Vertex Stabilizers for Network Bargaining Games

by

Matthew Gerstbrein

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Masters of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2022

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Network bargaining games form a prominent class of examples of game theory problems defined on graphs, where vertices represent players, and edges represent their possible interactions. An instance of a *network bargaining game* is given by a graph $G = (V, E)$ with edge weights $w \in \mathbb{R}_+^E$ and vertex capacities $c \in \mathbb{Z}_+^V$. A *solution* to an instance (G, w, c) of a network bargaining game is data (M, z), where $M \subset E$ is a c-matching, and $z \in \mathbb{R}_{\geq 0}^{2E}$ is a vector which assigns each edge $uv \in E$ a pair of values z_{uv} and z_{vu}, such that $z_{uv} + z_{vu} = w_{uv}$ if $uv \in M$, and $z_{uv} = z_{vu} = 0$ otherwise. An instance (G, w, c) is said to be *stable* if it admits a so-called 'stable' solution, which represents a solution where a player has no incentive to deviate. Not all instances of a network bargaining game have a stable solution, and this naturally motivates the problem of how to modify the underlying graph such that the instance becomes stable. In recent years, researchers have investigated various modifications, typically by adding or removing edges or vertices. The natural algorithmic question which stems from this is whether these modifications can be performed efficiently. The answer varies, depending on the modification in question, and on whether the edges/vertices have been restricted to be unit weight/capacity.

In this work, we consider the vertex-removal setting for a general instance (G, w, c) of a network bargaining game. A set of vertices whose deletion from G results in a stable instance of the induced network bargaining game is called a *vertex stabilizer*. We demonstrate in this work that the algorithmic problem of finding a minimum cardinality vertex stabilizer is NP-complete, and give an efficient 2-approximation algorithm. Further, we show that no efficient $(2-\epsilon)$-approximation for this problem exists for any $\epsilon > 0$, assuming the Unique Games Conjecture holds. These results hold even in the case when all edges are of unit-weight.

In contrast, if we are given an instance (G, w, c) together with a maximum weight c-matching M, we show that the problem of finding a minimum cardinality vertex stabilizer that avoids M can be solved efficiently. We provide a polynomial time algorithm for solving this problem.

Acknowledgements

There are two individuals who made this thesis possible: Laura Sanità and Jochen Koenemann. I am truly grateful for the instruction which they both have given me throughout the process. They have both offered me more than I deserve. I further extend my thanks to Melissa Cambridge - this process would have been far rockier had it not been for her support in negotiating the logistics inherent to life in a large university. I would also like to thank Peter Nelson, Kevin Purbhoo, and Steven Vavasis: each of these individuals has demonstrated to me a manner in which to excel in mathematical pedagogy, beyond what I would have otherwise thought possible. It is my good fortune to have had the opportunity to be taught by them.

To all of these individuals: my sincere thanks. It is easy to feel like a number within a much larger system, but your attitudes and actions demonstrate otherwise.

Finally, I would like to thank my family - specifically my parents, godparents, and grandmother. Their steadfast support is at the root of any success I might ever find. I am unable to imagine a world without them.

Table of Contents

List of Figures

Chapter 1

Introduction

Graphs are combinatorial objects which are defined in a very general way: formally a *graph* is a pair of sets $G = (V, E)$, where V is a set of points, called *vertices*, and E is a set of (unordered) pairs of vertices, called *edges*. Intuitively, one views the vertices as 'objects', and the edges as 'links' between the objects. The generality of the definition is due to the lack of restriction on what constitutes an 'object', and this flexibility allows one to robustly model many physical phenomena, among which game-theoretic situations feature prominently. In many cases, game theory problems are formulated on graphs such that the players and their potential interactions are modeled by vertices and edges of a graph, respectively. Classical examples include *cooperative matching games*, introduced by Shapley and Shubik [SS71], and *network bargaining games*, introduced by Kleinberg and Tardos [KT08]. Instances of these games are played on a given graph $G = (V, E)$, and the goal, roughly speaking, is to assign values to players in such a way that no player could achieve an individually greater value by interacting with neighboring players in some mutually beneficial manner. Before formally defining these games, we first introduce some underlying terminology.

Given a graph $G = (V, E)$, the *endpoints* of an edge $\{u, v\} \in E$ are the vertices u and v. We say that edge $\{u, v\}$ is *incident* with u (and v), and that u and v are *adjacent*. The *degree* of a vertex u is the number of edges incident with it. Then a *matching* in G is a set $M \subseteq E$ such that every vertex $u \in V$ is incident with at most one edge in M. A matching is a *maximum (cardinality) matching* in G if no other matching in G has strictly greater cardinality. This notion can be extended to a weighted generalization: if G comes equipped with edge weights, given by $w \in \mathbb{R}_+^E$, then the weight of a matching M is $\sum_{e \in M} w_e$, and M is a maximum weight matching if no other matching in G has strictly greater weight. We define $\nu(G)$ to be the weight of any maximum weight matching in G. Note that in the

cardinality-case (i.e if no edge weights are given, or equivalently if every component of w is 1), $\nu(G)$ represents the size of a maximum cardinality matching in G.

In both types of games, the total value that players can collectively accumulate is $\nu(G)$. If values $x \in \mathbb{R}^V_{\geq 0}$ are allocated to each player such that $\sum_{v \in V} x_v = \nu(G)$, players may or may not be content with their current allocation, as a given player may have been able to act in a way which would have increased their allocation. Such a consideration motivates the following general question: is it possible to allocate values $x \in \mathbb{R}^V_{\geq 0}$ in a way such that no player has an incentive to deviate from the current configuration? In many (but not all) cases, the answer is affirmative, and such configurations are said to be 'stable'. In the case of both cooperative matching games and network bargaining games, a formal notion of stability has been introduced, and moreover, a tight link between stable solutions of these types of games has been established. In particular, if an instance of each type of game is played on the same graph G, then it has been shown that either a stable solution exists for both games, or for neither game (with respect to each notion of stability). We now provide formal definitions for each type of game.

In an instance of a cooperative matching game, one wants to find an *allocation* of the value $\nu(G)$, given by a vector $x \in \mathbb{R}^V_{\geq 0}$ in which no subset S of the players has an incentive to deviate from the current allocation. This condition is formally described by the constraint $\sum_{v \in S} x_v \geq \nu(G[S])$ for all subsets $S \subset V$, where $G[S]$ indicates the subgraph of G induced by the vertices S. Such an allocation is called *stable*, and it is well-known (see e.g. [CEW11]) that a stable allocation exists if and only if G is *stable* - that is, if G is a graph whose *inessential* vertices form a stable set, where a vertex is said to be inessential if it is left exposed by at least one maximum matching. We remark here that a second characterization of stable graphs (which we will make use of later), given by Deng et al. [DINZ00], is in terms of *fractional matchings*. A fractional matching is a relaxation of a matching in which we assign fractional values between 0 and 1 to each edge, in such a way that the sum of the values assigned to the edges incident to any vertex does not exceed 1. The value of a maximum fractional matching on G is denoted $\nu_f(G)$. It is clear that $\nu(G) \leq \nu_f(G)$, but in fact, [DINZ00] shows G is stable if and only if $\nu(G) = \nu_f(G)$ (i.e. the inequality is tight). This characterization, based on linear programming, becomes more useful than the former when considering more general settings.

In a network bargaining game, an instance is described by a graph G, and each edge represents a potential deal of unit value that can be made between players. Here, a solution is given by a matching M of G (representing the deals that the players made) together with an allocation $x \in \mathbb{R}^V_{\geq 0}$ (representing how the players chose to split the deals of M). Kleinberg and Tardos [KT08] gave a definition of a *stable* solution in this setting in terms of an *outside option*. A player's outside option refers to the maximum profit a player

can receive by abandoning their current contract and forming a new deal with a different neighbor, under the condition that the new deal benefits both players, provided that such deals exist (if not, the outside option is defined to be 0). In particular, it must benefit the new neighbor with which the new contract is being formed, as this player may have also been engaged in a previous deal, which would have been terminated upon the formation of the new deal. Then a solution (M, x) is stable if each player's allocation is at least as large as their outside option (thus implying that they have no incentive to deviate). They further develop the notion of a *balanced* solution, in which the solution is both stable and 'fair'. The authors prove that for any given instance G of a network bargaining game, a balanced solution exists if and only if a stable solution exists, and this latter condition occurs if and only if G is stable. (The authors go on to prove this result in the more general setting of weighted edges, which we will remark upon momentarily.)

In this sense, the stability of G is a fundamental characterization of the existence of stable solutions of both cooperative matching games and network bargaining games. As a result of this characterization, it is easy to see that there are graphs which do not admit stable solutions (to either type of game), such as K_3, the complete graph on three vertices. Given that not all graphs are stable, a natural problem of how to algorithmically *stabilize* a given graph arises. One stabilizes a graph by making minimal modifications to it, such that the modified graph is stable. There are various types of modifications one might consider, but two of the most natural are that of *edge-removal* and *vertex-removal* (corresponding to blocking certain possible deals, or players, respectively). Formally, a subset of edges $F \subset E$ is called an *edge-stabilizer* if the graph $G \setminus F := (V, E \setminus F)$ is stable, and analogously, a set $S \subset V$ is a *vertex-stabilizer* if $G \setminus S := G[V \setminus S]$ is stable. In [BCK+15], the authors consider edge-removal, where they show that in general this is an NP-hard problem. In contrast, [AHS18] and [IKK+17] concurrently showed that stabilizing the graph by removing vertices can be done in polynomial time. Further, [AHS18] showed that the problem of blocking as few players as possible so as to make a given set of deals realizable as a stable outcome is also polynomial-time solvable, which is again in contrast with the edge-removal setting, in which [BCK+15] showed the analogous problem is NP-hard. Given these results, researchers began considering generalizations of the initial setting.

A natural generalization of graph-stabilization problems can be made by allowing for edge weights $w \in \mathbb{R}^E_{\geq 0}$. In this case, we generalize $\nu(G)$ and $\nu_f(G)$ in terms of *maximum-weight* matchings. Here, $\nu(G)$ denotes the value of a maximum-weight matching in G, and $\nu_f(G)$ can be naturally defined as the optimal value of the following linear program:

$$\nu_f(G) := \max\{w^T x : x(\delta(v)) \leq 1 \; \forall v \in V, x \geq 0\}$$

where $\delta(v)$ is the set of edges in the graph incident with vertex v. As remarked above, [KT08] generalizes to this edge-weighted setting, and en route to their generalized result, they proved that (G, w) admits a stable solution if and only if this linear program has an integral optimal solution, i.e. if $\nu(G) = \nu_f(G)$. This naturally generalizes the analogous result given by [DINZ00].

The topic of edge-stabilization in this case is immediately seen to be computationally hard, as it was already shown to be NP-hard in the unit-weight setting. However, the vertex-stabilization problems can be reconsidered in this more general setting, and in fact it was shown in [KS20] that the following results hold: (i) the problem of finding a minimum cardinality vertex stabilizer remains polynomial time solvable, and (ii) the problem of finding a minimum cardinality vertex stabilizer which realizes a given matching M as a stable outcome is polynomial-time solvable, provided that M is maximum weight, and otherwise the problem is at least vertex-cover-hard.

We can generalize the problem even further by allowing for vertex capacities $c \in \mathbb{Z}_+^V$, in addition to edge weights $w \in \mathbb{R}_+^E$, which is the setting considered in this paper. We further generalize the notion of a matching to that of a *c-matching*, where each vertex $v \in V$ is matched with at most c_v vertices. In this case, the value of a maximum weight c-matching of a graph G is given by $\nu^c(G)$, and the value of a maximum weight fractional c-matching is denoted $\nu_f^c(G)$. Formally, we define

$$\nu_f^c(G) := \max\{w^T x : x(\delta(v)) \leq c_v \; \forall v \in V, 0 \leq x \leq 1\}$$

Such a graph is said to be *stable* if it admits a stable solution to the associated network bargaining game. Here a solution to the network bargaining game is a c-matching M, together with a vector $z \in \mathbb{R}_{\geq 0}^{2E}$, where M indicates which deals have been chosen by the players, and z specifies how the value of each deal value was divided between players. To constitute a solution, we require that if $uv \in M$, then $z_{uv} + z_{vu} = w_{uv}$, and otherwise $z_{uv} = z_{vu} = 0$.

Again we might ask if the natural generalizations of the linear programming characterization of stable graphs persists (i.e. if $\nu^c(G) = \nu_f^c(G)$ exactly for stable graphs). This was answered affirmatively by Bayati et al. [BBC+15], showing that a graph admits a stable solution if and only if $\nu^c(G) = \nu_f^c(G)$.

The pattern of which questions to pose in these successively general settings is becoming clear: we now ask whether we can stabilize graphs with edge weights and vertex capacities by finding a minimum-cardinality vertex-stabilizer in polynomial-time, both in the case when a c-matching M is given, and when it is not. These are the questions we address in

this paper, and accordingly we formalize the following problems:

M-stabilizer problem (c-matching M given): *Given a graph $G = (V, E)$ with edge weights $w \in \mathbb{R}_+^E$ and vertex capacities $c \in \mathbb{Z}_+^V$, together with a maximum weight c-matching M, find a minimum cardinality vertex-stabilizer that realizes M as a stable solution.*
Vertex-stabilizer problem (no matching given): *Given a graph $G = (V, E)$ with edge weights $w \in \mathbb{R}_+^E$ and vertex capacities $c \in \mathbb{Z}_+^V$, find a minimum cardinality vertex stabilizer.*

In this paper, we consider the question of whether or not each of these problems can be solved in polynomial time.

Our results and methods. Our main result is providing a polynomial time algorithm for finding a minimum cardinality vertex-stabilizer which avoids M in the case when a maximum weight c-matching M is already provided, or determining that none exists. This is in contrast with our secondary result, which shows that in general, determining a minimum cardinality vertex stabilizer when no c-matching is given is NP-complete. Specifically, we show the following results:

`RESULT 1`. *There exists a polynomial time algorithm which computes a minimum cardinality M-stabilizer when given an instance of a network bargaining game along with a maximum weight c-matching M, or determining that none exists.*
`RESULT 2`. *The problem of finding a minimum cardinality vertex stabilizer when given an instance of a network bargaining game is NP-complete. Furthermore, no $(2 - \epsilon)$-approximation of this problem exists for any $\epsilon > 0$, assuming the Unique Games Conjecture.*

The fundamental distinction between the two results is whether or not we have access to a maximum-weight c-matching which we are attempting to avoid. It is worth noting that if no such matching is given, then the problem remains hard even in the special case of unit weights. The general outline of the proofs is as follows:
To develop the polynomial time algorithm when M is given, we employ an auxiliary construction given in [FGK13]. Specifically, it is shown in their paper that a graph with edge weights and vertex capacities admits a stable solution if and only if some auxiliary graph does as well. The auxiliary graph is an edge-weighted unit capacity graph, and is thus amenable to the techniques introduced in [KS20]. In particular, a result in [KS20] provides a polynomial time algorithm for finding a minimum cardinality vertex stabilizer that avoids a given matching for such a graph, provided that the matching in question is of maximum weight. In our case, the matching on the auxiliary graph is derived from the c-matching of the original graph, and as such, it need not be maximum weight, even

if the c-matching was. Consequently, their algorithm does not immediately extend to our setting. In particular, in some cases their algorithm can find a pair of vertices for which at least one of which need be removed, yet it is unable to distinguish which to choose. In our setting, the graph will be an auxiliary graph, and as such, we are able to efficiently resolve this potential ambiguity, and delete only the minimum necessary number of vertices. Each of the vertices deleted from the auxiliary graph corresponds to some vertex in the original graph. It is both necessary and sufficient to delete these vertices from the original graph so as to stabilize it, provided it is possible to do so. As a result, this process provides a polynomial time algorithm for finding a minimum vertex stabilizer of the original graph that avoids the given c-matching. With regards to determining whether or not it is possible to remove a vertex or not, we remark here that while a vertex may avoid the matching in the auxiliary graph, the corresponding vertex in the original graph may not avoid the c-matching. However, it is necessary to remove the vertex to stabilize the graph, which simply demonstrates that not all graphs can be stabilized.

