathbf{x}) \leq$

$\frac{1}{k}$. We note that the following inequalities are equivalent:

$$\lim_{x_1\to\infty} F(\mathbf{x}) \le \frac{1}{k} \Leftrightarrow \lim_{x_1\to\infty} \left(\sum_{i=1}^{k} i \prod_{j=i+1}^{k} x_j - k \sum_{i=1}^{k} i(1 - \frac{1}{x_i}) \right) \ge 0 \Leftrightarrow$$

$$\lim_{x_1\to\infty} \left(\sum_{i=1}^{k} (iz_i + ik \cdot \frac{z_i}{z_{i-1}}) - \sum_{i=1}^{k} ik \right) \ge 0, \tag{4.4}$$

$$\text{where } z_i = \prod_{j=i+1}^{k} x_j,\ i = 1, ..., k-1; z_k = 1; z_0 = x_1 z_1$$

Now define a function $C : [0, \infty)^{k-1} \to \mathbb{R}$, $C(\mathbf{z}) = \sum_{i=1}^{k} (iz_i + ik \cdot \frac{z_i}{z_{i-1}}) - \sum_{i=1}^{k} ik$. Notice that C is a function of $k-1$ variables since z_0 and z_k are fixed. Also notice that the domain of C strictly includes the domain of $\mathbf{z}$ as defined in Eq. (4.4). To complete the proof, we show that $C(\mathbf{z}) \ge 0$ for any $\mathbf{z} \in [0, \infty)^{k-1}$. We will do this in two steps: (i) showing that $C(\mathbf{z})$ has a unique stationary point, and then (ii) showing that $C(\mathbf{z}) \ge 0$ at any of the domain boundaries and the stationary point.

$\mathbf{C(z)}$ has a unique stationary point. Let $\mathbf{a} = (a_1, ..., a_{k-1})$ be a stationary point for function C, and let $a_0 = z_0 = x_1 z_1$ and $a_k = z_k = 1$:

$$\frac{\partial C}{\partial z_i}(\mathbf{a}) = i + \frac{ik}{a_{i-1}} - k(i+1)\frac{a_{i+1}}{a_i^2} = 0,\ i = 1, ..., k-1 \tag{4.5}$$

We show now by induction that each a_i can be written as a function of a_1. For the base case, let $a_0 = x_1 \cdot a_1 = f_0(a_1)$ and $f_1(a_1) = a_1$.

Now assume that $a_{i-1} = f_{i-1}(a_1)$ and $a_i = f_i(a_1)$. Then we will define a_{i+1} as a function of a_1 as follows. From Eq. (4.5) we can infer:

$$a_{i+1} = \left(i + \frac{ik}{a_{i-1}} \right) \cdot \frac{a_i^2}{k(i+1)}$$

$$a_{i+1} = \left(i + \frac{ik}{f_{i-1}(a_1)} \right) \cdot \frac{f_i^2(a_1)}{k(i+1)} \triangleq f_{i+1}(a_1) \tag{4.6}$$

where $f_{i+1}(\cdot)$ is the name given to the expression in Eq. (4.6) as a function of a_1.

Therefore the equations $a_i = f_i(a_1)$, $i = 1, ..., k-1$ uniquely define a stationary point $\mathbf{a}$ with respect to a_1. To show that the stationary point $\mathbf{a}$ is unique, we only need to show that $f_k(a_1) = 1$ has a unique solution. For this it is sufficient to show that the derivative of f_k with respect to a_1 is always positive: $f'_k(a_1) > 0$.

We show this by induction on $i = 0, ..., k$. Let $h_i = \frac{f_i}{f_{i-1}}$, $i = 2, ..., k-1$. The inductive hypothesis is that $f'_i(a_1) > 0$, $i = 1, ..., k$ and $h_j(a_1) > 0$ and $h'_j(a_1) > 0$, $j = 2, ..., k$.

For the base case, observe the following:

$$
\begin{aligned}
f_1(a_1) &= a_1 > 0 \text{ and } f'_1(a_1) = 1 > 0 \\
f_2(a_1) &= \frac{x_1 a_1^2 + k a_1}{2kx_1} \text{ and } f'_2(a_1) = \frac{2x_1 a_1 + k}{2kx_1} > 0 \\
h_2(a_1) &= \frac{f_2(a_1)}{f_1(a_1)} = \frac{x_1 a_1 + k}{2kx_1} > 0 \text{ and } h'_2(a_1) = \frac{x_1}{2kx_1} > 0
\end{aligned}
$$

Now assume that $f'_i(a_1) > 0$, $h_i(a_1) > 0$, and $h'_i(a_1) > 0$. We then observe that $f'_{i+1}(a_1)$, $h_{i+1}(a_1)$ and $h'_{i+1}(a_1)$ are all strictly positive:

$$
\begin{aligned}
f'_{i+1}(a_1) &= h'_i(a_1) \cdot f_i(a_1) + h_i(a_1) \cdot f'_i(a_1) > 0 \\
h_{i+1}(a_1) &= \left(i + \frac{ik}{f_{i-1}(a_1)} \right) \cdot \frac{f_i(a_1)}{k(i+1)} \\
&= \frac{i}{k(i+1)} f_i(a_1) + \frac{i}{i+1} \cdot h_i(a_1) > 0 \\
h'_{i+1}(a_1) &= \frac{i}{k(i+1)} f'_i(a_1) + \frac{i}{i+1} \cdot h'_i(a_1) > 0
\end{aligned}
$$

This shows that the equation $f_k(a_1) = 1$ has a unique solution, and thus concludes step (i).

$\mathbf{C}(\mathbf{z}) \geq 0$ at all boundary points and at the unique stationary point. First observe that $a_i = \frac{k}{i}$ satisfies Eq. (4.6), $i = 1, ..., k$ and hence $\mathbf{a} = (a_1, ..., a_{k-1})$ is the unique stationary

point for C. Now we show that $C(\mathbf{a}) \geq 0$:

$$C(\mathbf{a}) = \sum_{i=1}^{k}(ia_i + ik \cdot \frac{a_i}{a_{i-1}}) - \sum_{i=1}^{k} ik = \sum_{i=1}^{k}(k + k(i-1)) - \sum_{i=1}^{k} ik = 0$$

Let $\mathbf{b} = (b_1, ..., b_{k-1})$ be a boundary point. Then we must show that $C(\mathbf{b}) \geq 0$. Since $\mathbf{b}$ is a boundary point there must exist j such that $b_j = 0$ or $b_j = \infty$:

$$C(\mathbf{b}) = \sum_{i=1}^{k}(ib_i + ik \cdot \frac{b_i}{b_{i-1}}) - \sum_{i=1}^{k} ik$$

The only negative term is $\sum_{i=1}^{k} ik$, which is constant with respect to $\mathbf{b}$. If $b_j = 0$ for some j, then the positive term $(i+1)k \cdot \frac{b_{i+1}}{b_i}$ is infinite and $C(\mathbf{b}) > 0$. On the other hand, if $b_j = \infty$ for some j, then the positivwe term $ik \cdot \frac{b_i}{b_{i+1}}$ is infinite and again $C(\mathbf{b}) > 0$. Steps (i) and (ii) above show that $C(\mathbf{z}) \geq 0 \ \forall z \in [0, \infty)^{k-1}$ and therefore $C(\mathbf{z}) \geq 0$ on the restricted domain of equation (4.4), which completes the proof. □ □

Corollary 4.5.2. *The PoA for the k-round Funding Games with bundle ratios* $\frac{m_t}{m_1} = t$ *is* $1 + \frac{1}{k}$.

