Computing the Nucleolus of Matching and b-Matching Games

by

W. Justin Toth

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

Waterloo, Ontario, Canada, 2021

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Péter Biró, Senior Research Fellow Center for Economic and Regional Studies, Institute of Economics Hungarian Academy of Sciences
Supervisor:	Jochen Koenemann, Professor Department of Combinatorics and Optimization University of Waterloo
Internal Members:	Chaitanya Swamy, Professor Department of Combinatorics and Optmimization University of Waterloo
	Laura Sanità, Associate Professor Department of Combinatorics and Optimization University of Waterloo
Internal-External Member:	Kate Larson, Professor School of Computer Science University of Waterloo

Author's Declaration

Abstract

In the classical weighted matching problem the optimizer is given a graph with edge weights and their goal is to find a matching which maximizes the sum of the weights of edges in the matching. It is typically assumed in this process that the optimizer has unilateral control over the decision to take each edge. Where cooperative game theory intersects combinatorial optimization this assumption is subverted. In a cooperative matching game each vertex of the graph is controlled by a distinct player, and an edge can only be taken into a matching with the cooperation of the players at each of its vertices. One can think of the weight of an edge as representing the value the players of that edge generate by collaborating in partnership. In this setting the question is more than simply can we find an optimal matching, as in the classic matching problem, but also how should the players share the total value of the matching amongst themselves.

The players should share the value they generate in a way that fairly respects the contributions of each player, and which encourages as well as possible the stable participation of every player in the network. Cooperative game theory formulates such fair distributions of wealth as solution concepts. One classical and beautiful solution concept is the nucleolus. Intuitively the nucleolus distributes value so that the worst off groups of players are as satisfied as possible, and subject to that the second worst off groups, and so on. Here we think of satisfaction as the difference between how much value the players were distributed versus how much they could have generated on their own had they seceded from the grand coalition.

This thesis studies the nucleolus of matching games, and their generalization to b-matching games where each player can take on multiple partnerships simultaneously, from a computational perspective. We study when the nucleolus of a b-matching game can be computed efficiently and when it is intractable to do so.

Chapter 2 describes an algorithm for computing the nucleolus of any weighted cooperative matching game in polynomial time. Chapter 3 studies the computational complexity of b-matching games. We show that computing the nucleolus of such games is NP-hard even when every vertex has b-value 3, the graph is unweighted, bipartite, and of maximum degree 7. Finally, in Chapter 4 we show that when the problem of determining the worst off coalition under a given allocation in a cooperative game can be formulated as a dynamic program then the nucleolus of the game can be computed in time which is only a polynomial factor larger than the time it takes to solve said dynamic program. We apply this result to show that nucleolus of b-matching games can be computed in polynomial time on graphs of bounded treewidth.

Acknowledgements

First and foremost I would like to thank my supervisor Jochen Koenemann for his excellent mentorship. His optimism, honest advice, and enduring support led to an incredibly fulfilling graduate career for me. During the early research for this thesis I had the pleasure of working closely with Kanstantsin Pashovich, whose unique mix of talent, hard work, and humility made him an excellent role model.

I greatly appreciate the time commitment and valuable feedback of my thesis committee: Chaitanya Swamy, Laura Sanità, Kate Larson, and Peter Biró. I also appreciate the hard work of the staff in the C&O department for all they do to make our lives easier.

Thank you to my peers: Sifat Rahman, Edward Lee, Logan Grout, Harry Sivasubramaniam, William Dugan, Madison Van Dyk, Lily Wang, and Akshay Ramachandran for the unique mix of fun conversation and insightful discussion that came to characterize our office over the years. Thank you especially to Cedric Koh and Sharat Ibrahimpur for closely sharing with me the highs and lows of our collective journey through combinatorial optimization.

To my parents Bill and Cheryl, and to my brother Michael, thank you for always believing in me and supporting my decisions. To Maggie Justy, thank you for your incredible acts of thoughtfulness and endless encouragement. I could not ask for better support from my loved ones.

I acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

Cette recherche a été financée par le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG).

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine a network of players that form partnerships to generate value. For example, a tennis league pairing players to play exhibition matches [7], or people making trades in an exchange network [76]. These situations can be modelled by cooperative matching games. Cooperative game theory studies how value generated by players working in cooperation will be shared amongst the players. This thesis looks at matching games and one classical solution concept for describing how players in matching games will share value, the nucleolus, from the perspective of combinatorial optimization. That is, we seek to design algorithms for computing the nucleolus of matching games using tools from graph theory and linear programming. We also seek to prove certain difficult cases are computationally hard.