With regards to the second result, we demonstrate NP-completeness by reducing from the NP-complete problem of vertex-cover. In particular, we do not make use of edge weights, thus demonstrating that this result holds even when the edges are unit weight. This is again in contrast with the situation when there are also unit vertex capacities.

We summarize the complexity of the various results previously discussed for context in the following table:

(G, w, c)	$w = 1$, $c = 1$	$w \geq 0$, $c = 1$	$w = 1$, $c \geq 0$	$w \geq 0$, $c \geq 0$
Stabilizer Author(s)	poly-time [AHS18] [IKK+17]	poly-time [KS20]	NP-hard [this work]	NP-hard [this work]
M-stabilizer Author(s)	poly-time [AHS18]	poly-time [KS20]	poly-time [this work]	poly-time [this work]

Chapter 2

Related Works

Kleinberg and Tardos [KT08] develop the notion of network bargaining games for weighted graphs with unit vertex-capacity. They introduce the notion of stable solutions for such games, and show that an instance will admit a stable solution if and only if the associated matching LP has an integral optimal solution. They also introduce the notion of balanced solutions, which are a subset of stable solutions, and show that an instance of a network bargaining game has a balanced solution if and only if it has a stable solution. They also develop a polynomial time algorithm for computing all balanced solutions for any given instance.

König-Egerváry graphs (König graphs) are graphs G for which $\nu(G) = \tau(G)$, where $\tau(G)$ is the size of a minimum cardinality *vertex cover* of G. They comprise a fundamental class of graphs which are stable. Notably, the property of being a König graph is not hereditary under vertex/edge deletion, nor under vertex/edge induced subgraphs. This motivates two natural questions: KÖNIG VERTEX (EDGE) DELETION and VERTEX (EDGE) INDUCED KÖNIG SUBGRAPH. Given a graph G and non-negative integer k, the former question asks whether there exists at most k vertices (edges) whose deletion from G results in a König subgraph. The latter asks if there are at least k vertices (edges) in G which induce a König subgraph. [MRS+11] demonstrates the *fixed-parameter tractability* of KÖNIG VERTEX DELETION by reducing to another fixed-parameter tractable problem. They show that the problem admits an $O(\log n \log \log n)$-approximation, and does not admit a constant-factor approximation unless the Unique Games Conjecture (UGC) is false. The authors also study the INDUCED KÖNIG SUBGRAPH problems, in both the vertex and edge cases. They show that VERTEX INDUCED KÖNIG SUBGRAPH is unlikely to be fixed-parameter tractable (it is $W[1]$-hard), and it is inapproximable within a factor of

$O(n^{1-\epsilon})$ for any $\epsilon > 0$. Finally, they show that EDGE INDUCED KÖNIG SUBGRAPH is NP-complete, has a constant-factor approximation, and is also fixed-parameter tractable.

The *core* of a cooperative matching game is the set of payoff vectors $x \in \mathbb{R}^V_{\geq 0}$ which constitute stable allocations. [BKP12] studies solutions for matching games by first giving a new characterization of the core of matching games. They go on to present an $O(nm + n^2 log\ n)$-algorithm which tests if the core of the matching game on an n-vertex, m-edge graph is non-empty, and computes a core allocation if one exists. They further show that the *nucleolus* of an n-player matching game with non-empty core can be computed in $O(n^4)$-time. Lastly, when given an allocation $x \in \mathbb{R}^V$, a pair of adjacent vertices $\{u, v\}$ is said to be a *blocking pair* with respect to x if $x_u + x_v < w_{uv}$, and their blocking value is $w_{uv} - (x_u + x_v)$. $B(x)$ is the set of all blocking pairs with respect to x, and $b(x)$ is the sum of all blocking values. Then the BLOCKING PAIRS problem is that of determining for a given matching game and integer $k \geq 0$ whether the game admits an allocation $x \in \mathbb{R}^V_{\geq 0}$ with $|B(x)| \leq k$ and the BLOCKING VALUE problem similarly asks whether we can find an allocation $x \in \mathbb{R}^V_{\geq 0}$ such that $b(x) \leq k$, for a given rational number k. The authors here show that BLOCKING PAIRS is NP-complete, even for unit-weight games, while BLOCKING VALUE is polynomial time solvable for general matching games.

If instead of allocations $x \in \mathbb{R}^V$, we consider *matching payoffs* (M, p), where M is a matching and $p \in \mathbb{R}^V$ satisfies $p_u + p_v = w_{uv}$ for all edges $uv \in M$, and $p_u = 0$ for all unmatched $u \in V$, then we can analogously define the notion of blocking pairs and blocking values in this context. We may also pose the analogous BLOCKING PAIRS and BLOCKING VALUE problems. [BBG+14] shows that both BLOCKING PAIRS and BLOCKING VALUE are NP-complete in this case. The former is consistent with the result for BLOCKING PAIRS in the case of allocations, whereas the latter surprisingly diverges from its allocation-analogue. However, if we modify the problem by additionally assigning a matching M with respect to which we wish to find our payoff p, then the authors show that BLOCKING VALUE does become polynomial-time solvable, while BLOCKING PAIRS again remains NP-complete. They also show that any unstable solution (M, p) for a weighted n-vertex graph which admits stable solutions has a *path to stability* of length at most $2n$.

Given an allocation $x \in \mathbb{R}^V_{\geq 0}$, a blocking set of x is any subset of edges which contains all blocking pairs. Rephrasing a result of [BKP12] with this terminology, the authors there show that finding a minimum blocking set is NP-hard. Könemann et al. complement this result in [KLS15] by showing that there is an efficient $(8\omega + 2)$-approximation for computing blocking sets in a graph G which is ω-sparse (a graph is ω-sparse for some $\omega \geq 1$ if $|E(G[S])| \leq \omega|S|$ for all $S \subseteq V$).

Farczadi et al. [FGK13] study network bargaining games with general capacities. They show that an instance of such a bargaining game will have a balanced solution if and only if it has a stable solution. Further, they show that there exists a polynomial time algorithm which, when given an instance together with a maximum weight c-matching M, will compute a balanced solution (M, z) whenever one exists. They achieve these results by making of use of a reduction to an auxiliary graph, which has unit-capacities, and demonstrating an equivalence of stable solutions between the two graphs. This correspondence enables them to make use of pre-existing results regarding stable solutions on unit-capacity graphs (see [KT08]), and extend them into the general capacity setting.

Given an unweighted graph G, we may consider the associated edge stabilizer problems of finding either a minimum cardinality edge stabilizer, or a minimum cardinality edge M-stabilizer (which is an edge stabilizer having the additional property of being disjoint from M), where M is a maximum matching. Bock et al. [BCK+15] show that, given a graph G, deleting a minimum edge stabilizer does not decrease the size of a maximum matching (i.e. $\nu(G \setminus F) = \nu(G)$ for any minimum stabilizer F). This immediately motivates the edge M-stabilizer problem. The authors show that the edge M-stabilizer problem is NP-hard, and admits an efficient 2-approximation algorithm. Further, they show that no efficient $(2-\epsilon)$-approximation exists for any $\epsilon > 0$, assuming UGC. Turning to the edge stabilizer problem, they show that there exists an efficient $O(\omega)$-approximation, where $\omega(G)$ is the *sparsity* of G (i.e. the smallest ω such that G is ω-sparse). While they make no conclusions regarding constant-factor approximations for general graphs, they show there exists an efficient 2-approximation in *regular graphs*. Extending the complexity results of the edge M-stabilizer problem, they show that the edge stabilizer problem is NP-hard, and further that no efficient $(2-\epsilon)$-approximation exists for any $\epsilon > 0$, assuming UGC.

Turning briefly to the notion of additive stabilizers, Chandrasekaran et al. [CGK+19] studies the *minimum fractional additive stabilizer problem* (MFASP). Given a unit weight graph, a *fractional additive stabilizer* is a vector $c \in \mathbb{R}^E_{\geq 0}$ such that $(G, \mathbb{1} + c)$ is stable. Then MFASP is the problem of, given a unit weight graph G, finding a fractional additive stabilizer such that $\mathbb{1}^T c$ is minimized. While no polynomial-time approximation for MFASP is known to exist, the authors show that a $o(|V|^{\frac{1}{24}})$-approximation would imply a $o(|V|^{\frac{1}{4}})$-approximation for the so-called *densest k-subgraph* problem (yet the best known performance guarantee of any approximation algorithm for densest k-subgraph is $\approx O(|V|^{\frac{1}{4}})$). They also show that MFASP cannot admit a $o(\log|V|)$-approximation unless P = NP. On the positive side, they show that there is a polynomial time algorithm to solve MFASP in *factor critical* graphs (those in which every subgraph induced by $|V|-1$ vertices has a perfect matching). They also show MFASP can be solved for general graphs G in time $O(2^{\gamma(G)} poly(|V|))$, where $\gamma(G)$ is the size of the *Tutte set* of G. In this way, MFASP

can be seen to be fixed-parameter tractable, where $\gamma(G)$ is the parameter.

[AHS18] focuses on blocking players in cooperative matching games and network bargaining games in such a way that the network formed by the remaining participants admits a stable outcome. Since stable graphs are precisely the graphs which admit a stable outcome (for both types of games), the formal problem becomes: given a graph G, find a minimum cardinality set of vertices whose removal results in a stable graph. This problem can also be generalized to the weighted setting, where each vertex is given some non-negative weight. The authors provide a polynomial-time algorithm to find a minimum cardinality vertex stabilizer in the unit-weight case. They go on to show that the weighted setting becomes NP-hard even when there are only two distinct weights. Consequently, they provide efficient approximation algorithms for some weighted variants. They consider two problems: (i) min-weight vertex-stabilizer, and (ii) max weight vertex-stabilizer. In each problem, we are given a graph $G = (V, E)$ and vertex weights $w_v \geq 0 \ \forall v \in V$. The goal is to find a vertex-stabilizer $S \subset V$ which in the case of (i) minimizes $w(S)$, and in the case of (ii) maximizes the weight of $w(V \setminus S)$. They give a $O(\gamma)$-approximation for (i), and a 2-approximation in the case of (ii). As the algorithms are LP-based, they also consider the integrality gap of the relaxations they use, and show that their analyis are almost tight.

[IKK+17] studies a variety of network modifications which can be made, with the goal of stabilizing a given cooperative matching game. It was previously shown in [KT08] that the associated network bargaining game admits stable solutions precisely when the core is non-empty. This motivates the techniques used for the stabilization processes considered in this paper. There are four natural ways to modify the underlying graph: (i) edge removal, (ii) edge addition, (iii) vertex removal, (iv) vertex addition. In the case of (iv), we are also allowed to add edges from a new vertex to any existing vertices. The authors in [BCK+15] already showed that edge removal is NP-hard, even in the unweighted setting. This paper considers (ii)-(iv) in the unweighted setting, and (ii)-(iii) in the weighted setting. In the unweighted case, it is shown that each of (ii)-(iv) are solvable in polynomial time. In contrast, (ii)-(iii) are shown to be NP-hard in the weighted setting. Further, it is demonstrated that in the unweighted case of (iv), their algorithm will surprisingly add a minimum number of edges among all possible stable solutions.

All positive edge and vertex stabilization results mentioned above apply specifically to the unweighted setting. In the edge weighted setting, Koh and Sanita [KS20] show that a minimum cardinality vertex stabilizer can be efficiently computed for any graph. They provide a polynomial-time algorithm to find a minimum cardinality vertex stabilizer for any weighted graph. The fundamental tool they develop while working towards this result is the creation of a polynomial-time algorithm which computes a *basic fractional matching*

with a minimum number of odd cycles in its support, where a basic fractional matching is a basic feasible solution to the fractional matching LP. Given that finding a minimum cardinality edge stabilizer is NP-hard (see [BCK+15]), the authors here show that the problem of finding a minimum cardinality edge stabilizer in a weighted graph G does not admit any constant factor approximation, unless $P = NP$. They then develop an $O(\Delta)$-approximation, where $\Delta(G)$ is the maximum degree of a vertex in G. Turning to M-vertex stabilizers, they show that, assuming M is of maximum weight, an M-vertex stabilizer can always be found in polynomial time. If instead M is allowed to be an arbitrary matching, they show the problem becomes NP-hard, but admits an efficient 2-approximation algorithm. Further, they show no efficient $(2 - \epsilon)$-approximation exists for any $\epsilon > 0$, assuming UGC.

Chapter 3

Preliminaries

The fundamental object of our concern will be network bargaining games.

Definition 1. An instance of a *network bargaining game* (NBG) is a triple of data (G, w, c), where $G = (V, E)$ is a graph with vertex set V and edge set E, $w \in \mathbb{R}_+^E$ is a vector of edge weights, and $c \in \mathbb{Z}_+^V$ is a vector of vertex capacities. We say the instance is *unit-capacity* if $c_v = 1 \ \forall v \in V$.

We use the notation $\delta(u)$ to refer to the set of edges incident to vertex u, for each $u \in V$. Given a subset of edges $F \subset E$, let $\delta^F(u) := F \cap \delta(u)$, and $d_u^F := |\delta^F(u)|$, the number of edges of F incident to u, for each $u \in V$. Then we can define a subset of edges $M \subset E$ to be a *c-matching* in G if $d_u^M \leq c_u$ for every vertex $u \in V$. In other words, each vertex can be matched with at most c_u of its neighbors. Given a c-matching M, we call d_u^M the *M-degree* of u, and we say u is *M-saturated* if $d_u^M = c_u$. We simply use the term *saturated* if M is understood. We also define $M(u) := \{v \in V : uv \in M\}$. Clearly we have $|M(u)| = d_u^M$. Additionally, if $y \in \mathbb{R}^E$ is any vector, define $y(F) = \sum_{e \in F} y_e$.

Definition 2. A *solution* to the network bargaining game (G, w, c) is a pair (M, z), where M is a c-matching, and $z \in \mathbb{R}_+^{2E}$ is a vector which assigns each edge uv a pair of values z_{uv} and z_{vu}, such that $z_{uv} + z_{vu} = w_{uv}$ if $uv \in M$, and $z_{uv} = z_{vu} = 0$ otherwise.

Intuitively, we interpret z_{uv} (z_{vu}) to be the profit that player u (v) receives from engaging in deal $uv \in M$. Alternatively, if $uv \in E \backslash M$, then neither player derives any profit from this edge. The *allocation* of the solution (M, z) is a vector $x \in \mathbb{R}_+^V$, where $x_u := \sum_{v:uv \in M} z_{uv}$, which denotes the total value u receives from all deals they are engaged in. Typically, we are concerned with whether a player is content with their current allocation, or if they

are instead able to obtain a greater allocation by altering their current set of deals. For instance, if adjacent players u and v are both not saturated, then trivially both would prefer a different solution, as, in particular, u and v would prefer to engage in a deal with each other, in addition to the rest of the deals already specified by the given solution. Further, even if u is saturated, it is possible u might still prefer a different solution overall. Accordingly, the notion of 'outside option' captures whether or not a player is content with the current solution.