Theorem 4.5.3. *Let*

$$G(\mathbf{x}, \mathbf{y}) = \frac{\sum_{i=1}^{k} y_i(1 - \frac{1}{x_i})}{\sum_{i=1}^{k} y_i \prod_{j=i+1}^{k} x_j} \quad y_i \geq 0;\ x_i \geq 1,\ i = 1, ..., k$$

Then $\min_{\mathbf{y}} \sup_{\mathbf{x}} G(\mathbf{x}, \mathbf{y}) = \frac{1}{k}$.

4.6 Conclusions

In this chapter, we introduced the Funding Game, a novel formulation of resource allocation for players whose valuation declarations can be verified, but reveal only partial information. We analyzed the PoA for the pure and Bayesian Nash equilibrium and showed that allocating the resources in multiple successive rounds can improve the pure PoA arbitrarily close to 1. There are two directions in which this work can be extended. First, our mechanism relies on the assumption that the valuation functions are concave. An interesting open problem is finding an efficient mechanism for more general valuation functions. Second, it might be desirable to develop efficient verification mechanisms for combinatorial settings, where players' valuation functions are defined on subsets of items.

Chapter 5

Experimental Work

5.1 Introduction

This chapter is based on work presented at ASONAM, 2015 [50]. We attempt to understand how collaborative teams work towards solving problems, being particularly interested in: 1) within a particular team, how can we evaluate the cohesiveness of the team? 2) within a large network of potential collaborators, how can we identify teams that are likely to be highly effective?

In this effort, we construct a network-based mathematical model for collaborative work, use statistical learning techniques to motivate a focus on leader-based metrics for team evaluation, and discuss the algorithmic and complexity concerns inherent to this problem. As prescribed by much of the existing literature, incorporating all of the information about the pairwise connections between collaborators makes the problem of finding the most effective group computationally intractable. Instead, we demonstrate that the effectiveness of a team can be predicted nearly as well using only a vanishing fraction of these pairwise measurements. Moreover, by discarding most of the information, we make the problem computationally tractable.

5.1.1 Related Work

Understanding what are the components that contribute to successful team formation has long been the focus of researchers in various disciplines. A great deal of effort has been put into empirical studies of team performance of various sizes in multiple areas. Levine and Moreland in [76] publish their findings on the performance of small teams and have concluded that their progress depends on both individual skill and group cohesiveness. Hauptman and Hirji [77] monitored individual behaviour and communication patterns in tens of projects from companies in various countries and industries and found a positive correlation between individual interdependence and team performance.

More recently, research in the operation research community has been directed at finding analytical models for good team selection in projects that require multiple individuals with various sets of skills. Zakarian and Kusiak [78] build a mathematical programming model in which team membership is prioritized based on costumer requirements or product characteristics: hard constraints the members of the team must satisfy, either individually or collectively. Baykasoglu et al. [79] develop a fuzzy model aiming to address the imprecise nature of the problem. Their model is optimizing for the inclusion in the team of a varying array of hard and soft skills since the actual performance of individuals is not generally known from the get-go. Citing previous research that provided arguments for linking successful teams with group communication, Chen and Lin [80] include in their model a personality profiling indicator in addition to teamwork capability, communication skills and domain specific knowledge of individual team members.

Although many of the models proposed before took into account the importance of group communication and team cohesiveness, these were generally modelled as a set of indicators of individual team members. It wasn't until the work of Lappas, Liu and Terzi [81] that the pairwise collaboration, or compatibility, between any two members of the team has been

considered. Since every pair of individuals is assigned a compatibility score, the pool of candidates for membership in the team can be seen as a social network. Besides raising some interesting algorithmic problems, their work also implicitly posed the question of what are the meaningful ways to aggregate the pairwise compatibility between different members of a team, to a team cohesiveness score.

Different aggregation functions lead to algorithmic problems that differ substantially in their computational complexity and the resulting team structure. While in [81] the authors argue a cohesive team has small *diameter* or *minimum spanning tree* in the social network subgraph induced by the team, in their experiments this assumption often leads to large teams in which many of the team members are selected just to minimize the shortest path distance between individuals who possess the required skills. A similar model is adopted by Datta, Majumder and Naidu [82] with the objective of minimizing the maximum edge in any Steiner tree. Gajewar and Sarma [83] adopts the subgraph density (the ratio of the number of edges to the number of vertices) as the metric by which to measure cohesiveness while Li and Shan [48] use the average (weighted) degree of each individual in the subgraph. However, all of those metrics still suffer from the same flaw: they don't take into account the size of the team and often include individuals that don't add any desired hard skills, but only make the team appear more cohesive.

In the model introduced by Kargar and An [47] the team quality is judged by the sum of the distances between each pair of individuals which in general leads to much smaller teams. They consider for the first time in this line of research teams with a leader, or what we refer to as the *star topology*, in which only the compatibility between the leader and the rest of the team is considered. They show that good teams with a leader are computationally easy to find, and also give algorithms for finding multiple teams (with or without leader) for independent projects. Rangapuram, Buhler and Hein also address the problem of large teams by presenting heuristics that take as input a bound on the team size to be specified,

and a predetermined group of experts which are required to be part of the team. However, their solutions are only tested experimentally.

Other team formation literature has been concerned with online settings ([84]), where the goal is to allocate jobs fairly to individuals while still maintaining good team connectedness, minimizing the overall cost of the team ([85]) when each expert has an associated cost and pairwise compatibility reflects in the functioning cost of the team, or finding teams that can complete multiple projects, each one of them associated with a payoff ([86]).