Section 1.1 will discuss the preliminaries needed in combinatorial optimization for this thesis, including graph theory, algorithms and complexity, and linear programming. This section can be safely skipped for someone familiar with the foundations of this field. Section 1.2 will then introduce concepts from cooperative game theory relevant to this thesis, including explaining what matching games are and what the nucleolus of a game is. In particular Section 1.2.2 will discuss various types of cooperative games which arise from combinatorial optimization problems and survey the landscape of computational complexity results for computing solution concepts for such games. As is the focus of this thesis, we will pay special attention to complexity results around computing the nucleolus of matching games. Section 1.3 will briefly discuss connections between matching games and related areas of research. Section 1.4 will discuss the contributions of this thesis and outline the chapters to follow.

1.1 Combinatorial Optimization

We define an *instance of a mathematical optimization problem instance* to be a pair $(f, \mathcal{X})$ where $\mathcal{X} \subseteq \mathbb{R}^n$ is called the *feasible region* and $f : \mathcal{X} \to \mathbb{R}$ is called the *objective function*[1]. We often write such an optimization problem instance as

$$\begin{aligned} \max\ & f(x) \\ \text{s.t. } & x \in \mathcal{X} \end{aligned}$$

or more compactly as $\max\{f(x) : x \in \mathcal{X}\}$.

An element $x \in \mathcal{X}$ is called a *feasible solution*. A condition all feasible solutions must satisfy is called a *constraint*. If a feasible solution $x \in \mathcal{X}$ satisfies $f(x) \geq f(y)$ for all $y \in \mathcal{X}$ then x is called an *optimal solution*. If an optimization problem instance has a feasible solution but no optimal solution then we call the problem instance is *unbounded*. If $\mathcal{X} = \varnothing$ then we say the problem instance is *infeasible*. To *solve* a problem instance means to find an optimal solution if one exists or otherwise report that the problem is infeasible or unbounded, whichever is the case.

If $\mathcal{X} = \mathbb{R}^n$ then we say the problem instance is *unconstrained*, otherwise we say it is *constrained*. If $\mathcal{X} \subseteq \mathbb{Z}^n$ then we say the problem instance is a *discrete optimization* problem instance or *integer optimization* problem instance. If $\mathcal{X} \subseteq \{0, 1\}^n$ we say the problem instance is a *binary optimization* problem instance.

Oftentimes the feasible region will exhibit additional combinatorial structure such as when we have a binary optimization problem instance where $\mathcal{X}$ is encoding the set of spanning trees in a graph or the set of matchings in a graph. In this case we say the problem instance is a *combinatorial optimization* problem instance. To give us a shared language to discuss such problems, which are of central interest to this thesis, the next section describes introductory terminology in graph theory.

1.1.1 Graph Theory

In this thesis the sort of games we are interested in can often be represented by graphs. The language of graph theory is ideal for describing relationships between players in a network. This section will establish common definitions and notation from graph theory. A reader familiar with the subject can safely skip what follows. For a reader interested in going deeper we recomment Diestel [21].

[1]We chose feasible regions in $\mathbb{R}^n$ as we do not need anything more general for this thesis. As a disclaimer, more general spaces are studied in the broad optimization literature.

An undirected[2], simple *graph* is an ordered pair $G = (V, E)$ where V is a nonempty finite set called the *vertices* or *nodes* of G and $E \subseteq \{\{x, y\} : x, y \in V \text{ and } x \neq y\}$ is a finite set called the *edges* or *arcs* of G. We often use $xy \in E$ as a shorthand for $\{x, y\} \in E$. When V, E are not explicitly labelled it can be handy to use $V(G)$ to denote the vertices of G and $E(G)$ to denote the edges of G.

If $e = xy$ is an edge then we say x is *adjacent* to y. We also say y is a *neighbour* of x, and x and y are *incident* with e, x and y are *endpoints* of e, and e is *incident* with x and y or e is an edge *between* x and y. If $e, f \in E$ are two distinct edges which share a common node then we say e and f are *incident*.