Definition 3. The *outside option* of vertex u with respect to a solution (M, z) is

$$\alpha_u(M, z) := \max\Big(0, \max_{v:uv \in E \setminus M} \Big(w_{uv} - \mathbf{1}_{[d_v^M = c_v]} \min_{w:vw \in M} z_{vw}\Big)\Big),$$

where $\mathbf{1}_P$ is the indicator function for the statement P, which evaluates to 1 if P is true and 0 otherwise.

Player u examines all neighbors v he is not currently in a deal with, and considers the greatest value he could extract by engaging in a deal with v, worth a total of w_{uv}. If v is not saturated, then v is willing to engage in deal uv without extracting any profit, as they fare no worse than their current situation, and u would derive the entire value w_{uv} of the edge uv. On the other hand, if v is saturated, then u must entice v to engage in the deal, and does so by offering the same value as that of the least valuable allotment z_{vw} that v is currently extracting from a deal vw which v is engaged in. This in turn means u will only take allotment $w_{uv} - z_{vw}$, assuming such a deal is legal (i.e. $w_{uv} - z_{vw}$ is non-negative). Each player is content only when engaging in a new deal which realizes their outside option (possibly at the expense of a deal in which they are currently engaged) provides them with no greater personal allocation than that which they currently have.

Definition 4. A solution (M, z) is *stable* if for all $uv \in M$ we have $z_{uv} \geq \alpha_u(M, z)$ and $z_{vu} \geq \alpha_v(M, z)$, and for all unsaturated vertices u we have $\alpha_u(M, z) = 0$. Further, (M, z) is said to be *balanced* if in addition to being stable, it has the property that $z_{uv} - \alpha_u(M, z) = z_{vu} - \alpha_v(M, z)$ for every $uv \in M$.

We say that the NBG instance (G, w, c) is *stable* if it admits a stable solution, and otherwise we say it is *unstable*. See Figure 3.1 for an example of a NBG together with an unstable solution.

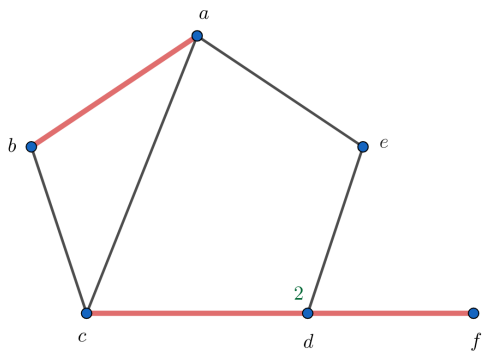

Figure 3.1: An NBG with solution (M, z). Here, all edges have unit weight, and all capacities are 1 except for d, which has capacity 2. M is given by the bold red edges, and z is given by $z_{uv} = z_{vu} = \frac{1}{2}$ if $uv \in M$, and 0 otherwise. It is easy to see this solution is not stable, since (for example) the outside option of e is $\frac{1}{2} > 0$, yet e is not M-saturated.

We can more cleanly characterize stable solutions by introducing the notation $\gamma_u(M, z)$ for a vertex u and solution (M, z), where

$$\gamma_u(M, z) := \begin{cases} \min\limits_{v \in M(u)} z_{uv} & \text{if } d_u^M = c_u \\ 0 & \text{if } d_u^M < c_u \end{cases}$$

We will frequently abbreviate $\gamma_u(M, z)$ by γ_u when the solution (M, z) is clear from context. Similarly with α_u for $\alpha_u(M, z)$. We now claim we can characterize stability as follows:

Claim 5. A solution (M, z) is stable if and only if $\gamma_u + \gamma_v \geq w_{uv}\ \forall uv \in E \setminus M$.

Proof.
$(\implies)$
Suppose $\gamma_u + \gamma_v < w_{uv}$ for some $uv \in E \setminus M$. Then $\gamma_u < w_{uv} - \gamma_v$, and so $\gamma_u < w_{uv} - \gamma_v \leq \alpha_u$, implying that u is currently engaged in a deal worth less than its outside option. This means (M, z) is unstable.
$(\impliedby)$
Suppose there is $uv \in M$ such that $z_{uv} < \alpha_u$. In particular, $\alpha_u > 0$. Then let $v' = \text{argmax}(\alpha_u)$, so $\alpha_u = w_{uv'} - \mathbf{1}_{d_{v'}^M = c_{v'}} \min\limits_{w : v'w \in M} z_{v'w} = w_{uv'} - \gamma_{v'}$. Then $w_{uv'} - \gamma_{v'} > z_{uv} \geq \gamma_u$, so $\gamma_u + \gamma_{v'} < w_{uv'}$. Since $uv' \notin M$, we see that the RHS of the bi-conditional does not hold either. $\square$

3.1 Linear programming characterization

The primary characterization of stable instances of NBGs we make use of is in terms of linear programming duality. We follow the method of [BHIM10] and [BBC+15], and introduce the following linear program, (P^c_{FM}) (with dual (D^c_{FM})), whose solutions are called *fractional c-matchings*:

$$\begin{aligned} \max \sum_{e\in E} w_e x_e \\ \text{s.t. } x(\delta(v)) \le c_v \quad \forall v \in V \\ 0 \le x \le 1 \\ (P^c_F M) \end{aligned} \qquad \begin{aligned} \min \sum_{v\in V} c_v y_v + \sum_{uv\in E} y_{uv} \\ \text{s.t. } y_u + y_v + y_{uv} \ge w_{uv} \quad \forall uv \in E \\ y \ge 0 \\ (D^c_{FM}) \end{aligned}$$

Let $\nu^c_f(G) := \mathrm{OPT}(P^c_{FM})$, and let $\nu^c(G)$ denote the total weight of a maximum weight c-matching in G, i.e. the value of an optimal integral solution to P^c_{FM}. It is trivially true that $\nu^c(G) \le \nu^c_f(G)$. The following lemma given by [BBC+15] characterizes stability of an NBG (G, w, c) in terms of tightness (or lack thereof) of this inequality.

Lemma 6. ([BBC+15]) (G, w, c) has a stable solution if and only if $\nu^c(G) = \nu^c_f(G)$. Additionally, if (M, z) is a stable solution, then it has optimal dual solution $y_v = \gamma_v$ and

$$y_{uv} = \begin{cases} w_{uv} - y_u - y_v & \text{if } uv \in M \\ 0 & \text{if } uv \notin M \end{cases}$$

Proof. We first record the complementary slackness conditions for future reference. In this case, the c.s. conditions are

$$\begin{aligned} &\text{(i)} && x_{uv}(y_u + y_v + y_{uv} - w_{uv}) = 0 \\ &\text{(ii)} && y_{uv}(x_{uv} - 1) = 0 \\ &\text{(iii)} && y_v(x(\delta(v)) - c_v) = 0 \end{aligned}$$

($\implies$)
Let (M, z) be a stable solution to (G, w, c). Then take x^M to be the characteristic vector of M, which is of course feasible for (P^c_{FM}). To demonstrate optimality (thus demonstrating (P^c_{FM}) has an integral optimal solution), we will find a feasible dual solution y^* that has the same objective value. Specifically, for each $u \in V$, take $y^*_u = \gamma_u$, and

$$y^*_{uv} = \begin{cases} w_{uv} - y^*_u - y^*_v & \text{if } uv \in M \\ 0 & \text{if } uv \in E \setminus M \end{cases}$$

This solution is seen to be feasible: for $uv \in M$, we have $y^*_u + y^*_v + y^*_{uv} = w_{uv}$. If $uv \in E \setminus M$, then $y^*_u + y^*_v + y^*_{uv} = \gamma_u + \gamma_v \geq w_{uv}$, by stability of (M, z) and by Claim 5. Thus y^* satisfies the first dual constraint. Now note that $\gamma_u \geq 0$ for all vertices u, so $y^*_u \geq 0$ and $y^*_{uv} \geq 0$ for all $uv \in E \setminus M$ (again by definition). Thus feasibility is reduced to confirming $y^*_{uv} \geq 0$ for $uv \in M$. But this is indeed the case, since $\gamma_u \leq z_{uv}$ and $\gamma_v \leq z_{vu}$, and so $\gamma_u + \gamma_v \leq z_{uv} + z_{vu} = w_{uv}$. Thus $y^*_{uv} = w_{uv} - \gamma_u - \gamma_v \geq 0$. Then the dual objective $\sum_{v \in V} c_v y^*_v + \sum_{uv \in E} y^*_{uv} = \sum_{v \,:\, d^M_v = c_v} c_v y^*_v + \sum_{uv \in M} (w_{uv} - y^*_u - y^*_v) = \sum_{uv \in M} w_{uv}$ is equal to the weight of M (which is the value of the primal), thus demonstrating that the primal integral solution x^M is optimal. Note that this also proves the second statement of the lemma.

($\Longleftarrow$)

Let x^* be an integral optimal solution to P^c_{FM}, and let M be the corresponding c-matching. Take y^* to be any optimal dual solution. Then the c.s. conditions in this case specialize

as (i) $y^*_u + y^*_v + y^*_{uv} = w_{uv}\ \forall uv \in M$
(ii) $y^*_{uv} = 0\ \forall uv \in E \setminus M$
(iii) $y^*_v = 0\ \forall v$ s.t. $d^M_v < c_v$

Then we can take z so that

$$z_{uv} = \begin{cases} y^*_u + \frac{y^*_{uv}}{2} & \text{if } uv \in M \\ 0 & \text{if } uv \notin M \end{cases}$$

Observe that z satisfies the constraints $z_{uv} = z_{vu} = 0\ \forall uv \notin M$ and $z_{uv} + z_{vu} = w_{uv}\ \forall uv \in M$ by definition, and by (i). Therefore (M, z) constitutes a solution. Now to confirm stability we must check that $\gamma_u + \gamma_v \geq w_{uv}$ for all $uv \notin M$. First, we can observe that $\gamma_u \geq y^*_u$ for every vertex u: for every unsaturated vertex u, we have that $\gamma_u = y^*_u = 0$, and for a saturated vertex u, we have $\gamma_u = z_{ut} = y^*_u + \frac{y^*_{ut}}{2}$ for some $t \in M(u)$, and this quantity is clearly at least y^*_u, since $y^*_{uv} \geq 0$ by feasibility of y^*. So if $uv \notin M$, then $\gamma_u + \gamma_v \geq y^*_u + y^*_v = y^*_u + y^*_v + y^*_{uv} \geq w_{uv}$ by (ii), and since y^* is feasible. Therefore, by Claim 5, (M, z) is a stable solution, and (G, w, c) is stable. □

By this lemma, we immediately obtain the following corollary.

Corollary 7. If (M, z) is stable, then M is a maximum weight c-matching.

Proof. Given a stable solution (M, z), we have by the above lemma that the characteristic vector x^M is an optimal solution to (P^c_{FM}). Since x^M is integral, then M is of maximum weight. □

3.2 Augmenting walks

Given a NBG (G, w, c), we denote a walk in the graph by $W = (u; e_1, \ldots, e_k; v)$, where $u \in V$ indicates the initial vertex where we begin the walk, $v \in V$ is the vertex at which the walk terminates, and $(e_1, \ldots, e_k)$ is a sequence edges to be traversed in the indexed order. We may also say call W a *uv-walk*. It is also equivalent to simply list the sequence of edges, since this determines the walk. We will also define $W^{-1} = (v; e_k, \ldots, e_1; u)$.
Given a NBG (G, w, c) with c-matching M, we say that a walk W is *M-alternating* if the edges alternate between edges belonging to M and $E \setminus M$. W is an *M-augmenting* walk if it is M-alternating and $w(W \setminus M) > w(W \cap M)$. An M-augmenting walk W is *proper* if $E(W)\Delta M$ is a c-matching, where $A\Delta B := A \setminus B \cup B \setminus A$ for two sets A and B. We define $W\Delta M = E(W)\Delta M$ for convenience. Four notable special classes of walks are *trails*, *paths*, *circuits*, and *cycles*. A trail is a walk in which no edge is repeated, and a path is a trail in which no vertex is repeated. Similarly, a circuit is a walk which has the same initial and terminal endpoints, and does not repeat an edge, and a cycle is a circuit which also does not repeat any vertices aside from the initial and terminal point. In the case when the NBG is unit-capacity, there are notable notable classes of subgraphs which will prove relevant.

Definition 8. An odd cycle $C = (e_1, \cdots, e_{2k+1})$ is an *M-blossom* if $e_i \in M$ for all even i, and $e_i \notin M$ for all odd i. The unique vertex $v := e_1 \cap e_{2k+1}$ is called the *base* of the blossom. The blossom is *augmenting* if $w(C \setminus M) > w(C \cap M)$. An *$M$-flower* $C \cup P$ is an M-blossom C with base v, and a proper M-alternating path $P = (v; e_1, \ldots, e_k; u)$, where $e_1 \in M$. The vertex u is the *root* of the flower. The flower is *augmenting* if $w(C \setminus M) + 2w(P \setminus M) > w(C \cap M) + 2w(P \cap M)$. An *$M$-bicycle* $C \cup P \cup D$ consists of two M-blossoms C, D with respective bases u and v, and an odd M-alternating path $P = (u; e_1, \ldots, e_k; v)$, where e_1, $e_k \in M$. It is *augmenting* if $w(C \setminus M) + 2w(P \setminus M) + w(D \setminus M) > w(C \cap M) + 2w(P \cap M) + w(D \cap M)$.

We will establish a relationship between augmenting flowers/bi-cycles and augmenting walks in the subsequent chapter, en route to computing a (vertex) stabilizer.

In the case of a unit-capacity graph, it is well-known that a matching on the graph is of maximum weight if and only if there does not exist any proper augmenting path or cycle (augmenting with respect the given matching). In a similar manner, we can generalize this result to the setting in which we allow for general capacities c.

Proposition 9. A c-matching M in (G, w, c) is of maximum weight if and only if (G, w, c) does not contain a proper M-augmenting trail.

Proof.
($\implies$)
If G contains a proper M-augmenting trail T, then $M \Delta T$ is a c-matching such that $w(M \Delta T) > w(M)$, which means M is not maximum weight.