The team formation model we propose leads to problems that are closely related to the graph packing literature. In the graph packing problem one is given an undirected, unweighted graph G and a collection of undirected, unweighted graphs $\mathcal{G}$, and has the task of finding a collection $\mathcal{M}$ of disjoint subgraphs of G such that every element of $\mathcal{M}$ is isomorphic to an element of $\mathcal{G}$. When $\mathcal{G}$ contains only a single edge graph, the graph packing problem is equivalent to the maximum matching problem. Hell and Kirkpatrick [31] showed that the problem is NP-complete for any collection $\mathcal{G}$ that is not of the form $\{K_{1,1}, K_{1,2}, ..., K_{1,T}\}$ where $T \geq 1$ is an integer and $K_{1,t}$ is the star with t leaves. When the edges are weighted even this special case is NP-complete with the best known approximation algorithm developed by Babenko and Gusakov [41] guaranteeing a $\frac{4}{9}\frac{T+1}{T}$ approximation. Approximation algorithms for packing triangles in a graph have been presented in [27] and [13] while the packing of paths of sizes two and three has been studied in [27], [13], [87], [88], [10] and [11].

5.1.2 Our Contribution

In this chapter we make several contributions towards an understanding of how teams collaborate effectively. First, we propose a robust, versatile, and realistic mathematical model based on a social network architecture. Second, we use statistical evidence from real-world data to understand what features of the social network within a team are relevant to team effectiveness. Contrary to previous assumptions, we find that it is not necessary to consider

all pairwise connections between collaborators in order to accurately predict the effectiveness of a group. Rather, the connections between the "leader" of the team and the other members are most important, the marginal predictive accuracy of the connections between other team members being small relative to the computational advantages of discarding most of the data. Third, we discuss how our method dramatically improves the computational complexity of the problem.

In Section 5.2, we describe our mathematical model of team collaboration, and our statistical model for learning about how teams effectively collaborate. The results of our statistical modeling on the DBLP data set is presented in Section 5.3, and our concluding remarks follow in Section 5.4.

5.2 Modeling

In this section, we present our mathematical model for collaboration on tasks. While our mathematical model is sufficiently general to model a variety of realistic scenarios, some of our language refers specifically to the setting of researchers writing papers collaboratively. This is simply a reflection of the fact that our primary data set is the DBLP, which contains information about authorship in computer science.

5.2.1 Mathematical Model

Let P be a set of workers, and let y be a set of tasks. Our goal is to understand how collaboration among the workers translates into success at completing tasks.

Among the workers we define a function $h : P \to \mathbb{R}^+$ that assigns an *expertise* value to each worker. In our DBLP example, we let $h(p)$ be the h-index of the researcher p. The *h-index* of a researcher is the largest integer H such that the researcher published at least H papers, each of which received at least H citations, and is generally believed to be a good

indication of a researcher's impact.

We measure two different types of collaboration among workers: pairwise collaboration between two workers, and team cohesiveness among three or more workers. To measure pairwise collaboration between researchers we posit five different functions $w_j : P \times P \to \mathbb{R}^+$ and we compare results based on different choices for the collaboration function:

1. Binary. Have these two researchers collaborated on a paper:

$$w_1(p_1, p_2) = \begin{cases} 1 & \text{if } p_1, p_2 \text{ are co-authors} \\ 0 & \text{otherwise} \end{cases}$$

2. H-index. Let A, B be the set of papers authored by p_1 and p_2, respectively. Then

$$w_2(p_1, p_2) = \text{h-index}(A \cap B).$$

3. Citations. The total number of citations that all joint papers of the two researchers received. Let $cit(A)$ denote the total number of citations papers in set A received. Then,

$$w_3(p_1, p_2) = cit(A \cap B).$$

4. J-index. The Jaccard index of the paper sets of the two authors:

$$w_4(p_1, p_2) = \frac{|A \cap B|}{|A \cup B|}.$$

5. J-citations. Jaccard index of the sets of citations received by the two authors:

$$w_5(p_1, p_2) = \frac{cit(A \cap B)}{cit(A \cup B)}.$$

Thus, each pairwise collaboration function w_j defines a *worker network* $G_j = (P, 2^P, w_j)$, where edges of zero weight correspond to non-existent edges.

A team of workers is a subgraph of the worker network, and we assume any team has a *collaboration topology*: an unweighted subgraph of the team graph in which an edge between two workers denotes the fact that the respective workers will have to collaborate. Therefore, we consider the following problem:

Problem 5.2.1. *Given a weighted graph $G = (V_G, E_G, w)$ and an unweighted graph $T(V_T, E_T)$ find a one to one mapping $g : V_T \to V_G$ which maximizes*

$$f_T(g) = f(h(g(u_1)), h(g(u_2)), \ldots, h(g(u_k)),$$
$$w_1(g(u_1), g(u_2)), \ldots, w_{\binom{k}{2}}(g(u_{k-1}, g(u_k))),$$

where k is the number of vertices in T.

In other words, we want to find a team of workers and assign them to the different nodes of the team graph (representing different job assignments), such that collaboration is maximized between all relevant pairs: the workers that are assigned to nodes connected by an edge in the graph T. We call T the collaboration topology of the team, and $f(g)$ the cohesive index of the worker assignment g. Clearly, the difficulty of computing an optimal worker assignment depends on the structure of graph T. We are interested in particular in comparing the advantages and disadvantages of creating teams with star versus clique collaboration topology. Observe that the star topology corresponds to a strictly hierarchical team structure with one leader and many independent workers, while the clique topology corresponds to a strictly flat, leaderless, team structure. While teams in real world organizations don't usually have either a perfectly flat nor perfectly hierarchical structure, we find that, for clarity, is useful to compare and contrast these two extreme topologies. When the graph T is a clique we will refer to problem 5.2.1 as the MAX-TEAMproblem, and when the graph T is a star

we will refer to the problem as the MAX-STARproblem.

When the size of T is k and all the edges of G are either 0 or 1, the MAX-TEAMproblem is equivalent to deciding whether there exists a clique of size k in G (the MAX-CLIQUEproblem) and is therefore NP-hard. However, the MAX-STARproblem can be solved in time polynomial in the size of the team and the size of G. A simple algorithm for this problem first finds for each worker v the best star centered at v by simply selecting the heaviest $k-1$ edges of G incident to v; then, it returns the best such star. Thus, while the problem statements are similar, their computational complexities are very different. The difference is obviously that in the first problem, all team subgroups—of which there are exponentially many—are equally important, whereas in the second case only the relation between the star center and the closest collaborators is important. The goal of our experimental section is to justify interest in MAX-STARin the service of answering the question posed by MAX-TEAM.