The graph $G' = (V', E')$ is called a *subgraph* of G if $V' \subseteq V(G)$ and $E' \subseteq E(G)$. If G' is a subgraph of G then we write $G' \subseteq G$ and can say G *contains* G'. For any $S \subseteq V(G)$ we use $E(S)$ to denote the set of edges between vertices of S. Formally $E(S) = \{uv \in E : u, v \in S\}$. For $S \subseteq V(G)$, the subgraph of G *induced* by S is denoted $G[S]$ and is defined to be the subgraph satisfying $V(G[S]) = S$ and $E(G[S]) = E(S)$. That is, the vertices of $G[S]$ are S and the subgraph contains every edge of G between vertices of S. The *union* of two graphs G_1 and G_2 is denoted $G_1 \cup G_2$ and is defined to be the graph with vertex set $V(G_1) \cup V(G_2)$ and edge set $E(G_1) \cup E(G_2)$.

Some important operations to obtain subgraphs from a graph $G = (V, E)$ are contraction and deletion. For $S \subseteq V$, the graph obtained from *deleting* S is denoted $G \backslash S$ and is defined to be the graph $(V \backslash S, E \backslash E(S))$. Similarly for $F \subseteq E$, the graph obtained from *deleting* F is denoted $G \backslash F$ and is defined to be the graph $(V, E \backslash F)$. When deleting a singleton $\{a\}$, $G - a$ is used as a shorthand for $G \backslash \{a\}$. For an edge $e = uv \in E$, the graph obtained from G by *contracting* e is denoted G/e and is defined to be the graph obtained from G by merging u and v into a single vertex w. The neighbours of w in G/e are $N(u) \cup N(v) \backslash \{u, v\}$.

For a node $v \in V$ we use $N(v) := \{u \in V : uv \in E\}$ to denote the set of neighbours of v. We can use $N(U) := \bigcup_{v \in U} N(v)$ to denote the neighbours of a set of vertices U. Similarly we use $\delta(v) := \{e \in E : v \in E\}$ to denote the set of edges incident with v. We can use $\delta(U) := \bigcup_{v \in U} \delta(v)$ to denote the edges incident with a set of vertices U. The *degree* of v is equivalently the number of neighbours of v and the number of edges incident with v. We use $d(v)$ to denote the degree of v, i.e. $d(v) = |N(v)| = |\delta(v)|$. If $d(v) = 0$ then v is called an *isolated* node.

We will now discuss some important special cases of graphs. A *complete* graph satisfies that edge possible edge is in E, i.e. for all $u \neq v \in V$, $uv \in E$. The complete graph on n nodes is denoted K_n. A k-partite graph is one where $V = V_1 \dot{\cup} \ldots \dot{\cup} V_k$ can be partitioned in k classes such that for all $i \in [k]$, $E(V_i) = \varnothing$. Here $[k]$ is shorthand for $\{1, \ldots, k\}$. When $k = 2$ we call the graph *bipartite*.

A *walk* is a graph $P = (V, E)$ where the vertices can be ordered $v_0, v_1, \ldots, v_n$ and the edges can be ordered $e_1, \ldots, e_n$ such that for all i in $[n]$, $e_i = v_{i-1}v_i$. In a walk, the sequence $e_1, \ldots, e_n$

[2]As opposed to a *directed* graph, where $E \subseteq V \times V \backslash \{(v, v) : V \in V\}$.

is not permitted to repeat at edge, but may repeat vertices. A *path* is a walk which does not repeat vertices. We say P *starts* at v_0 and *ends* at v_n. Also v_0 and v_n are *endpoints* of P, and P is called a path *between* v_0 and v_n. We refer to P as a v_0v_n-path. The *length* of P is n.

A *circuit* is a graph $C = (V, E)$ where the vertices can be ordered $v_0, v_1, \dots, v_n$ with $v_0 = v_n$ and the edges can be ordered $e_1, \dots, e_n$ such that for all i in $[n]$, $e_i = v_{i-1}v_i$. In a circuit, the sequence $e_1, \dots, e_n$ is not permitted to repeat an edge, but may repeat vertices. A *cycle* is a circuit which does not repeat vertices except for v_0 and v_n. The *length* of C is n. We denote the cycle on n vertices by C_n.

A graph G is called *connected* if it contains a path between every pair of vertices in $V(G)$, otherwise it is *disconnected*. A *connected component*, or sometimes just *component*, of G is a maximal connected subgraph of G. A component is called *even* (*odd* respectively) if it has an even (odd respectively) number of nodes.