($\impliedby$)
Let M be a c-matching in G that does not have any M-augmenting trails, and suppose by contradiction that M is not maximum. Then let N be a maximum c-matching, and consider the graph induced by edge set $M \Delta N$. There is at least one component H of this graph such that $w(H \cap N) > w(H \cap M)$, for otherwise we would not have that $w(N) > w(M)$. We will now construct a corresponding unit-capacity graph $\hat{H}$ as follows:

- initialize $V(\hat{H}) = \emptyset$, $E(\hat{H}) = \emptyset, \hat{M} = \emptyset, \hat{N} = \emptyset$
- for each vertex $u \in V(H)$:
  - define $b_u := \max\{d_u^{M \setminus N}, d_u^{N \setminus M}\}$
  - create copies $u_1, \ldots, u_{b_u}$ of u, and set the capacity of each to 1
  - $V(\hat{H}) \leftarrow V(\hat{H}) \cup \{u_1, \ldots, u_{b_u}\}$
  - initialize $J_M(b_u) = \{1, \ldots, b_u\}$, $J_N(b_u) = \{1, \ldots, b_u\}$
- for each edge $uv \in E(H)$:
  - if $uv \in M$
    - choose the smallest indices $i \in J_M(b_u)$, $j \in J_M(b_v)$
    - $\hat{M} \leftarrow \hat{M} \cup \{u_i v_j\}$
    - set $w_{u_i v_j} = w_{uv}$
    - $J_M(b_u) \leftarrow J_M(b_u) \setminus \{i\}$, $J_M(b_v) \leftarrow J_M(b_v) \setminus \{j\}$
  - if $uv \in N$
    - choose the smallest indices $i \in J_N(b_u)$, $j \in J_N(b_v)$
    - $\hat{N} \leftarrow \hat{N} \cup \{u_i v_j\}$
    - set $w_{u_i v_j} = w_{uv}$
    - $J_N(b_u) \leftarrow J_N(b_u) \setminus \{i\}$, $J_N(b_v) \leftarrow J_N(b_v) \setminus \{j\}$
- set $E(\hat{H}) = \hat{M} \cup \hat{N}$

Observe that this construction establishes a natural weight-preserving bijection between $E(H)$ and $E(\hat{H})$. Further, the constructed sets $\hat{M}$ and $\hat{N}$ are matchings in $\hat{H}$, where $u_i v_j$ is an edge in $\hat{M}$ if and only if uv is in $M \cap E(H)$, and analogously for $\hat{N}$. Due to the weight-preserving bijection, we know that $w(H \cap N) > w(H \cap M)$ implies $w(\hat{N}) > w(\hat{M})$. Suppose $\hat{H}$ has components $\hat{H}_1, \ldots, \hat{H}_l$. Then since $w(\hat{N}) > w(\hat{M})$, there must be some component $\hat{H}_t$ such that $w(\hat{H}_t \cap \hat{N}) > w(\hat{H}_t \cap \hat{M})$. In particular, $\hat{M}$ is not a maximum weight matching on $\hat{H}_t$. Then since $\hat{H}_t$ has unit capacities, it contains either a proper $\hat{M}$-augmenting path or cycle $\hat{T}$. In either case, we claim $\hat{T}$ corresponds (under the aforementioned natural bijection) to a proper M-augmenting trail $T = (u; e_1, \ldots, e_k; v)$ in H, and hence in G as well. Note that in either case, we already have that $w(\hat{T} \setminus \hat{M}) > w(\hat{T} \cap \hat{M})$, so correspondingly, we have $w(T \setminus M) > w(T \cap M)$. Since $\hat{T}$ does not repeat any edges, then neither does T. Therefore, T is an M-augmenting trail. Thus, we need only show that T is proper, i.e. that $d_u^{T \Delta M} \leq c_u$ for every $u \in V(G)$.

Case 1 $\hat{T}$ is an $\hat{M}$-augmenting path.
Let u and v be the initial and terminal vertices of T. For any vertex $r \in V(G) \setminus \{u, v\}$, the degree is invariant (i.e. $d_r^{T \Delta M} = d_r^M$), either because r is not included in $V(T)$, or if it is, then since T is M-alternating, exactly one of the preceding and succeeding edges of r in $E(T)$ will belong to M. Thus we need only consider the cases of u and v. We must consider two sub-cases, corresponding to whether or not $u = v$. If $u = v$, then provided that at least one of e_1 and e_k are in M, then $d_u^{T \Delta M} \leq d_u^M$. We have that $d_u^M \leq c_u$ since M is a c-matching, so therefore $d_u^{T \Delta M} \leq d_u^M \leq c_u$, and hence $T \Delta M$ is a c-matching. On the other hand, if neither of e_1 or e_k is in M, then the corresponding edges $\hat{e_1}$ and $\hat{e_k}$ in $\hat{H}$ are not in $\hat{M}$, and must therefore be in $\hat{N}$. Let u_i and u_j be the initial and terminal vertices of $\hat{T}$ in $V(\hat{H})$. Note that u_i and u_j must be distinct since we are assuming $\hat{T}$ is a path. The edges $\hat{e}_1$ and $\hat{e}_k$ in $\hat{N}$ are incident to u_i and u_j, respectively. Given that $\hat{T}$ is a proper $\hat{M}$-augmenting path, then u_i and u_j are not incident with edges in $\hat{M}$. Observe that by construction of $\hat{H}$, either every vertex in $\{u_1, \ldots, u_{b_u}\}$ is (a) $\hat{M}$-matched, or (b) $\hat{N}$-matched. Therefore, if there is a vertex in $\{u_1, \ldots, u_{b_u}\}$ that is $\hat{N}$-matched but not $\hat{M}$-matched, then it must be that every vertex in $\{u_1, \ldots, u_{b_u}\}$ is $\hat{N}$-matched. Moreover, since u_i and u_j are distinct vertices in $\{u_1, \ldots, u_{b_u}\}$ which are not $\hat{M}$-matched, then $d_u^{M \setminus N}(u) \leq d_u^{N \setminus M} - 2$. This in turn means $d_u^M \leq d_u^N - 2$, and since N is a c-matching, then $d_u^N \leq c_u$, which yields $d_u^M \leq c_u - 2$. Since $d_u^{T \Delta M} = d_u^M + 2$, then $d_u^{T \Delta M} \leq c_u$, and so $T \Delta M$ is a c-matching. Therefore, T is a proper M-augmenting trail.
On the other hand, if $u \neq v$ then we must again consider whether or not each of $\hat{e}_1$ and $\hat{e}_k$ are in $\hat{M}$. There are four cases, corresponding to a binary choice for whether each of $\hat{e}_1$ and $\hat{e}_k$ are in $\hat{M}$ or not. In all cases, if $\hat{e}_1 \in \hat{M}$, then $d_u^{T \Delta M} = d_u^M - 1$, and if $\hat{e}_1 \notin \hat{M}$, then

$d_u^{T\Delta M} = d_u^T + 1$, and analogously for $d_v^{T\Delta M}$. Since we need only show that $d_u^{T\Delta M}$ and $d_v^{T\Delta M}$ are no greater than c_u and c_v respectively, then it suffices to consider only the cases for which at least one of $\hat{e}_1$ and $\hat{e}_k$ are not in $\hat{M}$. Further, since $u \neq v$, we need only prove that $d_u^{T\Delta M} \leq c_u$, by symmetry of u and v. So assume $\hat{e}_1 \notin \hat{M}$. Then it must be that $\hat{e}_1 \in \hat{N}$. Let u_i be the vertex in $\{u_1, \ldots, u_{b_u}\}$ that is incident with $\hat{e}_1$. Since the initial vertex u_i of $\hat{T}$ is incident with an edge in $\hat{N}$, then u_i cannot be incident with an edge in $\hat{M}$, for otherwise $\hat{T}$ would not be a proper $\hat{M}$-augmenting path. Therefore, $d_u^{N\setminus M} \geq d_u^{M\setminus N} + 1$, which in turn means $d_u^N \geq d_u^M + 1$. Thus we have that $d_u^{T\Delta M} = d_u^M + 1 \leq d_u^N \leq c_u$. This means $T\Delta M$ is a c-matching.

Case 2 $\hat{T}$ is an $\hat{M}$-augmenting cycle.
Let T be the corresponding subgraph of $\hat{T}$ in G. In this case, since no two copies of a vertex in $V(G)$ are adjacent, then T will be an M-augmenting trail for which $d_u^{T\Delta M} = d_u^M$ for every $u \in V(G)$. Therefore, $d_u^{T\Delta M} \leq c_u$ for each $u \in V(G)$, and hence $T\Delta M$ is a c-matching. □

Remark 10. A proper augmenting trail may be decomposed into a sequence of paths and cycles, at least of which must be augmenting. However, it is possible that no constituent path or cycle is both augmenting and proper.

3.3 Auxiliary construction

The primary tool in we will utilize in demonstrating stability will be the auxiliary construction provided by [FGK13]. This will provide a correspondence between a c-matching M on a NBG (G, w, c) and a matching on an auxiliary graph G' which has unit capacities. Given an instance (G, w, c) together with a c-matching M, we construct an auxiliary graph $(G', w', \mathbb{1})$ together with an associated matching M'. We say $[(G', w', \mathbb{1}), M']$ is the *auxiliary* of $[(G, w, c), M]$. In particular M' is called the *auxiliary* matching. Both the cardinality and the weight of M' are invariant with respect to M, i.e. $|M'| = |M|$, and $w'(M') = w(M)$. We also call the map $[(G, w, c), M] \longrightarrow [(G', w', \mathbb{1}), M']$ the *auxiliary map*, denoted by *aux*. We perform the construction - and hence define the auxiliary map - as follows:

Algorithm 1: Auxiliary map aux

Result: Auxiliary graph $(G', w', \mathbb{1})$ with $\mathbb{1}$-matching M'
Input: NBG (G, w, c) with c-matching M;
for $u \in V(G)$ **do**
  Create a set $C_u = \{u_1, u_2, \ldots, u_{c_u}\}$ of c_u copies of u
  Initialize $J(u) = [d_u^M]$
end
-Set $V(G') = \bigcup_{u \in V(G)} C_u$
for $uv \in M$ **do**
  Create a single edge $u_{i^*}v_{j^*}$, where $i^* \in J(u)$ and $j^* \in J(v)$ are chosen arbitrarily
  $J(u) \leftarrow J(u) \setminus \{i^*\}$
  $J(v) \leftarrow J(v) \setminus \{j^*\}$
end
-Set $M' = \bigcup_{uv \in M} \{u_{i^*}v_{j^*}\}$
for $uv \in E(G) \setminus M$ **do**
  Create an edge u_iv_j for every $u_i \in C_u$ and every $v_j \in C_v$
end
-Set $E(G') = M' \cup \bigcup_{uv \in E(G) \setminus M} \{u_iv_j\}$
for $uv \in E(G)$, $u_i \in C_u$, $v_j \in C_v$ **do**
  Set $w'_{u_iv_j} = w_{uv}$
end

In the above construction, we create an independent set of vertices C_u for each $u \in V(G)$. We call C_u the *cloud* of u. If u and v share an unmatched edge in G, then every pair of vertices (u_i, v_j) shares an unmatched edge in G' (in other words, the graph induced by $C_u \cup C_v$ is a complete bipartite graph). If instead u and v share a matched edge in G, then exactly one pair (u_{i^*}, v_{j^*}) will share an edge in G', and this edge will also be matched.

If $[(G', w', \mathbb{1}), M'] = [(G, w, c), M]_{aux}$, then for any vertex or edge of G', there is a unique corresponding vertex or edge in G. We define a map η to capture this correspondence. Specifically, if $u_i \in V(G') \cap C_u$ for some $u \in V(G)$, then $\eta(u_i) := u$, and if $u_iv_j \in E(G')$ such that $u_i \in C_u$ and $v_j \in C_v$ for some $u, v \in V(G)$, then $\eta(u_iv_j) := uv$. We extend the notation so that if $H' \subseteq G'$ is a subgraph, then $\eta(H') := \eta(V(H')) \cup \eta(E(H'))$. Roughly speaking, $\eta(H')$ is the smallest subgraph of G (inclusion-wise) which gives rise to H' under aux. Typically, we will assume $\eta(H')$ comes paired with the associated edge weights and vertex capacities. In particular, if $P' := (u_i; e'_1, e'_2, \ldots, e'_k; v_j)$ is a walk on G',

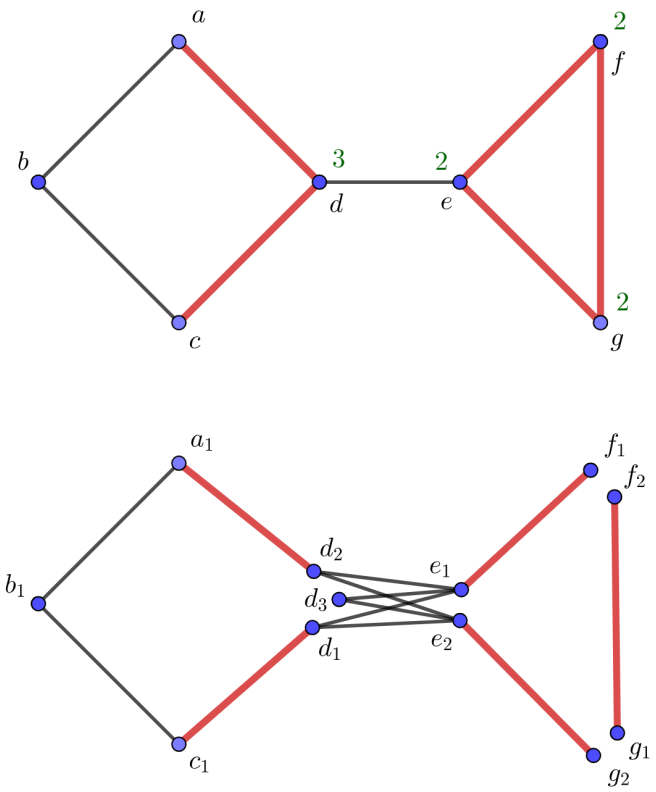

Figure 3.2: A graph and matching$[(G, w, c), M]$ (above), together with its auxiliary $[(G', w'), M']$ (below). Here (for instance) $\sigma_a(d) = 1$, and $\sigma_d(a) = 2$.

then $\eta(P') = (\eta(u_i); \eta(e'_1), \eta(e'_2), \ldots, \eta(e'_k); \eta(v_j))$ is the corresponding walk in G. We will frequently consider a subgraph of G' which admits some type of 'augmenting structure', which will then lead us to examine what type of structure must have existed within G to give rise to the augmentation. Further, for future convenience of notation, define a map

$$\sigma_u : M(u) \longrightarrow [N]$$

for each $u \in V(G)$ such that $|M(u)| \geq 1$, where $N = \max\{c_x : x \in M(u)\}$ as follows: for each $v \in M(u)$, if $u_{i^*}v_{j^*}$ is the unique edge between the clouds C_u and C_v, then $\sigma_u(v) = i^*$ (Similarly, we require that $\sigma_v(u) = j^*$). Observe that with this notation, we have that $u_{\sigma_u(v)}v_{\sigma_v(u)}$ is the unique edge between the clouds C_u and C_v, and it is also in M'. See Figure 3.2 for an example.

Given a solution z to $[(G, w, c), M]$, we can construct a naturally corresponding solution x to $[(G', w', \mathbb{1}), M']$, by letting $x_{u_i} = 0$ if u_i is not matched by M', and alternatively, if u_i is matched by M', say to v_j, we take $x_{u_i} = z_{uv}$, where $u_i \in C_u$ and $v_j \in C_v$. Note that this usage of the variable vector x is consistent with our usage above, where we used x as referring to the allocation of a vertex. Observe that in the case of a unit matching, the allocation of a vertex u_i is equal to the value of the unique deal u_i is engaged in, provided it is engaged in a deal at all (if not, then the allocation is 0, which is still equal to the total value of deals it is engaged in). We can unambiguously assign solution values to vertices, rather than only edges in this unit-capacity case. Note that this correspondence is one-to-one.