More generally, we consider the question of how to meaningfully reduce the information embedded in any given subgraph of our social network into a single number. Our results suggest that a function which considers only a small fraction of the information in G can be just as effective in addressing the real-world question of interest.

Clique Topology. One reasonable way [48, 83] to assess the collaborative strength of the team is to assume that *all* of the pairwise collaborations are relevant, and simply take the average of the edge weights. Since all of the edge weights (some of which may be 0) are included, we call this the *clique topology*.

$$f_{clique}(Q) = \frac{1}{\binom{|Q|}{2}} \sum_{(q_1,q_2)\in Q\times Q} w_j(q_1,q_2)$$

Note that using the clique topology to assess team strength implicitly asserts that all pairwise collaboration ties are meaningful. Although this may be reasonable, it has the downside of being computationally difficult. That is, to find the strongest team in a graph is

equivalent to solving the MAX-CLIQUE problem, which is not only NP-hard, but also hard to approximate [89].

Star Topology. Conversely, we consider an alternative assessment of team strength based on the *star topology*. Here, we assert that only *some* of the ties between workers are meaningful. In particular, we assert that only the pairwise collaboration between a single *team leader* and the other members of the team are important. Critically, this dramatically reduces the complexity of the computational problem, since finding the maximum-weighted star can be solved in polynomial time.

Let $q^* \in Q$ be the *team leader*. Here, q^* is defined by $q^* = \max_{q \in Q} h(q)$. Then

$$f_{star}(Q) = \frac{1}{deg_{G_j(Q)}(q^*)} \sum_{q \neq q^* \in Q} w_j(q^*, q)$$

5.2.2 Statistical Model

While the preceding discussion presents computational motivation for focusing on *stars* as opposed to *cliques*, this motivation is pointless if collaboration cannot be meaningfully characterized on the star topology. Thus, we perform statistical analysis on the DBLP data set that justifies our focus on stars.

Generally, we consider the problem of *learning* a function $f : 2^G \rightarrow \mathbb{R}^+$ that models collaboration accurately.

We consider this problem in the context of machine learning. That is, given a vector of outcomes y, and a matrix of attributes X, what function f minimizes the (squared) difference between $f(X)$ and y? The difficulty is that in our problem X is not a matrix—but rather it is a subgraph with weighted nodes and weighted edges. What function most accurately aggregates this information into a prediction?

In our analysis, y is the number of citations earned by papers listed in DBLP. Thus, we must learn a function f, taking as its input a weighted subgraph of authors and their

pairwise connections, that predicts the number of citations that a paper written by that group of authors would produce.

The function f_{star} presented above is one such function, but most likely it is not the best one. We tested a variety of candidate functions with various properties.

The performance of functions based on the star topology leads us to consider the possibility that only a few of those "spokes" are in fact meaningful. With q^* defined as above, let $w_1, w_2, \ldots, w_k$ be the edge weights between q^* and every other member of the team, *sorted* from largest to smallest. Then we consider the regression model

$$f_{star,reg}(Q) = \beta_0 + \beta_1 \cdot w_1 + \cdots + \beta_k \cdot w_k + \epsilon\,,$$

where $\epsilon \sim N(0, \sigma_\epsilon^2)$ for some constant variance σ_ϵ^2. We use training data from the DBLP to estimate the values of the β_i's.

5.2.3 Validation

For any collaboration function f, we can measure the *root mean squared error* ($RMSE_f$) of the predictions made by the function. That is,

$$RMSE_f = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(Q_i) - y_i)^2}$$

where $Q_i \subseteq P$ is the set of authors of the i^{th} paper, and y_i is the number of citations earned by the i^{th} paper.

Functions that produce smaller $RMSE$ are more accurate, and are thus preferred. Our analysis demonstrates that collaboration functions based on the star topology are competitive with those based on the clique topology.

Problem 5.2.2. COLLABORATION*: Given a weighted network $G = (V, E, w)$, a vector of*

outcomes y, and a map $M : 2^{|}G|\rightarrow y$ that maps subgraphs of G to elements of y, find a function $f : 2^G \rightarrow \mathbb{R}^+$ that minimizes $||f(Q) - y||_2^2$ (e.g. $RMSE_f$) over all subsets $Q \subseteq V$.

5.3 Experimental Results

In this section we describe our efforts to learn about the collaboration function that translates attributes of researchers in the DBLP into predictions about how many citations their papers will earn. The DBLP contains data about computer science authorship, and we obtained data about the number of citations earned by each paper from [90].

5.3.1 Small Teams

In the first experiment we evaluated how well different combinations of pairwise collaboration functions and team collaboration topology assumptions predict the strength of a team. The data set we used for this experiment consists of 24,020 papers with at most five authors published in eight journals and twenty conferences in theoretical computer science between 1954 and 2002. This data was then further divided into two subsets: *training* data consisting of 21,006 papers published from 1954-2000, and *testing* data consisting of the remaining 3,014 papers published in 2001-2002.

We evaluate each of the combinations of the pairwise collaboration function with a team collaboration aggregation function compared to the observed strength of a team: the number of citations the newly published paper received. We use the Pearson product-moment correlation coefficient (PCC) to assess how good the author collaboration and aggregation functions are. The PCC of two random variables is a number between -1 and 1 describing how the variables are correlated. I our setting, a PCC of 1 would suggest that the number of ciatations of a paper grows proportionally with the team cohesiveness, while a PPC of 0 would suggest that team cohesiveness does not have any predictive power in identifying

successful teams. The PCC of two random variables is defined as the covariance of the two variables divided by the product of their standard deviations:

$$\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

Table 5.1: PPC between team cohesiveness and number of citations with different pairwise collaboration metrics and aggregation functions.

	star	clique
binary	.218	.163
h-index	.150	.130
citations	.195	.194
j-index	.007	-.013
j-citations	-.034	-.037

Table 5.1 shows how the prediction of the team strength based on previous collaboration of authors is related to the number of citations of respective papers. Each row shows the results for a different function for computing the pairwise author collaboration, as described in section 5.2. The columns show the results when different team collaboration topologies are considered.

In each cell we list the Pearson correlation coefficients between the team cohesiveness score and the number of citations of a paper. For example, when the pairwise collaboration between two authors is calculated as the total number of citations the authors received from common papers, and the team collaboration topology is assumed to be a clique, then the team cohesiveness score had a .194 Pearson correlation with the number of citations the paper received. It is interesting to notice that the best correlation is achieved in the setting that considers the least amount of information: the star topology with the binary pairwise collaboration function. The binary collaboration function is the simplest, in that it only records whether two authors collaborated previously or not, without taking into account the performance of the teams in which the authors collaborated previously. Besides, the

correlation coefficient is always better when the star topology is assumed than when the collaboration between all pairs of authors are given equal importance.