We say G is *k-connected* if the smallest size of a set of vertices whose deletion disconnects G is k. Notice that being 1-connected is equivalent to being connected. Similarly we say G is *k-edge-connected* if the smallest size of a set of edges whose deletion disconnects G is k. Being k-connected is a stronger condition than being k-edge-connected.

A *forest* is a graph with no cycles. A graph with no cycles is called *acyclic*. A connected forest is called a *tree*. A tree with at most vertex of degree greater than 1 is called a *star*. The k-star is the star graph with at most one vertex of degree k, and that vertex is called the *center* of the star. The only star with without a uniquely defined center is the 1-star in which either vertex could be thought of as the center. A subgraph G' of G is said to *span* $V(G')$. If T is a subgraph of G which spans $V(G)$ then T is called a *spanning tree*.

We will often want to label the edges or vertices of a graph with some numbers. A *weighted* graph associates with a graph G a weight function $w : E \to \mathbb{R}$ on its edges. In this case we say G is *w-weighted*. Similarly a *degree-bounded* graph associates with G a degree-bound function $b : V \to \mathbb{R}$ on its vertices. In this case we say G is *b-valued*. When we have a function $f : S \to \mathbb{R}$ where S is a finite set, for any $S' \subseteq S$ we use $f(S') := \sum_{s \in S'} f(s)$ to compactly write the sum of f values on elements in S'. This notation is used often with functions of the edges or vertices of a graph.

A *matching* in a graph G is a set of edges $M \subseteq E(G)$ such that every node of G is incident with at most one edge of M, i.e. for all $v \in V(G)$, $|M \cap \delta(v)| \leq 1$. We use $V(M)$ to denote the set of vertices incident with an edge in M. Formally we write $V(M) := \{v \in V(G) : \exists e \in M, v \in e\}$. If a node v is in $V(M)$ we say that v is *M-covered*, otherwise we say v is *M-exposed*. If $V(M) = V(G)$ the we say M is a *perfect* matching.

Example: We are now ready to state our favourite example of a combinatorial optimization problem instance: instances of the *matching problem*. Recall that an optimization problem instance is determined by its feasible region $\mathcal{X}$ and its objective function $f : \mathcal{X} \to \mathbb{R}$.

In a (weighted) matching problem instance we are given a w-weighted graph $G = (V, E)$. The feasible region is the set *characteristic vectors* of matching in G. In general on a ground set T, for $S \subseteq T$ we use $\chi(S)$ to denote the characteristic vector of S. The entries of $\chi(S)$ are indexed by the elements of T and for each $t \in T$, $\chi(S)_t$ is equal to the number of times[3] t appears in S. So then $\mathcal{X} = \{\chi(M) : M \text{ is a matching in } G\}$. From the definition of a matching this can be equivalently written as $\mathcal{X} = \{x \in \{0,1\}^E : x(\delta(v)) \leq 1 \text{ for all } v \in V\}$. The objective function is determined by $f(\chi(M)) = w^T\chi(M) = w(M) = \sum_{e \in E} w_e$.

Oftentimes in combinatorial optimization we will discuss sets and their characteristic vectors interchangeably, with the intent always clear from context. Combinatorial optimization problem instances are often written with only references to the sets in question, as in this equivalent formulation of the matching problem instance on w-weighted graph G:

$$\begin{aligned} \max\ & w(M) & \\ \text{s.t. } & |M \cap \delta(v)| \leq 1 & \forall v \in V \\ & M \subseteq E. & \end{aligned}$$

■

Suppose we wanted to find an optimal solution to a matching problem instance. What strategy would we use to do this? How long would that strategy take find a solution? The next section gives us language for formalizing such questions.

1.1.2 Algorithms and Complexity

We aim to give a high level overview of algorithms and complexity concepts in this section useful for this thesis. For a more formal treatment see Hopcroft and Ulman [43] or Garey and Johnson [35].

A *problem* is a relation $R \subseteq \mathcal{I} \times \mathcal{S}$ where $\mathcal{I}$ is a set of problem instances and $\mathcal{S}$ is a set of problem solutions. That is (I, S) is in R if and only if S is a solution to problem instance I.

Example: Let $\mathcal{I}$ be a set of mathematical optimization problem instances. Let $\mathcal{S}$ be the union of all optimal solutions to instances in $\mathcal{I}$. Then we can form a *mathematical optimization problem* $R \subseteq \mathcal{I} \times \mathcal{S}$ where $((f, \mathcal{X}), x) \in R$ if and only if x is an optimal solution to $(f, \mathcal{X})$.