This correspondence of solutions is in fact a bijection. Construct the sets of solutions $\mathcal{X}$ and $\mathcal{Z}$, where

$$\mathcal{X} := \{x \in \mathbb{R}^{V(G')} : (M', x) \text{ is a solution to } (G', w', \mathbb{1})\},$$

and

$$\mathcal{Z} := \{z \in \mathbb{R}^{2E(G)} : (M, z) \text{ is a solution to } (G, w, c)\}.$$

The formal correspondence ϕ between solution sets $\mathcal{Z}$ and $\mathcal{X}$ is defined in [FGK13] as follows:

$\phi : \mathcal{X} \longrightarrow \mathcal{Z}$ such that for all $uv \in E(G)$, we have

$$(\phi(x)_{uv}, \phi(x)_{vu}) := \begin{cases} (x_{u_{\sigma_{u(v)}}}, x_{v_{\sigma_v(u)}}) & \text{if } uv \in M \\ (0, 0) & \text{otherwise} \end{cases}$$

$\phi^{-1} : \mathcal{Z} \longrightarrow \mathcal{X}$ such that for all $u_i \in V(G')$, we have

$$\phi^{-1}(z)_{u_i} := \begin{cases} z_{uv} & \text{if } i = \sigma_u(v) \\ 0 \text{ otherwise} \end{cases}$$

Informally, if $x \in \mathcal{X}$ is a solution to $(G', w', \mathbb{1})$, and if players u and v made a deal $uv \in M$, then a particular copy of u in C_u and v in C_v made a corresponding deal in G'. The particular copies of u and v are $u_{\sigma_u(v)}$ and $v_{\sigma_v(u)}$. In G', these values are uniquely associated to these vertices as $x_{u_{\sigma_u(v)}}$ and $x_{v_{\sigma_v(u)}}$. Since this is the unique deal between a copy of u and a copy of v in G', then we may associate the allocated values to the edge uv in G, corresponding to how much each player received in G'. If on the other hand we start with a solution $z \in \mathcal{Z}$, then if a particular deal uv was made in G, there will be a

unique edge $u_{\sigma_{u(v)}}v_{\sigma_v(u)}$ in G' corresponding to uv. For the amount z_{uv} allocated to u in G, allocate the same amount to $u_{\sigma_u(v)}$, and allocate 0 to all other copies of $u_i \in C_u$ (and similarly for $v_{\sigma_v(u)}$).

As shown by [FGK13], the prescribed allocations do in fact constitute genuine solutions. Moreover, they prove that $\mathcal{X}$ and $\mathcal{Z}$ are in a bijective correspondence under the map ϕ. Specifically, they prove that ϕ and ϕ^{-1} are indeed inverse functions, thus justifying the inverse notation, and the claim of a bijection. We provide a lightly modified version of their proof:

Lemma 11. ([FGK13]) The map ϕ is a bijection between $\mathcal{X}$ and $\mathcal{Z}$, and ϕ^{-1} as defined above is the inverse of ϕ.

Proof. It suffices to show that the co-domains of ϕ and ϕ^{-1} specified above are indeed correct (it is not entirely obvious at a glance that the images of solutions are also solutions). It will then follow from the definitions of ϕ and ϕ^{-1} that they are inverse maps, thus proving the lemma. Specifically, we prove the following two statements.

1. $\phi(\mathcal{X}) \subseteq \mathcal{Z}$ and $\phi^{-1}(\mathcal{Z}) \subseteq \mathcal{X}$
2. $z = \phi(x)$ if and only if $x = \phi^{-1}(z)$, i.e. ϕ and ϕ^{-1} are indeed inverses (which justifies the notation).

To see the first statement, let $x \in \mathcal{X}$ and $z = \phi(x)$. To see that $z \in \mathcal{Z}$, first choose an edge $uv \in E(G) \cap M$. Suppose $i = \sigma_u(v)$ and $j = \sigma_v(u)$. By the construction of the auxiliary $[(G', w', \mathbb{1}), M']$, we have that $z_{uv} + z_{vu} = x_{u_i} + x_{v_j} = w_{u_iv_j} = w_{uv}$. If instead $uv \in E(G) \setminus M$, then by definition of $\phi(x)$ we have $z_{uv} = z_{vu} = 0$.

Now let $z \in \mathcal{Z}$ and $x = \phi^{-1}(z)$. To see that $x \in \mathcal{X}$, take $u_iv_j \in E(G') \cap M'$. By the construction of $[(G', w', \mathbb{1}), M']$, there must exist an edge $uv \in E(G) \cap M$ such that $i = \sigma_u(v)$ and $j = \sigma_v(u)$. Then we have that $x_{u_i} + x_{v_j} = z_{uv} + z_{vu} = w_{uv} = w_{u_iv_j}$. If u_i is M'-exposed, then $x_{u_i} = 0$ by definition.

By definition of ϕ and ϕ^{-1}, we have that $\phi^{-1}(\phi(x)) = x$ and $\phi(\phi^{-1}(z)) = z$. The first statement shows that both $\phi^{-1}(\phi(x))$ and $\phi(\phi^{-1}(z))$ are well-defined quantities. Therefore, the second statement is proven. □

Given this bijective correspondence ϕ, we say that (M, z) and (M', x) are *equivalent* if $z = \phi(x)$. We use the notation $(M, z) \cong (M', x)$ to indicate that (M, z) and (M', x) are equivalent. Further, the authors prove the following result:

Theorem 12. ([FGK13]) Let $[(G, w, c), M]$ be a NBG together with a c-matching M, and let $[(G', w', \mathbb{1}), M']$ be the auxiliary. Then there exists a stable solution for $[(G, w, c), M]$ if and only if there exists a stable solution for $[(G', w', \mathbb{1}), M']$. Moreover, if $(M, z) \cong (M', x)$, then (M, z) is a stable solution if and only if (M', x) is a stable solution.

This invariant of stability between $[(G, w, c), M]$ and its auxiliary is the key feature we rely upon within this work.

Chapter 4

Computing an M-stabilizer

Given a NBG (G, w, c) together with a maximum weight c-matching M, we call a set $S \subset V(G)$ an *M-stabilizer* if it is a vertex stabilizer of (G, w, c) with the additional property that it avoids M, i.e. no edge in M is incident with a vertex in S. Our goal in this section is to compute an M-stabilizer when given $[(G, w, c), M]$, that is, a NBG (G, w, c) together with a c-matching M. We will assume M is of maximum weight. We shall perform this process by mapping to the auxiliary $[(G', w', \mathbb{1}), M']$, stabilizing it, and determining a meaningful way of mapping the stabilizer back to the original NBG to determine an M-stabilizer of $[(G, w, c), M]$, thus solving our problem.

Definition 13. A *network assignment* is a NBG (G, w, c) together with a maximum weight c-matching. A *solution* to the network assignment $[(G, w, c), M]$ is a vector $z \in \mathbb{R}^{2E(G)}$ such that (M, z) is a solution to (G, w, c).

Given that a matching which is not maximum weight in a unit-capacity graph cannot admit a stable solution, we first ask whether or not a maximum weight c-matching in a NBG (G, w, c) will correspond to a maximum weight matching in the auxiliary. However, this is seen to be false. Consider the network assignment $[(G, w, c), M]$, where G is given by $V(G) = \{u, v, x, y\}, E(G) = \{uv, vx, vy, xy\}$, $w_{uv} = w_{vx} = w_{vy} = w_{xy} = 1$, $c_u = c_v = 2$, $c_x = c_y = 1$, together with the c-matching $M = \{vx, vy\}$. See Figure 4.1 for illustration.

We can see that G' admits a proper M'-augmenting path $P' = (u_1; u_1v_1, v_1x_1, x_1y_1, y_1v_2, v_2u_2; u_2)$, thus demonstrating that M' is not a maximum weight matching. On the other hand, it is easy to verify that M is nonetheless a maximum weight c-matching on G. This justifies the following fact.

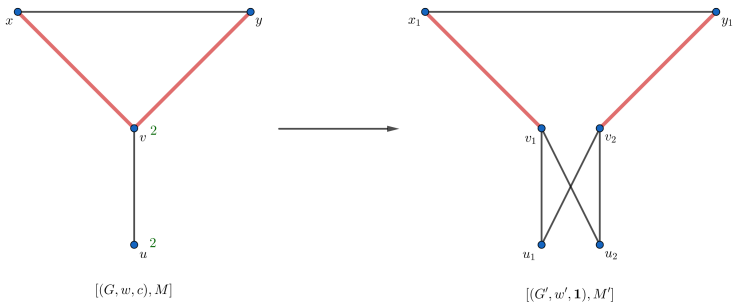

Figure 4.1: A network assignment together with its auxiliary. We can easily see that the c-matching M in G is of maximum weight, yet M' in G' is not.

Observation 14. There exists a network assignment $[(G, w, c), M]$ with auxiliary $[(G', w', \mathbb{1}), M']$ such that M is of maximum weight in G while M' is not of maximum weight in G'.

Proof. The (counter) example in Figure 4.1 satisfies the conditions: M is of maximum weight while M' is not. □

Observe that in the given counterexample, the edges u_1v_1 and u_2v_2 in M' are distinct, which is in contrast to the single edge uv in M. This contrast is effectively the reason that M cannot be augmented in an analogous way as M' to create a c-matching of (strictly) greater weight in G. This cautionary observation motivates the following definition.

Definition 15. Let $[(G, w, c), M]$ be a network assignment with auxiliary $[(G', w', \mathbb{1}), M']$, and suppose P' is an M'-alternating walk. Then a *tie* in P' is a pair of edges $\{ab, cd\} \subset P' \cap (E(G') \setminus M')$ such that for some distinct $u, v \in V(G)$, either $\{a, c\} \subset C_u$ and $\{b, d\} \subset C_v$ or $\{a, d\} \subset C_u$ and $\{b, c\} \subset C_v$. We say P' is *tieless* if it does not contain a tie.

Intuitively, a tie is simply a pair of unmatched edges in an M'-augmenting path that extend between the same two clouds. For instance, the pair of edges $\{u_1v_1, u_2v_2\}$ in the above example form a tie in the augmenting path P'. Note that while P' does not repeat any edge, $\eta(P')$ does, precisely due to the tie in P'. This informs the following general observation.

Observation 16. Suppose $[(G, w, c), M]$ be a network assignment with auxiliary $[(G', w', \mathbb{1}), M']$. If P' is a proper M'-augmenting path or cycle, then $\eta(P')$ need merely be an M-alternating walk.

In other words, the augmenting property of P' need not be preserved in $\eta(P')$. With this correspondence in mind, we establish the following stronger relationships:

Lemma 17. Let $[(G, w, c), M]$ be a network assignment, and let $[(G', w', \mathbb{1}), M']$ be the auxiliary. Suppose P' is a tieless M'-augmenting cycle in G', and let $P = \eta(P')$. Then:

1. $|E(P) \cap M| = |E(P) \setminus M|$
2. $d_t^{M \Delta P} = d_t^M$ for all $t \in V(P)$

Proof. First, note that since P' is an M'-augmenting cycle in G', we have $|E(P') \cap M'| = |E(P') \setminus M'|$. Since P' is tieless, there is a bijection between $E(P')$ and $E(P)$. Further, there is a bijection between M' and M (as always). Therefore, $|E(P) \cap M| = |E(P) \setminus M|$, which proves (1). To see (2), note that P is M-alternating. Therefore, if $t \in V(P)$ has an edge $e \in E(P) \setminus M$ incident to it, the subsequent edge $f \in E(P)$ of the sequence belongs to M. By (1), we can pair such edges e and f together in a well-defined (but arbitrary) way. Upon augmentation, we have that $e \in E(P) \Delta M$ and $f \notin E(P) \Delta M$, meaning the degree of t will not change with respect to e and f. By (1), this generalizes to all such pairs (e, f), and hence the degree of t will remain invariant overall. This proves (2). □

Proposition 18. Let P' be a proper M'-augmenting path or cycle in G', and let $P = \eta(P')$ be the corresponding M-alternating walk. Then P is a proper M-augmenting trail if and only if P' is tieless.

Proof.
$(\implies)$
Suppose P is a proper M-augmenting trail. If P' contained a tie $\{u_1v_1, u_2v_2\}$, where $u_1, u_2 \in C_u$ and $v_1, v_2 \in C_v$, then $\eta(u_1v_1) = \eta(u_2v_2) = uv$, and hence the walk P would traverse the edge uv more than once, contradicting that P is a trail.

$(\impliedby)$
By Lemma 17, if P' is an M'-augmenting cycle, then the degree of each vertex remains constant after augmentation. Since P' is tieless, then in this case P is a proper M-augmenting trail. Thus it suffices to assume P' is a proper M'-augmenting path. If P' does not contain a tie, then if u_iv_j is an edge in P' such that $u_i \in C_u$ and $v_j \in C_v$ for some $u, v \in V(G)$, then there is not a different edge $u_{i'}v_{j'}$ that is also in P'. Therefore P only contains the edge uv once, and so it repeats no edges. Therefore it is a trail. Further, since P' is tieless, then there is a bijection between edges $u_iv_j \in E(P')$ and $uv \in E(P)$. Additionally, $w_{u_iv_j} = w_{uv}$ for each $u_iv_j \in E(P)$. Therefore, $w(P' \setminus M') = w(P \setminus M)$, and

$w(P' \cap M') = w(P \cap M)$. Since P' is M'-augmenting, then $w(P' \setminus M') > w(P' \cap M')$, so therefore $w(P \setminus M) > w(P \cap M)$, and hence P is M-augmenting. Finally, to see that P is proper, we must show that $d_t^{P\Delta M} \leq c_t$ for every $t \in V(G)$. Note that $d_t^{P\Delta M} = d_t^M$ is trivially true for all $t \in V(G) \setminus V(P)$. For all interior vertices of P the degree will remain constant as well, so therefore $d_t^{E(P)\Delta M} = d_t^M$ for all $t \in V(G) \setminus \{u, v\}$, where u and v are the endpoint vertices of P. To see $d_t^{E(P)\Delta M} \leq c_t$ for $t \in \{u, v\}$, we can verify it is true for each of multiple cases, categorized by whether or not $u = v$, and whether or not u_i and v_j are M'-exposed in G', where u_i and v_j are the endpoint vertices of P'. Corresponding to these vertices, let e'_1 and e'_k be the initial and terminal edges of P', respectively. Additionally, let $e_1 = \eta(e'_1)$ and $e_k = \eta(e'_k)$ be the corresponding initial and terminal edges in P.

Case 1; $u = v$
There are four sub-cases to consider, which are categorized by whether or not each of u_i and v_j are M'-exposed. However, the cases when one of u_i and v_j are M'-exposed while the other is not are symmetric, so without loss we will only consider the case when u_i is M'-exposed and v_j is not.

Sub-case 1.1; Neither of u_i or v_j are M'-exposed
In this case, we have $e'_1 \in M'$ and $e'_k \in M'$, so therefore $e_1 \in M$ and $e_k \in M$. This means $e_1 \notin M\Delta E(P)$ and $e_k \notin M\Delta E(P)$. Then since $u = v$, we will have $d_u^{M\Delta E(P)} = d_u^M - 2$. Thus $d_u^{M\Delta E(P)} < d_u^M \leq c_u$.

Sub-case 1.2; u_i is M'-exposed and v_j is not M'-exposed
Here, we have $e'_1 \notin M'$ and $e'_k \in M'$, and so $e_1 \notin M$ and $e_k \in M$. Then $e_1 \in M\Delta E(P)$ and $e_k \notin M\Delta E(P)$. Therefore $d_u^{M\Delta E(P)} = d_u^M - 1 + 1 = d_u^M$, and hence $d_u^{M\Delta E(P)} = d_u^M \leq c_u$.

Sub-case 1.3; Both of u_i and v_j are M'-exposed
We have that $e'_1 \notin M'$ and $e'_k \notin M'$, and so $e_1 \notin M$ and $e_k \notin M$. In this case, since both u_i and v_j are M'-exposed, then by construction of the auxiliary, this means that $d_u^M \leq c_u - 2$. We have that both $e_1 \in M\Delta E(P)$ and $e_k \in M\Delta E(P)$. This gives us that $d_u^{M\Delta E(P)} = d_u^M + 2 \leq c_u - 2 + 2 = c_u$.