It is surprising as well to observe that some author collaboration functions don't perform as well as one would expect. The Jaccard index based collaboration functions do not show a relevant correlation to the number of citations in any of the team collaboration topologies. This collaboration function has been used in the experimental evaluation sections of [83] and [81]. To get a perspective on the relevance of the correlation coefficients in Table 5.1 note that a (positive or negative) Pearson correlation coefficient of .037 can occur by coincidence in unrelated distributions with a probability of about 10%, while a coefficient of .15 can occur in unrelated distributions with a probability of 10^{-6}.

We also analyzed the the correlation between the number of citations and the sum of the author expertise or h-indexes to obtain a better correlation: .323. This confirms the common sense expectation that individual expertise is most crucial to the success of a team, but, the small difference between this correlation coefficient and the best correlation coefficients in table 5.1 suggests that choosing team members that collaborate well is almost as important as choosing team members with high expertise.

5.3.2 Large teams

This data set consists of 622,094 papers published in a wide array of venues between 1990 and 2003. From this data we were able to construct a graph G with 43,453 authors, having between them 181,812 pairwise connections, and authoring 240,621 papers.

Given the larger size of the latter data set, we focus our presentation here on those results. However, the results from the smaller data set were consistent.

As noted in Section 5.2, we take as input a large social network G, where the nodes are weighted by the h-index of each author on all papers published before 2001. Similarly, the pairwise connections between authors were computed as the h-index of the shared publi-

cations of those authors on all papers published before 2001. Thus, the background data embedded in G consists entirely of publications before 2001.

The results of testing six regression-based prediction models are shown in Table 5.3 and Table 5.4. Each model is evaluated in terms of its ability to estimate the logarithm of the number of citations of each paper (+1). That is, the response variable is $\ln(citations + 1)$ and thus the units of the RMSE figures are in log-citations. The models are as follows:

- *mean*: assign every paper the mean number of citations. This serves as our baseline.
- *year*: since papers only accrue more citations over time, use the year in which the paper was published to predict its citations. Here, both a linear and quadratic term for year are allowed.
- *numAuthors*: use the number of authors on the paper to predict citations. It is assumed that there is no correlation between citations and number of authors, but this serves as a useful verification of a random noise model.
- *density*: $density(G) = \frac{1}{\binom{n}{2}} \sum_{e \in E(G)} w(e)$. This captures the idea from the clique topology that all edges are important, and that teams with stronger pairwise connections will perform better.
- *star*: $star(G) = \max_{v \in V(G)} \frac{\sum_{w(e) \in N_G(v)}}{deg_G(v)+1}$, where $N_G(v)$ is the neighborhood of v in G. This captures the idea from the star topology that only the mostly strongly-connected member of the team is important.
- *topTwo*: use the largest two h-indices and the largest two edge weights in the subgraph to predict the number of citations. While not a star topology, this captures the notion that only the two best researchers and two strongest ties are important. The magnitude of the other weights in the subgraph are less important.

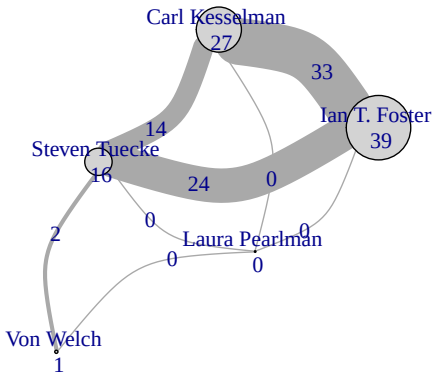


Figure 5.1: An example team graph.

An example illustrating these collaboration functions in shown in Figure 5.1. Here, we consider the paper "A community authorization service for group collaboration", published in 2002 by Laura Pearlman, Von Welch, Ian T. Foster, Carl Kesselman, and Steven Tuecke [91]. The h-indices and joint h-indices of the five co-authors prior to 2001 are indicated numerically in the figure. This paper has received 504 citations.

Table 5.2 shows the predictions of the six models above for the paper considered in Figure 5.1. The mean citations predicted by the baseline model is 17.5 (for all papers). In this case, only the *topTwo* model performs significantly better than that. While the scale of the scores is irrelevant, and the details of how these scores translate into predicted citations are omitted, what should be clear is how the topology of this group is reflected in the different scores. In this example, three people with high h-indices have very strong connections, while two people appear to new to the group, and perhaps computer science research in general. Although in this example, the density metric performs slightly better than the star, it is clear how the strength of a few people can drive the success of the group.

Table 5.3 compares the performance of each of these function on the training data, which consisted of 15,845 papers which had at least three authors. As expected, *year* and *numAuthors* are largely indistinguishable from the baseline model that simply uses the mean. Moreover, the model based on *density* is not much better. However, predictions

Table 5.2: The predicted number of citations for each of six prediction models, for the paper whose collaboration graph is shown in Figure 5.1.

f	score	predicted citations
mean		17.5
year	2001	15.4
numAuthors	5	17.6
density	7.3	17.4
star	5.7	16.6
topTwo		36.5

based on the *star* metrics offer a considerable improvement, and focusing on the *topTwo* is even better.

Table 5.3: Results from in-sample testing of various collaboration functions on the DBLP. Each function was trained on 15,845 papers and evaluated on those same papers.

	RMSE
mean	1.705
year	1.703
numAuthors	1.705
density	1.711
star	1.669
topTwo	1.540

In Table 5.4, we compare the performance of these same functions on a hold-out set of 13,753 papers on which they were not trained. The conclusions are largely the same. While the *density* metric performs better than the baseline model against this sample, it is still surpassed by the *star* and *topTwo* models.

In Figures 5.2 and 5.3, we show the bivariate relationship between the actual number of citations earned by each paper, and the team cohesiveness score with the *star* metric and *density* metric respectively.

Table 5.4: Results from out-of-sample testing of various collaboration functions on the DBLP. Each function was trained on 15,845 and evaluated on 13,753 different papers.

	RMSE
mean	1.706
year	1.690
numAuthors	1.708
density	1.710
star	1.695
topTwo	1.582

5.4 Conclusion

In this section we present the limitations of our work, explore avenues for future study, and summarize our findings.

5.4.1 Limitations & Future Work

We have explored only a small portion of the infinite space of potential collaboration functions. Our motivation in this effort was to find collaboration functions that led to computationally tractable algorithms that performed as well or better than exponential time algorithms necessary for MAX-TEAM. In that effort we have been successful, but it is certain that these collaboration are not optimal. A worthwhile goal for future study would be to explore the infinite space of potential collaboration functions to find an optimal one.