Let $\mathcal{I}$ be the set of optimization problem instances $(f, \mathcal{X})$ such that $(f, \mathcal{X})$ is a matching problem instance for some w-weighted graph G. Then the associated mathematical optimization problem R is called the *(weighted) matching problem.* ■

[3]We express the definition of $\chi(S)_t$ this way so that we may generalize to multisets.

Example: A *decision problem* is a problem $R \subseteq \mathcal{I} \times \mathcal{S}$ where $\mathcal{S} \subseteq \{0, 1\}$. We can think of decision problem instances as asking a YES-NO question, with answer YES corresponding to 1 and answer NO corresponding to 0.

For every mathematical optimization problem $R \subseteq \mathcal{I} \times \mathcal{S}$ we can associate a decision problem $R' \subseteq (\mathcal{I} \times \mathbb{Q}) \times \{0, 1\}$ where for every $(f, \mathcal{X})$ in $\mathcal{I}$ and for every $k \in \mathbb{Q}$, R' relates $((f, \mathcal{X}), k)$ with 1 if there exists $x \in \mathcal{X}$ such that $f(x) \geq k$, otherwise it relates the instance with 0.

We call this associated decision problem the *decision version* of the mathematical optimization problem. Notice that by using binary search ■

An *algorithm* for a problem $R \subseteq \mathcal{I} \times \mathcal{S}$ is a list of instructions which takes an instance $I \in \mathcal{I}$ as input and outputs a solution[4] $S \in \mathcal{S}$ such that (I, S) is in R. We are interested in measuring the efficiency of an algorithm. To do this we will consider the classic paradigm of worst-case time complexity.

The the *runtime* of an algorithm on a problem instance is the number of elementary operations it takes produce a solution to that instance. The term elementary operations depends on the model of computation in question. Fortunately for most reasonable models of computation, the runtimes only vary by a polynomial factor as the set of elementary operations changes [35, p. 11]. Intuitively one can think of a physical computer and the operations on one which essentially take constant time: assignment, addition, subtraction, multiplication, and comparision for instance.

We are interested in knowing the worst-case runtime as the size of instances grow. This will give us as sense of the asymptotic time complexity. By asymptotic complexity we mean we are only interested in a rough measure that captures the order of growth of the runtime on sufficiently large instances. To formalize this notion we use the *Big-Oh* notation. A function $f : \mathbb{N} \to \mathbb{R}$ is $O(g)$ if there exists non-negative constants c and n_0 such that for all $n \geq n_0$, $f(n) \leq cg(n)$.

Now we need to explain what the size of an instance means. Computers store information in binary. The number of bits needed to encode an instance will define the *size* of the instance. This means encoding the number n will take $O(\log(n))$ bits, and encoding a graph with n vertices will take roughly $O(n^2)$ bits to specify the edges (in an adjacency matrix for instance).

There are multiple possible choices of encoding schemes. We did not need to to choose binary. Since the number of digits needed to represent n in base b is $O(\log_b(n))$, the choice of numerical base only impacts our encoding size by a constant factor. There also multiple ways to encode combinatorial structures like graphs. Graphs can be encoded as adjacency matrices or adjacency lists for example. Each method has its pros and cons, but fortunately they are all polynomially related in the number of vertices. For our purposes it suffices to say that a graph G with n vertices and m edges can be encoded with $O(n + m)$ bits.

[4]We limit ourselves to algorithms which are always correct as opposed to randomized algorithms or approximation algorithms which have different criteria.

We say an algorithm $\mathcal{A}$ which solves $R \subseteq \mathcal{I} \times \mathcal{S}$ has time complexity $r : \mathbb{N} \to \mathbb{R}$ if for every $n \in \mathbb{N}$ the maximum runtime of $\mathcal{A}$ on instances of size n is $r(n)$. If the runtime is bounded by a polynomial, i.e. if there exists a constant d such that $r(n) = O(n^d)$, then we say that $\mathcal{A}$ is a *polynomial time algorithm* and that R can be *solved in polynomial time*.