Case 2; $u \neq v$
The three sub-cases are categorized as in Case 1.

Sub-case 2.1; Neither of u_i or v_j are M'-exposed
As above, we will have that $e_1 \in M$ and $e_k \in M$, and so $e_1 \notin M\Delta E(P)$ and $e_k \notin M\Delta E(P)$.

Since $u \neq v$, we will have $d_u^{M\Delta E(P)} = d_u^M - 1$, and $d_v^{M\Delta E(P)} = d_v^M - 1$. Therefore $d_u^{M\Delta E(P)} \leq c_u$ and $d_v^{M\Delta E(P)} \leq c_v$.

Sub-case 2.2; u_i is M'-exposed and v_j is not M'-exposed
Note that we will have $d_v^{M\Delta E(P)} \leq c_v$ for the same reason as in sub-case 2.1. Since u_i is M'-exposed, then by construction of the auxiliary, we have $d_u^M \leq c_u - 1$. We again have that $e_1 \in M\Delta E(P)$, and so $d_u^{M\Delta E(P)} = d_u^M + 1 \leq c_u - 1 + 1 = c_u$.

Sub-case 2.3; Both of u_i and v_j are M'-exposed
Since $u \neq v$, the same reasoning that applied to u_i in sub-case 2.2 can again be used here, for both u_i and v_j in this case. Therefore we have $d_u^{M\Delta E(P)} \leq c_u$ and $d_v^{M\Delta E(P)} \leq c_v$. □

Proposition 19. Let $[(G, w, c), M]$ be a network assignment with auxiliary $[(G', w', \mathbb{1}), M']$. If M' is a maximum weight matching in G', then M is a maximum weight c-matching in G. Moreover, if M is of maximum weight and M' is not, then all proper M'-augmenting paths and cycles contain ties.

Proof. Assume that M' is of maximum weight in G', and suppose for a contradiction that M is not a maximum weight c-matching in G. Then by Proposition 9, G must contain a proper M-augmenting trail $T = (u; e_1, e_2, \ldots, e_k; v)$. Let $T' = (u_i; e'_1, e'_2, \ldots, e'_k; v_j)$ be any corresponding proper M'-augmenting walk (there may be more than one, depending on whether e_1 and e_k are in M or not, but at least one must exist). But in fact T' must either be a proper M'-augmenting path or cycle, which contradicts our assumption that M' is a maximum weight matching. Therefore, M must actually be a maximum weight c-matching.

To see the latter statement, first note that since M' is not of maximum weight, then G' must contain a proper M'-augmenting path or cycle P'. If P' were tieless, then by Proposition 18, $\eta(P')$ would be a proper M-augmenting trail in G. However, the existence of such a proper M-augmenting trail would contradict the assumption that M is of maximum weight. Therefore, P' cannot be tieless. □

As demonstrated in Observation 14, given $[(G, w, c), M]$ and $[(G', w', \mathbb{1}), M']$, the fact that M is of maximum weight does not imply that M' is as well. We will describe such matchings as 'insubstantial':

Definition 20. Given a network assignment $[(G, w, c), M]$ with auxiliary $[(G', w', \mathbb{1}), M']$, we say M is *insubstantial* if M is a maximum weight c-matching in G but M' is not a maximum weight matching in G'. We say M is *substantial* if it is not insubstantial.

Insubstantial matchings are 'bad' in the sense that if $[(G, w, c), M]$ is a network assignment such that M is insubstantial, then $[(G, w, c), M]$ does not have a stable solution. From one perspective, if M' is not of maximum weight, then the auxiliary is not stable. Then by Theorem 12, $[(G, w, c), M]$ is not stable either. From another perspective, if we can find a proper M'-augmenting path or cycle P', then the corresponding M-alternating walk $\eta(P')$ will demonstrate the instability of $[(G, w, c), M]$ via Lemma 6. We will make this latter perspective precise by proving a more general result. To do so, we will first need to introduce some relevant concepts.

Definition 21. Let $[(G, w, c), M]$ be a network assignment and let $W = (u; e_1, \ldots, e_k; v)$ be an M-alternating walk. Then for any $\epsilon > 0$, the *ϵ-augmentation* of W is the vector $x^{M/W}(\epsilon) \in \mathbb{R}^{E(G)}$ given by

$$x_e^{M/W}(\epsilon) = \begin{cases} 1 - \kappa(e, W)\epsilon & \text{if } e \in M \\ \kappa(e, W)\epsilon & \text{if } e \notin M \end{cases}$$

where $\kappa(e, X) = |\{i \in [k] \mid e_i = e, e \in E(X)\}|$ for any subsequence X of W. We say that W is *feasible* if the ϵ-augmentation of W is feasible for some $\epsilon > 0$. We may omit the parameter ϵ from the notation when the particular choice of $\epsilon > 0$ is not relevant. We also introduce the notation $K_r(X) = \sum_{e \in \delta(r)} \kappa(e, X)$, for any $r \in V(G)$.

Observation 22. An augmenting path is feasible if and only if it is proper.

Then as a corollary to Lemma 6, we obtain the following.

Corollary 23. If $W = (u; e_1, \ldots, e_k; v)$ is a feasible M-augmenting walk, then
(i) $[(G, w, c), M]$ is not stable.
(ii) If $e_1 (e_k) \notin M$ and $u \neq v$, then $u (v)$ is not M-saturated. Similarly, if $u = v$ and both $e_1 \notin M$ and $e_k \notin M$, then $u = v$ is not M-saturated.

Proof. We first show (i). Let $x^M \in \mathbb{R}^{E(G)}$ be the characteristic vector of M. Since we assume M is of maximum weight, then by Lemma 6, $[(G, w, c), M]$ is stable if and only if x^M is an optimal solution to (P^c_{FM}). However, we can see that the latter condition does not hold: since W is feasible, then $x^{M/W}$ is feasible, and since W is augmenting, then $w(W \setminus M) - w(W \cap M) > 0$. Therefore $w^T x^{M/W} = w^T x^M + \epsilon(w(W \setminus M) - w(W \cap M)) > w^T x^M$, and we see that $x^{M/W}$ is a feasible solution to (P^c_{FM}) whose objective value is strictly greater than that of any integral solution. Thus $[(G, w, c), M]$ is not stable.

To see (ii), first assume $u \neq v$, and $e_1 \notin M$. Then $x^{M/W}(\delta(u)) = x^M(\delta(u)) + \epsilon(K_u(W \setminus M) - K_u(W \cap M)) = x^M(\delta(u)) + \epsilon \leq c_u$. If u was M-saturated, then we would have

$x^M(\delta(u)) = c_u$, but this would contradict $x^M(\delta(u)) + \epsilon \leq c_u$. By symmetry, the analogous result with respect to v holds when $e_k \notin M$. The final statement also holds through a similar calculation. □

We can now make our previous notion precise. In fact, we can derive a stronger result.

Proposition 24. $[(G, w, c), M]$ has a feasible M-augmenting walk if and only if $[(G', w', \mathbb{1}), M']$ has a feasible M'-augmenting walk.

Proof.
($\Longrightarrow$)
Let $P = (u; e_1, \ldots, e_k; v)$ be a feasible M-augmenting walk. Note that P determines a M'-augmenting walk which is unique up to the initial and terminal edges. If $e_1 \in M$, then the associated edge in the walk determined by P is unique, since it must be in M'. Similarly with regards to e_k. If $e_1 \notin M$, then by Corollary 23, u is not M-saturated. This means there exists some vertex $u_i \in C_u$ in the auxiliary which is M'-exposed. Select the associated walk such that u_i is the initial vertex. The analogous choice is made with respect to e_k and v. Let $P' = (u_i; e'_1, \ldots, e'_k; v_j)$ be a walk in the auxiliary determined by the above. This P' is then a feasible M'-augmenting walk.
($\Longleftarrow$)
Let $P' = (u_i; e'_1, \ldots, e'_k; v_j)$ be a feasible M'-augmenting walk, and let $\eta(P') = (u; e_1, \ldots, e_k; v)$. Then $\eta(P')$ is a feasible M-augmenting walk. □

Corollary 25. Let (G, w, c) be a NBG. If there exists a maximum weight matching M such that the associated auxiliary $[(G', w', \mathbb{1}), M']$ contains a feasible M'-augmenting walk, then (G, w, c) does not admit a stable solution. In particular, if M is insubstantial, then (G, w, c) does not admit a stable solution.

Proof. If $[(G', w', \mathbb{1}), M']$ contains a feasible M'-augmenting walk, then $[(G, w, c), M]$ contains a feasible M-augmenting walk by Proposition 24. This implies that (P^c_{FM}) does not have an integral optimal solution, and so by Lemma 6 (G, w, c) does not have a stable solution. □

In view of Corollary 25, we can interpret feasible augmenting walks as 'unstable substructures', which prevent the network assignment from admitting a stable solution. Then it is clear that if we have any hope of stabilizing our given network assignment, we must ensure that neither it, nor its auxiliary, contains any feasible augmenting walks.
We remark now that in the case of unit-capacity graphs, a (more refined) characterization of 'unstable substructures' has already been established. It is given by the following theorem.

Theorem 26. ([KS20]) If a (unit-capacity) graph is stable, then it does not have an M-augmenting flower or bicycle, for every maximum weight matching M. If it is not stable, then it has an M-augmenting flower or bi-cycle for every maximum weight matching M.

Hence if $[(G, w, c), M]$ is any network assignment such that M is substantial, we can also detect stability of $[(G', w', \mathbb{1}), M']$ (and thus $[(G, w, c), M]$) in terms of the existence of an M'-augmenting flower or bi-cycle. On the other hand, if M is insubstantial, then $[(G', w', \mathbb{1}), M']$ cannot admit a stable solution (by Corollary 7). Additionally, M' is not of maximum weight if and only if there exists a proper M'-augmenting path or cycle. Therefore these structures also demonstrate instability (of both the auxiliary and the original network assignment). We summarize this discussion in the following result.

Proposition 27. A network assignment $[(G, w, c), M]$ is not stable if and only if the auxiliary $[(G', w', \mathbb{1}), M']$ contains at least one of the following:
(i) an M'-augmenting flower
(ii) an M'-augmenting bi-cycle
(iii) a proper M'-augmenting path
(iv) an M'-augmenting cycle

Proof. $[(G, w, c), M]$ is not stable if and only if $[(G', w', \mathbb{1}), M']$ is not stable. Consider when the latter condition occurs in two distinct scenarios: a) M' is of maximum weight, and b) M' is not of maximum weight. If M' is of maximum weight, then by Theorem 26, $[(G', w', \mathbb{1}), M']$ contains an M'-augmenting flower or bi-cycle, which covers (i) and (ii). If instead M' is not of maximum weight, then $[(G', w', \mathbb{1}), M']$ must contain a proper M'-augmenting path or cycle. Moreover, by Corollary 7, $[(G', w', \mathbb{1}), M']$ is not stable. Then this covers (iii) and (iv). □

We have now given two distinct combinatorial characterizations the instability of a network assignment $[(G, w, c), M]$ - one in terms of walks, the other in terms of flowers, bi-cycles, paths and cycles. It then seems natural that these structures are related, which indeed is the case.

Proposition 28. ([KS20]) In a unit-capacity graph, a feasible augmenting uv-walk contains at least one of the following: i) an augmenting flower rooted at u or v, ii) an augmenting bi-cycle, iii) a proper augmenting uv-path, iv) an augmenting cycle.

Remark 29. In their proof, the authors prove this algorithmically. They provide an efficient method decomposing the walk, and subsequently finding an augmenting structure of type (i)-(iv).

It follows that an M'-stabilizer of an auxiliary graph $[(G', w', \mathbb{1}), M']$ must include some vertex from every feasible M'-augmenting walk (if an M'-stabilizer exists at all). The only possible vertices of an augmenting walk which might not be M'-covered are the endpoints. Thus an M'-stabilizer must remove at least one endpoint from every feasible M'-augmenting walk. In particular, if there exists a feasible M'-augmenting walk such that both endpoints are M'-covered, then no M'-stabilizer can exist. Consequently, no M-stabilizer can exist either. If a feasible M'-augmenting walk has exactly one M'-exposed endpoint, then this vertex must necessarily be removed. On the other hand, if both endpoints are M'-exposed, then there initially appears to be some ambiguity in terms of which vertex to remove. We shall soon see that this is a red herring, and it will in fact be possible to make an unambiguous choice regarding which vertex to remove.

Since Proposition 28 is able to find augmenting paths, cycles, flowers, and bi-cycles from augmenting walks, this motivates the preliminary problem of finding augmenting walks to begin with. Accordingly, the authors in [KS20] provide an efficient algorithm for finding optimal feasible M-alternating walks of a bounded length in unit-capacity graphs, where a feasible M-alternating walk P is said to be *optimal* if $w(P \setminus M) - w(P \cap M) \geq w(Q \setminus M) - w(Q \cap M)$ for all feasible M-alternating walks Q. Note that if P is an optimal feasible alternating walk such that $w(P \setminus M) - w(P \cap M) \leq 0$, then no feasible M-augmenting walk exists. For referencing purposes, we summarize this in the following result.

Proposition 30. ([KS20]) Given a unit-capacity graph $(G, w, \mathbf{1})$ with matching M, there is an efficient algorithm for finding optimal feasible M-alternating sv-walks of length at most k, for any $k \geq 1$ and source vertex $s \in V(G)$.

Finally, we address the issue of how to make an unambiguous choice of which vertex to remove in the case that we find a feasible M'-augmenting walk in the auxiliary in which both endpoints are M'-exposed. Due to Proposition 28, it suffices to consider the case when the walk is a path. We introduce the following definition:

Definition 31. Let $[(G, w, c), M]$ be a network assignment and let $P = (u; e_1, \ldots, e_k; v)$ be an M-alternating walk which repeats an edge in opposing directions. Let t be the least index such that $e_t = e_s$ for some $s < t$, and e_s and e_t are traversed by P in opposing directions. Then the *u-traceback* of P is the walk $\mathrm{tb}(P, u) = (e_1, \ldots, e_t, e_{s-1}, e_{s-2}, \ldots, e_1)$, and the *v-traceback* of P is similarly defined as $\mathrm{tb}(P, v) = (e_k, e_{k-1}, \ldots, e_s, e_{t+1}, e_{t+2}, \ldots, e_k)$.

The relevancy of this definition is given by the following lemma.

Lemma 32. Given a network assignment $[(G, w, c), M]$ with auxiliary $[(G', w', \mathbb{1}), M']$, let $P' = (u_i; e'_1, \ldots, e'_k; v_j)$ be a feasible M'-augmenting path such that both u_i and v_j are M'-exposed and in distinct clouds. Then $\mathrm{tb}(\eta(P'), \eta(u_i))$ and $\mathrm{tb}(\eta(P'), \eta(v_j))$ are well-defined, and at least one of them is a feasible M-augmenting walk.

Proof. Let $P = \eta(P') = (u; e_1, \ldots, e_k; v)$. To see that $\mathrm{tb}(P, u)$ and $\mathrm{tb}(P, v)$ are well-defined, we must show that P traverses some edge in opposing directions. This is indeed the case: first note that by Proposition 19, P' contains a tie, and hence P must repeat some (unmatched) edge. Then let t be the least index such that $e_t = e_s$ for some $s < t$. Decompose P as $(P_1, e_s, P_2, e_t, P_3)$, where $P_1 = (e_1, \ldots, e_{s-1})$, $P_2 = (e_{s+1}, \ldots, e_{t-1})$, and $P_3 = (e_{t+1}, \ldots, e_k)$. Suppose P traverses e_s and e_t in the same direction.