Here, we have considered only the DBLP data set, which is specific to computer scientists writing academic papers. Thus, our conclusions about the usefulness of the star topology relative to the clique topology apply narrowly to this setting. While we suspect that finding likely extends to other application domains, we have no evidence other than intuition to support this claim.

One challenging aspect of this data set was the heavily right-skewed distributions for h-index and number of citations. This was especially troublesome for the response variable of

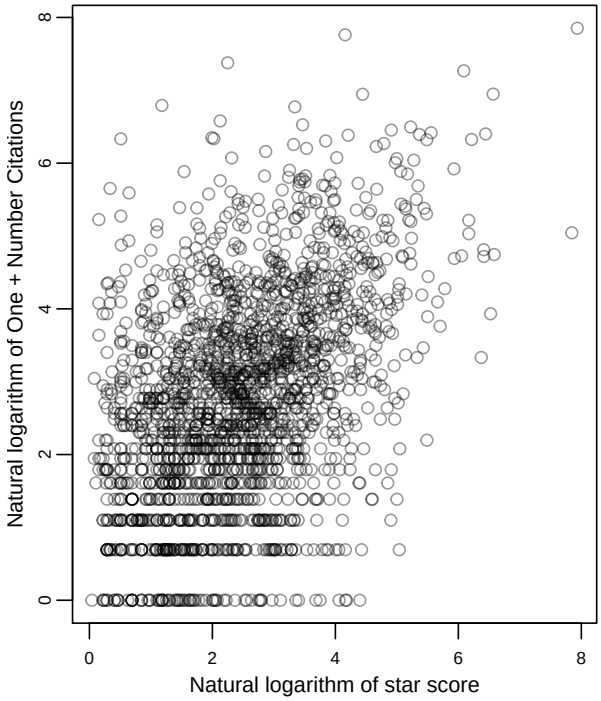


Figure 5.2: Paper citations vs. the team cohesiveness score using the *star* metric and number of citations.

number of citations. While our remedy of taking the logarithm was an improvement, a more careful study might explore more sophisticated statistical transformations. In particular, both quantities appear to follow zero-inflated negative binomial distributions (see Figure 5.4). It seems likely that successfully transforming these variables could result in more accurate predictions.

5.4.2 Conclusion

The question of how people collaborate most effectively is both important and not fully understood. We used the DBLP database of computer science authorship to address this question. Whereas previous research focused on the strength of ties among *all* members of a team, we focused on the strength of ties among only *some* of the members of a team. In particular, we used the DBLP to provide experimental justification for a focus on the star

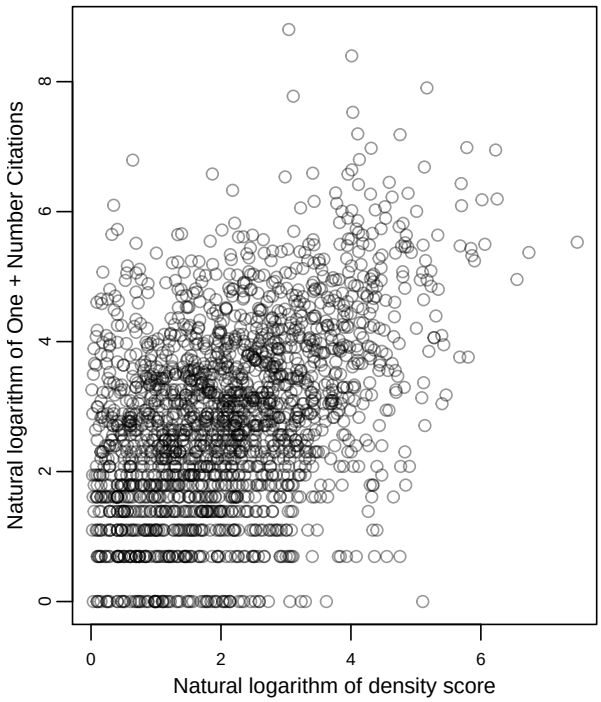


Figure 5.3: Paper citations vs. the team cohesiveness score using the *density* metric and number of citations.

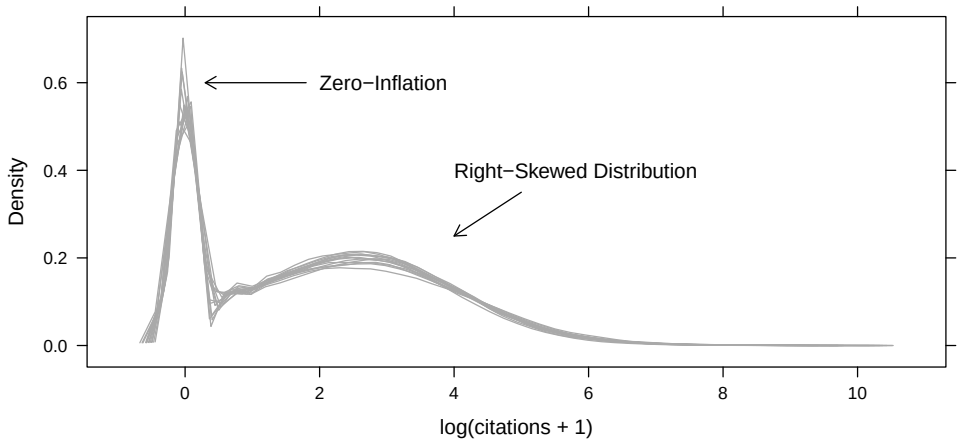


Figure 5.4: Distribution of the Logarithm of the Number of Citations. Each line represents one year from 1990 to 2003. Note the zero-inflation and right-skewed distribution.

topology over the clique topology. This experiment evidence suggests that metrics based on the star topology are competitive with those based on the clique, but are computationally tractable. This makes a practical case for using the star topology when mining for effective subgroups in large social networks.

Bibliography

[1] Kurt Lewin. Group decision and social change. *Readings in social psychology*, 3:197–211, 1947.

[2] J. Richard Hackman. *The design of work teams*. Prentice-Hall, Englewood Cliffs, NJ, 1987.

[3] Daniel Levi. *Group dynamics for teams*. Sage Publications, 2015.

[4] Warren Bennis and Patricia Biederman. *Organizing genius*. Basic Books, 2007.

[5] Richard Guzzo and Marcus Dickson. Teams in organizations: Recent research on performance and effectiveness. *Annual review of psychology*, 47(1):307–338, 1996.

[6] Charles Evans and Kenneth Dion. Group cohesion and performance: A meta-analysis. *Small group research*, 22(2):175–186, 1991.