Sometimes confused for polynomial time algorithms, *pseudopolynomial time* algorithms solve problems in a runtime which is bounded by a polynomial in the size of the instance and the value of the numbers the instance encodes. For example the Knapsack problem: given an instance determined by a vector $a \in \mathbb{R}^m$ and $b \in \mathbb{R}$, solve

$$\max\{w^T x : a^T x \leq b, x \in \{0,1\}^m\}.$$

There is a classic algorithm using the dynamic programming technique which solves this problem in time $O(mb)$. Since the encoding size of b is $\log(b)$ this is not a polynomial time algorithm, but rather a pseudopolynomial time algorithm.

The class P is the set of decision problems which can be solved in polynomial time. Due to Edmonds [23] the decision version of the matching problem is in P. The class NP is the set of decision problems which can be solved by a non-deterministic Turing machine. For our purposes these are decision problems whose Yes instances have a *certificate* which can be checked in polynomial time. Formally a problem R is in NP if for every instance I of R there is an algorithm $\mathcal{A}$ such that if I is a Yes instance then there exists a witness W such that $\mathcal{A}$ run on input (I, W) returns Yes in polynomial time (in terms of the size of I), and if I is a No instance then for every witness W the algorithm $\mathcal{A}$ run on (I, W) returns No. Note that $\mathsf{P} \subseteq \mathsf{NP}$ since there is a polynomial algorithm for every problem in P that does not need the help of a witness certificate. The biggest open question in computer science is whether or not $\mathsf{P} = \mathsf{NP}$.

There are many decision problems which are in NP but for which no polynomial time algorithm is known. For example the Hamiltonian Cycle problem, where each instance is a graph G and the instance is a Yes instance if and only if G contains a cycle of length $|V(G)|$. It is easy to provide witness for Yes instances of Hamiltonian Cycle. If a graph G has a Hamiltonian Cycle, then the witness is simply the vertices of the graph in the order they appear on the cycle. An algorithm can easily check this in polynomial time by verifying their is an edge between each subsequent pair of vertices and that every vertex appears exactly once. Notice that it is much more difficult to conceive of a witness which can be checked in polynomial time for No instances of Hamiltonian Cycle. Problems where that can be done are said to be in the class coNP.

The Hamiltonian Cycle problem is a classic example of an NP-complete problem. A problem is NP-hard if there exists a polynomial time reduction to that problem for every problem in NP. A problem is NP-complete if it is both in NP and NP-hard. A polynomial time reduction from a problem R to a problem R' is an algorithm, $\mathcal{A}$, which given access to an oracle algorithm $\mathcal{A}'$ for R', can solve every instance of R in polynomial time. Notice that giving a polynomial time algorithm for an NP-complete problem would immediately yield a polynomial time algorithm for every problem in NP and thus prove that $\mathsf{P} = \mathsf{NP}$.

Despite an ever growing list of problems which are NP-hard and much attention from the research community no polynomial time algorithm for an NP-hard problem has ever been found. For this reason, amongst others, it is widely believed in computer science that P $\neq$ NP and showing a problem is in NP is commonly taken as evidence that no efficient algorithm can be found. That said, as scientists and mathematicians we must keep a healthy skepticism of this fact until a proof is found or it is determined that no proof can be found.

1.1.3 Linear Programming and Polyhedra

Linear programming is will be a ubiquitous tool in this thesis. This section will cover the basics relevant for this thesis. For anyone intereted in learning more about linear programming, and geometric methods in general, particularly with applications to combinatorial optimization we recommend Grotschel, Lovasz, and Schrijver [40] and Lau, Ravi, and Singh [57].

A *linear optimization problem* or linear program is a mathematical optimization problem $(f, \mathcal{X})$ where $f(x) = c^T x$ for some $c \in \mathbb{R}^n$ and $\mathcal{X}$ is a polyhedron. A *polyhedron* is the solution set of a system of linear inequalities. That is, a polyhedron is a set of form $\{x \in \mathbb{R}^n : Ax \leq b\}$ for some matrix $A \in \mathbb{R}^{m\times n}$ and $b \in \mathbb{R}^m$. If a polyhedron is *bounded*, by which we mean that there exists a positive constant which upper bounds the 2-norm of every point in the polyhedron, then we call the polyhedron a *polytope*.

Let $x_1, \ldots, x_m \in \mathbb{R}^n$ be a set of points in R^n, and consider a vector $\lambda \in \mathbb{R}^m$. We use $\mathbb{1}$ to denote the vector of all ones, with the dimension being clear from context. When $\mathbb{1}^T\lambda = 1$ we say that the linear combination $\lambda^T x$ is an *affine combination*. If, in addition, $\lambda \geq 0$ then we say $\lambda^T x$ is a *convex combination*.