Claim 33. If P traverses e_s and e_t in the same direction, then (P_1, e_s, P_3) is a shorter feasible M-augmenting walk.

Proof. For notation, define $P_1^+ = (P_1, e_s)$, $P_2^+ = (P_2, e_t)$. Note that P_2^+ is an M-alternating circuit by definition, which is then necessarily proper as well. However, if $w(P_2^+ \setminus M) - w(P_2^+ \cap M) > 0$, then P_2^+ would be a proper M-augmenting circuit, which would contradict the fact that M is of maximum weight in G. Therefore, we must have that $w(P_2^+ \setminus M) - w(P_2^+ \cap M) \leq 0$. However, P is an M-augmenting walk, meaning that $w(P_1^+ \setminus M) + w(P_2^+ \setminus M) + w(P_3 \setminus M) - (w(P_1^+ \cap M) + w(P_2^+ \cap M) + w(P_3 \cap M)) > 0$. Combining this with the previous inequality, we can see that $w(P_1^+ \setminus M) + w(P_3 \setminus M) - (w(P_1^+ \cap M) + w(P_3 \cap M)) > 0$. Since (P_1^+, P_3) is indeed a walk, the conclusion follows. □

By Claim 33, we may assume w.l.o.g. that P traverses e_s and e_t in opposing directions. If they did not, we could find a shorter feasible M-augmenting walk $W = (P_1, e_s, P_3)$ which is a sub-walk of P. This would also necessarily repeat an (unmatched) edge - for if it did not, W would be feasible M-augmenting trail. Since the endpoints of W are distinct and it is a trail, then it would be a proper augmenting trail. But then by Proposition 9, this would contradict M being of maximum weight. Therefore $\mathrm{tb}(P, u)$ and $\mathrm{tb}(P, v)$ are well-defined. Then consider the walks $P_u = \mathrm{tb}(P, u)$ and $P_v = \mathrm{tb}(P, v)$. Note $P_u = (P_1, P_2^{++}, P_1^{-1})$ and $P_v = (P_3^{-1}, (P_2^{++})^{-1}, P_3)$, where $P_2^{++} = (e_s, P_2, e_t)$. We have that either $w(P_1 \setminus M) - w(P_1 \cap M) \geq w(P_3 \setminus M) - w(P_3 \cap M)$ or vice-versa (or both, in the case of equality). Suppose w.l.o.g. the former. We also know that $w(P_1 \setminus M) + w(P_2^{++} \setminus M) + w(P_3 \setminus M) > w(P_1 \cap M) + w(P_2^{++} \cap M) + w(P_3 \cap M)$. Combining these inequalities yields $2w(P_1 \setminus M) + w(P_2^{++} \setminus M) > 2w(P_1 \cap M) + w(P_2^{++} \cap M)$. Note that the LHS and RHS of this inequality are equal to $w(P_u \setminus M)$ and $w(P_u \cap M)$, respectively. Since $e_1 \notin M$ and u is not M-saturated, then this shows that P_u is a feasible M-augmenting uu-walk. If the previous

inequality held tight with equality, i.e. if $w(P_1 \setminus M) - w(P_1 \cap M) = w(P_3 \setminus M) - w(P_3 \cap M)$, then by symmetry, the above analysis shows that both P_u and P_v are feasible M-augmenting uu and vv - walks, respectively. □

Note that this lemma only accounts for the case when u_i and v_j are in distinct clouds. However, this is sufficient, due to the following observation:

Observation 34. If $[(G, w, c), M]$ is a network assignment with auxiliary $[(G', w', \mathbb{1}), M']$, and $X \subset V(G)$ is any set of vertices which avoids M, then $(G \setminus X)' = (G' \setminus X')$, where $X' = \bigsqcup_{u \in X} C_u$.

We abuse notation in the above observation, by letting H' represent the auxiliary of a graph H (with associated weight, capacity, and matching data). We are now able to introduce the algorithm for the M-stabilizer problem.

Algorithm 2: Finding a minimum cardinality M-stabilizer, if one exists

Result: M-stabilizer of minimum cardinality, or result of infeasibility
Input: Network assignment $[(G, w, c), M]$
-Compute the auxiliary $[(G', w', \mathbb{1}), M']$ using Algorithm 1
-Initialize $S \leftarrow \emptyset$ and $S' \leftarrow \emptyset$
foreach *M'-exposed vertex u_i* **do**
  Search for feasible M-augmenting $u_i v_j$-walks of length at most $3n$ using Proposition 30
  if $\exists$ *a feasible M'-augmenting $u_i v_j$-walk such that v_j is M'-covered* **then**
    if *$\eta(u_i)$ is M-covered* **then**
      return INFEASIBLE
    else
      $S \leftarrow S \cup \eta(u_i)$, $S' \leftarrow S' \cup C_{\eta(u_i)}$, $G \leftarrow G \setminus \eta(u_i)$, $G' \leftarrow G' \setminus C_{\eta(u_i)}$
    end
  if $\exists$ *a feasible M'-augmenting $u_i v_j$-walk such that v_j is M'-exposed* **then**
    if *u_i and v_j belong to the same cloud* **then**
      if *$\eta(u_i)$ is M-covered* **then**
        return INFEASIBLE
      else
        $S \leftarrow S \cup \{\eta(u_i)\}$, $S' \leftarrow S' \cup C_{\eta(u_i)}$, $G \leftarrow G \setminus \eta(u_i)$, $G' \leftarrow G' \setminus C_{\eta(u_i}$
      end
    else
      Find a feasible augmenting path, cycle, flower, or bi-cycle P' contained in the walk using Proposition 28
      if *P' is an augmenting cycle or bi-cycle* **then**
        return INFEASIBLE
      else if *P' is a $u_i u_i$-augmenting flower* **then**
        if *$\eta(u_i)$ is M-covered* **then**
          return INFEASIBLE
        else
          $S \leftarrow S \cup \{\eta(u_i)\}$, $S' \leftarrow S' \cup C_{\eta(u_i)}$, $G \leftarrow G \setminus \eta(u_i)$, $G' \leftarrow G' \setminus C_{\eta(u_i)}$
        end
      else if *P' is a $v_j v_j$-augmenting flower* **then**
        if *$\eta(v_j)$ is M-covered* **then**
          return INFEASIBLE
        else
          $S \leftarrow S \cup \{\eta(v_j)\}$, $S' \leftarrow S' \cup C_{\eta(v_j)}$, $G \leftarrow G \setminus \eta(v_j)$, $G' \leftarrow G' \setminus C_{\eta(v_j)}$
        end
      else
        Compute $\mathrm{tb}(\eta(P'), \eta(u_i))$ and $\mathrm{tb}(\eta(P'), \eta(u_i))$ (P' is a path in this branch)
        if *$tb(\eta(P'), \eta(u_i))$ is M-augmenting* **then**
          $S \leftarrow S \cup \{\eta(u_i)\}$, $S' \leftarrow S' \cup C_{\eta(u_i)}$, $G \leftarrow G \setminus \eta(u_i)$, $G' \leftarrow G' \setminus C_{\eta(u_i)}$
        if *$tb(\eta(P'), \eta(v_j))$ is M-augmenting* **then**
          $S \leftarrow S \cup \{\eta(v_j)\}$, $S' \leftarrow S' \cup C_{\eta(v_j)}$, $G \leftarrow G \setminus \eta(v_j)$, $G' \leftarrow G' \setminus C_{\eta(v_j)}$
      end
    end
end
if $w(M) < \nu_f^c(G)$ **then**
  return INFEASIBLE
else
  return S
end

Theorem 35. The M-stabilizer problem is polynomial time solvable. In particular, Algorithm 2 is a polynomial time algorithm to find a minimum cardinality M-stabilizer, if one exists.

Proof. Let $[(G, w, c), M]$ be an input network assignment, with auxiliary $[(G', w', \mathbb{1}), M']$. If $[(G', w', \mathbb{1}), M']$ contains an augmenting structure P' which is either an M'-augmenting flower whose root is M'-covered, an M'-augmenting cycle or bi-cycle, or a feasible M'-augmenting path whose endpoints are both M'-covered, then $[(G', w', \mathbb{1}), M']$ cannot have an M'-stabilizer, because the removal of any M'-exposed vertices will not destroy P', since every vertex in P' is M'-covered. Correspondingly, $\eta(P')$ is a feasible M-augmenting walk, and every vertex in this walk is M-covered. Therefore no M-stabilizer can exist either. Consistent with this observation, the algorithm will return "INFEASIBLE" in any of these cases, as it will either find P' explicitly, or calculate that $w(M) < \nu_f^c(G)$. Suppose instead that $[(G', w', \mathbb{1}), M']$ contains none of the aforementioned structures. Then the only subgraphs which can make $[(G', w', \mathbb{1}), M']$ unstable are M'-augmenting flowers with an M'-exposed root and feasible M'-augmenting paths with at least one M'-exposed endpoint.

Claim 36. Every feasible augmenting walk contains a feasible augmenting walk of length at most $3n$.

Proof. By Proposition 28, a feasible M'-augmenting u_iv_j-walk must contain a feasible augmenting u_iv_j-path with endpoint or augmenting flower with M'-exposed root u_i or v_j. A path is trivially a walk of length at most n, and an augmenting flower $R \cup B$ naturally induces a feasible augmenting walk (R, B, R^{-1}) of length at most $3n$ by definition, where R is the proper alternating path (of length at most n), and B is the alternating blossom (of length at most n). □

By Claim 36, it suffices to search for feasible augmenting walks of length at most $3n$. Let P' be a feasible M'-augmenting walk with source vertex u_i which was found by the algorithm on some iteration. If the other endpoint is M'-covered, then P' can only be destroyed by removing u_i. Correspondingly, $\eta(P')$ is a feasible augmenting walk by Proposition 24, and it may only be destroyed by removing $\eta(u_i)$. If $\eta(u_i)$ is M-covered, then this is not possible, and hence $[(G, w, c), M]$ does not have an M-stabilizer. In this case the algorithm returns "INFEASIBLE". If instead $\eta(u_i)$ is not M-covered, then it is added to S and removed from G. Due to Observation 34, we remove all vertices in $C_{\eta(u_i)}$ from G'. On the other hand, if the other endpoint, v_j, was M'-exposed, then the algorithm will first check if the endpoints belong to the same cloud. If so, then $\eta(P')$ is a feasible M-augmenting walk in which the only vertex which is potentially not M-covered is $\eta(u_i)$.

If $\eta(u_i)$ is M-covered, then we again conclude no M stabilizer exists, and the algorithm returns "INFEASIBLE". If it is not M-covered, then it is removed from G and added to S. Correspondingly, $C_{\eta(u_i)}$ is removed from G' and added to S'. Now suppose the endpoints do not belong to the same cloud. The algorithm finds a feasible M'-augmenting u_iv_j-path or augmenting flower rooted at u_i or v_j via Proposition 28. If it finds a flower, suppose w.lo.g. it is rooted at u_i. Then just as above, the $\eta(u_i)$ will be the only vertex which is potentially not M-covered in some feasible M-augmenting walk (namely the η-image of the walk induced by the flower). If it is M-covered, no M-stabilizer exists and the algorithm returns "INFEASIBLE", and if not, then $\eta(u_i)$ is removed from G and added to S, and $C_{\eta(u_i)}$ is removed from G' and added to S'. If the algorithm had instead found a feasible M'-augmenting u_iv_j-path Q', then the algorithm computes $\text{tb}(\eta(Q'), \eta(u_i))$ and $\text{tb}(\eta(Q'), \eta(v_j))$. By Lemma 32, at least one of these will be a feasible M-augmenting walk. Suppose w.l.o.g. that $\text{tb}(\eta(Q'), \eta(u_i))$ is a feasible M-augmenting walk. Similar to the previous cases, $\eta(u_i)$ is the only vertex in the walk which may not be M-covered. If it is M-covered, there cannot be an M-stabilizer, and the algorithm returns "INFEASIBLE". Otherwise, $\eta(u_i)$ is removed from G and added to S, and $C_{\eta(u_i)}$ is removed from G' and added to S'.

Claim 37. If Algorithm 2 successfully returns a set S of vertices, then S is a minimum-cardinality M-stabilizer of the (initial) network assignment $[(G, w, c), M]$. If it does not, then no M-stabilizer exists.

Proof. If the algorithm fails to return a set of vertices, it returns a result of "INFEASIBLE", which occurs either when it explicitly finds a feasible M or M'-augmenting walk in which every vertex is M or M'-covered (respectively), or if it calculates (P^c_{FM}) does not have an integral optimal solution. Both cases prove that no M-stabilizer can exist. Suppose the algorithm returns a set S of vertices. By the above analysis, every vertex in S was the unique M-exposed vertex of some feasible M-augmenting walk in G. The existence of a feasible M-augmenting walk certifies the instability of the network assignment, and hence S must be contained in any M-stabilizer. Then suppose S were not an M-stabilizer. This would mean the (induced) network assignment on $(G \setminus S)$ does not have a stable solution. By Proposition 27, its auxiliary must contain a feasible M'-augmenting path, flower, cycle or bi-cycle. However, any of these structures would have been detected when searching for feasible M'-augmenting walks with an exposed source vertex, or when calculating whether $w(M) < \nu^c_f(G \setminus S)$ was true. But if such a walk was found, or the inequality held, the algorithm either would have continued iterating, or would have returned "INFEASIBLE", respectively. Yet neither of these events occurred, since the algorithm returned S. Thus we have a contradiction, and so S must in fact be an M-stabilizer. The fact that S is of

minimum cardinality follows from the fact that S is contained in any M-stabilizer. □

Every computation in Algorithm 2 can be done in polynomial time. The result follows. □

Chapter 5

Computing a vertex-stabilizer

Previously, we were given a graph (G, w, c) together with a c-matching M, and wanted to find an M-stabilizer. We now turn to the question of finding a minimum stabilizer when no c-matching M is specified. When our stabilizer is not forced to avoid certain vertices (as is the case with an M-stabilizer), we will show that the problem becomes hard. In this section, we will prove the following result:

Theorem 38. Given an instance (G, w, c) of a NBG, the problem of finding a minimum cardinality vertex stabilizer is NP-complete, even if all edges have unit weight. Furthermore, no efficient $(2 - \epsilon)$-approximation of this problem exists for any $\epsilon > 0$, assuming UGC.

We first introduce the well-known vertex cover problem, and will subsequently demonstrate a reduction to it from the vertex stabilizer problem.

Definition 39. Given a graph $G = (V, E)$, a *vertex cover* of G is a set $C \subseteq V$ such that every edge in E is incident to at least one vertex in C.

Minimum Vertex Cover
Given a graph $G = (V, E)$, find a vertex cover of minimum cardinality.