[7] Michael Campion, Gina Medsker, and Catherine Higgs. Relations between work group characteristics and effectiveness: Implications for designing effective work groups. *Personnel psychology*, 46(4):823–847, 1993.

[8] Sven O. Krumke and Clemens Thielen. The generalized assignment problem with minimum quantities. *European Journal of Operational Research*, 228(1):46–55, 2013.

[9] R. Hassin and S. Rubinstein. An approximation algorithm for maximum triangle packing. *Discrete Applied Math.*, 154(6):971 – 979, 2006.

[10] R. Tanahashi and Z. Chen. A deterministic approximation algorithm for maximum 2-path packing. *IEICE Trans. on Information and Systems*, February 2010.

[11] Anke van Zuylen. Multiplying pessimistic estimators: Deterministic approximation of max tsp and maximum triangle packing. *COCOON*, pages 60–69, 2010.

[12] R. Hassin and O. Schneider. A local search algorithm for binary maximum 2-path partitioning. *Discrete Optimization*, 10(4):333 – 360, 2013.

[13] Z.Z. Chen, R. Tanahashi, and L. Wang. An improved randomized approximation algorithm for maximum triangle packing. *Discrete Applied Mathematics*, 157(7):1640 – 1646, 2009.

[14] Amotz Bar-Noy and George Rabanca. Tight approximation bounds for the seminar assignment problem. In *International Workshop on Approximation and Online Algorithms*, pages 170–182. Springer, 2016.

[15] Marco Bender, Clemens Thielen, and Stephan Westphal. A constant factor approximation for the generalized assignment problem with minimum quantities and unit size items. *Mathematical Foundations of Computer Science*, pages 135–145, 2013.

[16] Marco Bender, Clemens Thielen, and Stephan Westphal. Erratum: A constant factor approximation for the generalized assignment problem with minimum quantities and unit size items. *Mathematical Foundations of Computer Science*, pages E1–E3, 2013.

[17] Dirk G. Cattrysse and Luk N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research*, 60(3):260 – 272, 1992.

[18] O. Erhun Kundakcioglu and Saed Alizamir. *Generalized assignment problem*, pages 1153–1162. Springer US, Boston, MA, 2009.

[19] David B. Shmoys and Éva Tardos. An approximation algorithm for the generalized assignment problem. *Math. Program.*, 62(3):461–474, December 1993.

[20] Reuven Cohen, Liran Katzir, and Danny Raz. An efficient approximation for the generalized assignment problem. *Inf. Process. Lett.*, 100(4), November 2006.

[21] Lisa Fleischer, Michel X. Goemans, Vahab S. Mirrokni, and Maxim Sviridenko. Tight approximation algorithms for maximum general assignment problems. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, 2006.

[22] George L. Nemhauser and Laurence A. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. *North-Holland Mathematics Studies*, 59:279–301, 1981.

[23] Michele Conforti and Grard Cornujols. Submodular set functions, matroids and the greedy algorithm: Tight worst-case bounds and some generalizations of the rado-edmonds theorem. *Discrete Applied Mathematics*, 7(3):251 – 274, 1984.

[24] Rakesh V. Vohra and Nicholas G. Hall. A probabilistic analysis of the maximal covering location problem. *Discrete Appl. Math.*, 43(2), May 1993.

[25] Samir Khuller, Anna Moss, and Joseph (Seffi) Naor. The budgeted maximum coverage problem. *Inf. Process. Lett.*, 70(1):39–45, April 1999.

[26] Maxim Sviridenko. A note on maximizing a submodular set function subject to a knapsack constraint. *Oper. Res. Lett.*, 32(1), 2004.

[27] A. Bar-Noy, D. Peleg, G. Rabanca, and I. Vigan. Improved approximation algorithms for weighted 2-path partitions. *ESA*, 2015.

[28] H. Gabow. An efficient implementation of Edmonds' algorithm for maximum matching on graphs. *J. ACM*, 23(2):221–234, April 1976.

[29] S. Micali and V.V. Vazirani. An algorithm for finding maximum matching in general graphs. In *Proc. 21st FOCS*, pages 17–27, 1980.

[30] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[31] P. Hell and D. C. Kirkpatrick. Packings by complete bipartite graphs. *SIAM J. Algebraic Discrete Methods*, 7(2):199–209, April 1986.

[32] Z. Lonc. On the complexity of some edge-partition problems for graphs. *Discrete Applied Mathematics*, 70(2):177 – 183, 1996.

[33] C.A.J. Hurkens and A. Schrijver. On the size of systems of sets every t of which have an sdr, with an application to the worst-case ratio of heuristics for packing problems. *SIAM J. Discrete Math.*, 2(1):68–72, 1989.

[34] M. Halldórsson. Approximating discrete collections via local improvements. In *SODA*, pages 160–169, 1995.

[35] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37(1):27–35, 1991.

[36] M. Chlebík and J. Chlebíková. Approximation hardness for small occurrence instances of NP-hard problems. In *ISAAC*, pages 152–164, 2003.

[37] V. Guruswami, C.P. Rangan, M.S. Chang, G.J. Chang, and C.K. Wong. The vertex-disjoint triangles problem. In *WG*, pages 26–37, 1998.

[38] G. Manic and Y. Wakabayashi. Packing triangles in low degree graphs and indifference graphs. *Discrete Math.*, 308(8):1455 – 1471, 2008.

[39] René van Bevern, Robert Bredereck, Laurent Bulteau, Jiehua Chen, Vincent Froese, Rolf Niedermeier, and Gerhard J. Woeginger. Partitioning perfect graphs into stars. *Journal of Graph Theory*, 2016.

[40] E. Prieto and C. Sloper. Looking at the stars. *Theoretical Computer Science*, 351(3): 437–445, 2006.

[41] Maxim Babenko and Alexey Gusakov. New exact and approximation algorithms for the star packing problem in undirected graphs. *STACS*, pages 519–530, 2011.

[42] E. Arkin and R. Hassin. On local search for weighted k-set packing. *Math. Operations Research*, 23(3):640–648, 1998.

[43] R. Hassin and S. Rubinstein. Erratum to "An approximation algorithm for maximum triangle packing": [Discrete Applied Math. 154 (2006) 971-979]. *Discrete Applied Math.*, 154(18):2620 –, 2006.

[44] Z.Z. Chen, R. Tanahashi, and L. Wang. Erratum to, "An improved randomized approximation algorithm for maximum triangle packing". *Discrete Applied Mathematics*, 158(9):1045 – 1047, 2010.

[45] R. Hassin and S. Rubinstein. An approximation algorithm for maximum packing of 3-edge paths. In *Information Processing Letters 63*, pages 63–67, 1997.

[46] A. Gajewar and A. S. Sarma. Multi-skill collaborative teams based on densest subgraphs. In *SDM*, pages 165–176. SIAM / Omnipress, 2012.

[47] M. Kargar and A. An. Discovering top-k teams of ex-perts with/without a leader in soc. net. *CIKM*, 2011.

[48] C. Li and M. Shan. Team formation for generalized tasks in expertise social networks. *ICSC*, 2010.

[49] S. Rangapuram, T. Buhler, and M. Hein. Towards realistic team formation in social networks based on densest subgraphs. *WWW*, 2013.

[50] A. Bar-Noy, P. Basu, B. Baumer, and G. Rabanca. Star search: Effective subgroups in collaborative social networks. *ASONAM*, 2015.

[51] B. Courcelle. The monadic second-order logic of graphs xiv: uniformly sparse graphs and edge set quantifications. *Theoretical Computer Science*, 299(13):1 – 36, 2003.

[52] Amotz Bar-Noy, Yi Gai, Matthew P Johnson, Bhaskar Krishnamachari, and George Rabanca. Funding games: The truth but not the whole truth. In *WINE*, pages 128–141. Springer, 2012.

[53] Ariel D. Procaccia and Moshe Tennenholtz. Approximate mechanism design without money. *EC*, 2009.

[54] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 2001.

[55] Ahuva Mu'alem and Noam Nisan. Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract. *Eighteenth national conference on Artificial intelligence*, 2002.

[56] Patrick Briest, Piotr Krysta, and Berthold Vöcking. Approximation techniques for utilitarian mechanism design. *STOC*, 2005.

[57] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. *FOCS*, 2005.

[58] Shahar Dobzinski and Noam Nisan. Mechanisms for multi-unit auctions. *EC*, 2007.

[59] Shahar Dobzinski and Noam Nisan. Multi-unit auctions: beyond roberts. *EC*, 2011.

[60] Noga Alon, Felix Fischer, Ariel Procaccia, and Moshe Tennenholtz. Sum of us: strategyproof selection from the selectors. *TARK*, 2011.

[61] Itai Ashlagi, Felix Fischer, Ian Kash, and Ariel D. Procaccia. Mix and match. *EC*, 2010.

[62] Ning Chen, Nick Gravin, and Pinyan Lu. Mechanism design without money via stable matching. *CoRR*, abs/1104.2872, 2011.

[63] Shaddin Dughmi and Arpita Ghosh. Truthful assignment without money. *EC*, 2010.

[64] Mingyu Guo and Vincent Conitzer. Strategy-proof allocation of multiple items between two agents without payments or priors. *AAMAS*, 2010.

[65] Pinyan Lu, Xiaorui Sun, Yajun Wang, and Zeyuan Allen Zhu. Asymptotically optimal strategy-proof mechanisms for two-facility games. *EC*, 2010.

[66] Liad Blumrosen and Noam Nisan. Auctions with severely bounded communication. *FOCS*, 2002.

[67] Liad Blumrosen, Noam Nisan, and Ilya Segal. Multi-player and multi-round auctions with severely bounded communication. *ESA*, 2003.

[68] Wolfram Cohen and Tuomas Sandholm. Partial-revelation vcg mechanism for combinatorial auctions. *AAAI*, 2002.

[69] Paolo Penna and Carmine Ventre. Collusion-resistant mechanisms with verification yielding optimal solutions. *ESA*, 2008.

[70] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, Giuseppe Persiano, and Carmine Ventre. New constructions of mechanisms with verification. *ICALP*, 2006.

[71] Vincenzo Auletta, Roberto De Prisco, Paolo Penna, and Giuseppe Persiano. The power of verification for one-parameter agents. *J. Comput. Syst. Sci.*, 75:190–211, May 2009.

[72] Alessandro Ferrante, Gennaro Parlato, Francesco Sorrentino, and Carmine Ventre. Fast payment schemes for truthful mechanisms with verification. *Theor. Comput. Sci.*, 410: 886–899, March 2009.

[73] Carmine Ventre. Mechanisms with verification for any finite domain. *WINE*, 2006.

[74] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[75] Tim Roughgarden. The price of anarchy in games of incomplete information. *SIGecom Exch.*, 11(1), June 2012.

[76] J. Levine and R. Moreland. Progress in small group research. *Annual Review of Psychology*, 41, 1990.

[77] O. Haupman and K. K. Hirji. The influence of process concurrency on project outcomes in product development: an empirical study of cross-functional teams. *IEEE Trans. on Eng. Management*, 44, 1996.

[78] A. Zakarian and A. Kusiak. Forming teams: an analytical approach. *IIE Trans.*, 31(1): 85–97, 1999. doi: 10.1023/A:1007580823003.

[79] A. Baykasoglu, T. Dereli, and S. Das. Project team selection using fuzzy optimization approach. *Cybern. Syst.*, 38(2), February 2007.

[80] S. Chen and L. Lin. Modeling team member characteristics for the formation of a multifunctional team in concurrent engineering. *IEEE Trans. on Engineering Management*, 51(2), May 2004.

[81] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts. *KDD*, 2009.

[82] S. Datta, A. Majumder, and K. Naidu. Capacitated team formation problem on social networks. *KDD*, 2012.

[83] A. Gajewar and A. Sarma. Multi-skill collaborative teams based on densest subgraphs. *SDM*, 2012.

[84] A. Anagnostopoulos, L. Becchetti, C. Castillo, A. Gionis, and S. Leonardi. Online team formation in social networks. *WWW*, 2012.

[85] M. Kargar, A. An, and M. Zihayat. Efficient bi-objective team formation in social networks. *ECML PKDD*, 2012.

[86] B. Golshan, T. Lappas, and E. Terzi. Profit-maximizing cluster hires. In *KDD*, 2014.

[87] R. Hassin and S. Rubinstein. An approximation algorithm for maximum triangle packing. *Discrete Appl. Math.*, 154(6):971–979, April 2006.

[88] R. Hassin and O. Schneider. A local search algorithm for binary maximum 2-path partitioning, November 2013.

[89] J. Hastad. Clique is hard to approximate within $n^{(1-\epsilon)}$. In *Acta Mathematica*, pages 627–636, 1996.

[90] ArnetMiner.org. *DBLP Papers + Citation Relationship*, accessed June, 2014. URL `http://arnetminer.org/DBLP_Citation`.

[91] L. Pearlman, V. Welch, I. Foster, C. Kesselman, and S. Tuecke. A community authorization service for group collaboration. In *Policies for Distributed Systems and Networks*, pages 50–59. IEEE, 2002.