It is well-known that Minimum Vertex Cover is NP-complete (see e.g. [Kar72]). Due to the reduction we will establish, this will show that the problem of finding a minimum cardinality vertex stabilizer is also NP-complete. Further, [DS05] shows that it is NP-hard to approximate Minimum Vertex Cover within a factor of $10\sqrt{5} - 21 \approx 1.3606$. Additionally, assuming the Unique Games Conjecture (UGC), [KR08] shows that it is

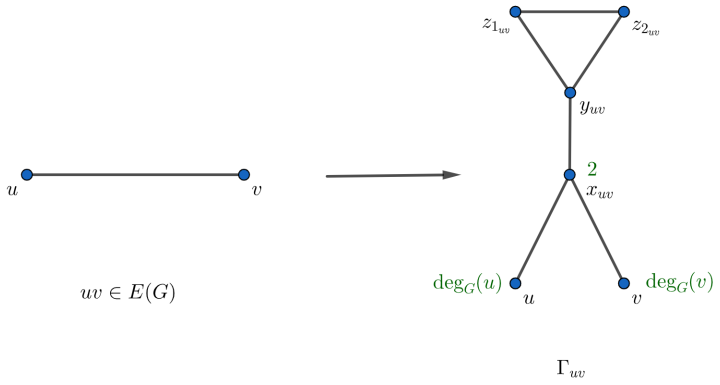

Figure 5.1: The construction of the gadget Γ_{uv} from a pair of adjacent vertices u and v in G. The capacity of u (v) in the derived graph is the degree of u (v) $\in V(G)$. The overall graph $(G_\Gamma, \mathbf{1}, c)$ is the union of all gadgets Γ_{uv} over all adjacent pairs $uv \in E(G)$.

NP-hard to approximate Minimum Vertex Cover within a factor of $2 - \epsilon$, for any $\epsilon > 0$. Since the reduction is approximation-preserving, these inapproximability results extend to the problem of finding a minimum vertex stabilizer.

The idea of the reduction is as follows: given a graph G, we will construct a derived graph G_Γ with specified vertex capacities c via a gadget Γ. Vertex covers in G will correspond to vertex stabilizers in G_Γ with no larger cardinality, and vice-versa, hence demonstrating the hardness and inapproximability results of finding a minimum cardinality vertex stabilizer in $(G_\Gamma, \mathbf{1}, c)$.

Proof. (Theorem 38) Given a graph G, we first describe how to construct the derived NBG $(G_\Gamma, \mathbf{1}, c)$:
Given a pair of adjacent vertices u and v in G, construct $(G_\Gamma, \mathbf{1}, c)$ by replacing the edge uv with a gadget Γ_{uv} consisting of vertices $V(\Gamma_{uv})$ and edges $E(\Gamma_{uv})$ given by

$$V(\Gamma_{uv}) = \{u, v, x_{uv}, y_{uv}, z_{1_{uv}}, z_{2_{uv}}\}$$
$$E(\Gamma_{uv}) = \{u_{uv}x_{uv}, v_{uv}x_{uv}, x_{uv}y_{uv}, y_{uv}z_{1_{uv}}, y_{uv}z_{2_{uv}}, z_{1_{uv}}z_{2_{uv}}\}$$

See Figure 5.1 for reference. We drop the subscripts in the notation when discussing a given gadget, as the vertices u and v will be understood. For each gadget Γ_{uv}, the capacities of u and v are their respective degrees in G, the capacity of x is 2, and the capacity of y, z_1, and z_2 are each 1. The edges are all unit weight. We now demonstrate the following correspondence:

Lemma 40. G has a vertex cover of size at most k if and only if $(G_\Gamma, \mathbf{1}, c)$ has a vertex stabilizer of size at most k.

Proof.
$(\implies)$
Let C be a vertex cover of G such that $|C| = k$. C also denotes the corresponding vertices in G_Γ. We claim that C is a vertex stabilizer of G_Γ: if we remove C from G_Γ, note that since C is a vertex cover of G, then no gadget Γ_{uv} in $G_\Gamma \setminus C$ will retain both u and v. Therefore, on each gadget Γ_{uv} we can create the $c_{\Gamma_{uv}}$-matching $M_{\Gamma_{uv}} = \bigcup_{t \in \{u,v,y\} \setminus C} \{xt\} \cup \{z_1 z_2\}$, where $c_{\Gamma_{uv}}$ is the sub-vector of c indexed by $V(\Gamma_{uv})$. Note that $M_{\Gamma_{uv}}$ is indeed a $c_{\Gamma_{uv}}$-matching, since the capacity of x is 2, and $|\{u, v, y\} \setminus C| \leq 2$. Then $M := \bigcup_{uv \in E(G)} M_{\Gamma_{uv}}$ is a c-matching on $G_\Gamma \setminus C$, by the choice of capacity of each vertex from $V(G)$. Moreover, it is a maximum cardinality c-matching, since every vertex is M-saturated, or is matched with all of its neighbors. Construct a solution (M, z), where $z_{st} = z_{ts} = \frac{1}{2}$ if $st \in M$, and 0 otherwise. Further, note that for each gadget Γ_{uv}, we have $\alpha_t = 0$ for every $t \in \{u, v, x\} \setminus C$, and $\alpha_t = \frac{1}{2}$ for every $t \in \{y, z_1, z_2\}$. Then $\alpha_s \leq z_{st} = \frac{1}{2}$ (and $\alpha_t \leq z_{ts} = \frac{1}{2}$) for every edge $st \in M$. Additionally, the outside option of every unsaturated vertex is 0. Thus (M, z) is a stable solution of $(G_\Gamma \setminus C, \mathbf{1}, c)$ by definition, and so C is a vertex stabilizer.

$(\impliedby)$
Let S' be a vertex stabilizer of $(G_\Gamma, \mathbf{1}, c)$ with cardinality $|S'| = k$. We will find a vertex cover of G_Γ with size at most k using the following claim:

Claim 41. S' contains at least one vertex from every gadget Γ_{uv} of G_Γ.

Proof. Suppose for a contradiction there was a gadget Γ_{uv} of G_Γ which did not contain any vertices of S'. Since $(G_\Gamma \setminus S', \mathbf{1}, c)$ is stable, then by Lemma 6, the associated c-matching LP has an integral optimal solution x^*. Consider a related feasible solution $\bar{x}$, which is equal to x^* for each component corresponding to an edge not contained in $E(\Gamma_{uv})$, and for each edge contained in Γ_{uv}, take $\bar{x}_{ux} = \bar{x}_{vx} = 1$, $\bar{x}_{xy} = 0$, and $\bar{x}_{yz_1} = \bar{x}_{yz_2} = \bar{x}_{z_1 z_2} = \frac{1}{2}$. Since x^* is feasible, then clearly $\bar{x}$ is as well. But observe that $\sum_{e \in E(\Gamma_{uv})} \bar{x}_e = 3.5 > \sum_{e \in E(\Gamma_{uv})} x^*_e$, and hence $\sum_{e \in E(G_\Gamma)} \bar{x}_e > \sum_{e \in E(G_\Gamma)} x^*_e$, which contradicts the optimality of x^*. □

Let $S'_{uv} := S' \cap V(\Gamma_{uv})$ for each gadget Γ_{uv}. Create a new set T'_{uv} by

$$T'_{uv} = \begin{cases} S'_{uv} \cap \{u, v\} & \text{if } S'_{uv} \cap \{u, v\} \neq \emptyset \\ \{u\} \text{ xor } \{v\} \text{ (chosen arbitrarily)} & \text{otherwise} \end{cases}$$

Note that $\emptyset \neq T'_{uv} \subseteq \{u, v\}$ by construction of T'_{uv}, and by Claim 41. Let $T' := \bigcup_{uv \in E(G)} T'_{uv}$. Since every T' contains at least one of u and v for every gadget Γ_{uv}, and does not contain any vertices in $V(\Gamma_{uv}) \setminus \{u, v\}$, then T' is clearly a vertex cover in G. By definition, we have $|T'| \leq |S'|$, and so we have found a vertex cover whose size is at most k. □

By Lemma 40, any minimum vertex stabilizer of $(G_\Gamma, \mathbf{1}, c)$ is of the same size as any minimum vertex cover of G. If we had an efficient algorithm for finding a minimum vertex stabilizer S', then we could efficiently construct a minimum vertex cover T' of G using the described method. Further, any efficient algorithm which gave an efficient α-approximation of a minimum vertex stabilizer (where $\alpha \geq 1$) would also yield an efficient α-approximation of a minimum vertex cover. Since Minimum Vertex Cover does not admit any efficient α-approximation for $1 \leq \alpha < 2$ assuming UGC, the result follows. □

References

[AHS18] Sara Ahmadian, Hamideh Hosseinzadeh, and Laura Sanità. Stabilizing network bargaining games by blocking players. *Mathematical Programming*, 172(1):249–275, 2018.

[BBC+15] Mohsen Bayati, Christian Borgs, Jennifer Chayes, Yash Kanoria, and Andrea Montanari. Bargaining dynamics in exchange networks. *Journal of Economic Theory*, 156:417–454, 2015.

[BBG+14] Péter Biró, Matthijs Bomhoff, Petr A Golovach, Walter Kern, and Daniël Paulusma. Solutions for the stable roommates problem with payments. *Theoretical computer science*, 540:53–61, 2014.

[BCK+15] Adrian Bock, Karthekeyan Chandrasekaran, Jochen Könemann, Britta Peis, and Laura Sanità. Finding small stabilizers for unstable graphs. *Mathematical Programming*, 154(1):173–196, 2015.

[BHIM10] MohammadHossein Bateni, MohammadTaghi Hajiaghayi, Nicole Immorlica, and Hamid Mahini. The cooperative game theory foundations of network bargaining games. In *International Colloquium on Automata, Languages, and Programming*, pages 67–78. Springer, 2010.

[BKP12] Péter Biró, Walter Kern, and Daniël Paulusma. Computing solutions for matching games. *International journal of game theory*, 41(1):75–90, 2012.

[CEW11] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

[CGK+19] Karthekeyan Chandrasekaran, Corinna Gottschalk, Jochen Könemann, Britta Peis, Daniel Schmand, and Andreas Wierz. Additive stabilizers for unstable graphs. *Discrete Optimization*, 31:56–78, 2019.

[DINZ00] Xiaotie Deng, Toshihide Ibaraki, Hiroshi Nagamochi, and Wenan Zang. Totally balanced combinatorial optimization games. *Mathematical Programming*, 87(3):441–452, 2000.

[DS05] Irit Dinur and Samuel Safra. On the hardness of approximating minimum vertex cover. *Annals of mathematics*, pages 439–485, 2005.

[FGK13] Linda Farczadi, Konstantinos Georgiou, and Jochen Könemann. Network bargaining with general capacities. In *European Symposium on Algorithms*, pages 433–444. Springer, 2013.

[IKK+17] Takehiro Ito, Naonori Kakimura, Naoyuki Kamiyama, Yusuke Kobayashi, and Yoshio Okamoto. Efficient stabilization of cooperative matching games. *Theoretical Computer Science*, 677:69–82, 2017.

[Kar72] Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.

[KLS15] Jochen Könemann, Kate Larson, and David Steiner. Network bargaining: Using approximate blocking sets to stabilize unstable instances. *Theory of Computing Systems*, 57(3):655–672, 2015.

[KR08] Subhash Khot and Oded Regev. Vertex cover might be hard to approximate to within 2- ε. *Journal of Computer and System Sciences*, 74(3):335–349, 2008.

[KS20] Zhuan Khye Koh and Laura Sanità. Stabilizing weighted graphs. *Mathematics of Operations Research*, 45(4):1318–1341, 2020.

[KT08] Jon Kleinberg and Éva Tardos. Balanced outcomes in social exchange networks. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 295–304, 2008.

[MRS+11] Sounaka Mishra, Venkatesh Raman, Saket Saurabh, Somnath Sikdar, and CR Subramanian. The complexity of könig subgraph problems and above-guarantee vertex cover. *Algorithmica*, 61(4):857–881, 2011.

[SS71] Lloyd S Shapley and Martin Shubik. The assignment game i: The core. *International Journal of game theory*, 1(1):111–130, 1971.

APPENDICES

Appendix A

Unit-Capacity Structures

A.1 Algorithmic Results

We provide here the algorithms treated as black boxes in the main body. Namely, we provide the algorithms for finding optimal feasible alternating walks of bounded length, decomposing alternating walks, and finding augmenting subgraphs contained in a feasible augmenting walk. Here, all graphs are unit-capacity, and walks will typically be described by a sequence of vertices. All of these algorithms are expressed in [KS20] - implicitly in the cases of Algorithms 4, 5, and 6, and explicitly in the case of Algorithm 3.

We will refer to the result of Algorithm 4 for an input P as the *decomposition* of P, or Decompose(P).

Algorithm 3: Finding optimal feasible M-alternating walks of length at most k

Result: Optimal feasible M-alternating walk of length at most k

Input: Graph G with edge weights w, together with a matching M, source vertex s, and parameter k;

-Initialize $y_1, y_2, z_1, z_2 \in \mathbb{R}^n$

if *s is M-covered* **then**

  $y_1(s) \leftarrow 0$

  $y_2(s) \leftarrow -\infty$

else

  $y_1(s) \leftarrow 0$

  $y_2(s) \leftarrow 0$

end

foreach *vertex $v \neq s$* **do**

  $y_1(v) \leftarrow -\infty$

  $y_2(v) \leftarrow -\infty$

end

for $i = 1$ *to* k **do**

  foreach *vertex v* **do**

    $z_1(v) \leftarrow \max_{u:uv \in E \setminus M} \{y_2(u) + w_{uv}\}$

    $z_2(v) \leftarrow \max_{u:uv \in M} \{y_1(u) - w_{uv}\}$

  end

  foreach *vertex v* **do**

    $y_1(v) \leftarrow \max(y_1(v), z_1(v))$

    $y_2(v) \leftarrow \max(y_2(v), z_2(v))$

  end

end

return y_1, y_2

Algorithm 4: Decomposition of M-alternating walk P

Result: Decomposition $P = (P_1, \ldots, P_l)$ such that every P_i is either an alternating path, cycle or blossom, and no consecutive (P_i, P_{i+1}) are both alternating paths or blossoms

Input: M-alternating walk $P = (v_1, \ldots, v_k)$;

if P *simple* **then**
  return P
else
  -Let j be the least index such that $v_j = v_i$ for some $i < j$
  -Set $P_1 = (v_1, \ldots v_i)$ and $P_2 = (v_i, \ldots, v_j)$
  -Decompose $(v_j, \ldots, v_k)$
  return $(P_1, P_2, \text{Decompose}(v_j, \ldots, v_k)$
end

Algorithm 5: Removal of M-alternating cycles, or detection of M-augmenting cycle

Result: M-augmenting cycle or decomposition $P = (P'_1, \ldots, P'_r)$ such that no P'_i is an M-alternating cycle and P is M-augmenting

Input: (Decomposed) M-augmenting walk $P = (P_1, \ldots, P_l)$;

if *some* P_i *is an augmenting cycle* **then**
  return P_i for smallest such i
else if *no* P_i *is an* M*-alternating cycle* **then**
  return P
else
  while *Decomposition of* P *contains an* M*-alternating cycle* **do**
    Let i be least index such that P_i is an M-alternating cycle;
    Set $P = (P_1, \ldots, P_{i-1}, P_{i+1}, P_l)$
  end
end
;

Algorithm 6: Finding an augmenting path, flower, or bi-cycle

Result: M-augmenting path, flower, or bi-cycle
Input: Decomposed $P = (P_1, \ldots, P_l)$ such that no P_i is an M-alternating cycle;
if l **then**
  return P_1
else if (P_1, P_2) *is an augmenting flower* **then**
  return (P_1, P_2)
end
else if (P_{l-1}, P_l) *is an augmenting flower* **then**
  return (P_{l-1}, P_l)
end
else
  -Set $i = 1$
  while $(P_{2i}, P_{2i+1}, P_{2i+2})$ *not an augmenting bi-cycle* **do**
    $i = i + 1$
  end
  return $(P_{2i}, P_{2i+1}, P_{2i+2})$
end