For any $X \subseteq \mathbb{R}^m$, the *dimension* of X is the cardinality of the largest affinely independent subset of X minus 1. We denote by $\text{aff}(X)$ ($\text{conv}(X)$ respectively) the set of affine (convex respectively) combinations of vectors in X. If $X = \text{conv}(X)$ then we say that X is a *convex set* or that X is *convex*. It is not hard to see that polyhedra are convex. If x is a point in a convex set P such that the only way to write x as a convex combination of points in P is the trivial combination $x = 1 \cdot x$ then we call x an *extreme point* of P. When a linear program has an optimal solution and an extreme point then it always has an optimal solution which is an extreme point of its feasible region.

Theorem 1.1.1 *Consider a linear program* $\max\{c^T x : x \in P\}$ *where P is a polyhedron. Suppose this linear program has an optimal solution and that P has an extreme point. Then there exists an optimal solution $x^* \in P$ such that x^* is an extreme point of P.*

For $a \in \mathbb{R}^n$ and $\beta \in \mathbb{R}$, the polyhedron $\{x \in \mathbb{R}^n : a^T x \leq \beta\}$ is called a *halfspace* and the polyhedron $\{x \in \mathbb{R}^n : a^T x = \beta\}$ is called a *hyperplane*. Often we will compactly write a halfspace as $a^T x \leq \beta$ and a hyperplane as $a^T x = \beta$. By definition every polyhedron is the intersection of finitely many halfspaces.

Consider a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. An inequality $c^T x \leq \beta$ is called *valid* for P if $c^T x \leq \beta$ holds for all x in P. Note that if we let a_i denote the i-th row of A then every inequality $a_i^T x \leq b_i$ is valid. A *face* of P is a set $F \subseteq P$ such that there exists a valid inequality $z^T x \leq \beta$ for P such that $F = \{x \in P : z^T x = \beta\}$. In this case the inequality $z^T x \leq \beta$ is said to *induce* or *define* the face F.

Observe that if $F = \{x \in P : c^T x = \beta\}$ is a face of P and $F' = \{x \in F : d^T x = \gamma\}$ is a face of F then F' is a face of P. If a face is a singleton, i.e. $F = \{v\}$ for some $v \in \mathbb{R}^n$, then v is called a *vertex* of P. Polyehdra with vertices are called *pointed*. Whereas vertices are the smallest non-empty faces of a polyehdron, facets are what we call the largest faces of a polyhedron not equal to the polyhedron itself. That is, a *facet* F is an inclusionwise maximal face of P such that $\varnothing \subset F \subset P$. The notion of vertex and extreme point are equivalent

Theorem 1.1.2 *Consider a polyhedron P. Then $x \in P$ is an extreme point of P if and only if x is a vertex of P.*

The following result provides an algebraic characterization of the extreme points of a polyhedron written in *symmetric form*: $P = \{x \in R^n : Ax \leq b, x \geq 0\}$.

Lemma 1.1.3 *[57] Let $P = \{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ be a polyhedron. Let $x \in P$ and let $\bar{A}$ be the matrix consisting of rows of A which x satisfies at equality. That is, if $a^T x \leq b_i$ is in an equality of $Ax \leq b$ and a^T is a row of $\bar{A}$ then $a^T x = b_i$. Then x is an extreme point of P if and only if the columns of $\bar{A}$ which correspond to non-zero entries of x are linearly independent.*

In contrast to extreme points, which lie on the boundary of a polyhedron we will also be interested in points which lie inside the polyhedron. A point x is said to lie in the *relative interior* of P if the containment minimal face of P containing x if P itself.

We are interested in polyhedra which have a finite encoding length for the purposes of discussing algorithms and complexity. We say a polyehdron P is *rational* if there exists $A \in \mathbb{Q}^{m \times n}$ and $b \in Q^m$ such that $P = \{x \in \mathbb{R}^n : Ax \leq b\}$. A rational polyhedron is said to have *facet complexity* ϕ if there exists $A \in Q^{m \times n}$ and $b \in Q^m$ such that each inequality $a_i^T x \leq b_i$ can be encoded using at most ϕ bits, where a_i is the i-th row of A. It turns out the facet complexity bounds the encoding size of a vertex.

Theorem 1.1.4 *[40] Let $P \subseteq \mathbb{R}^n$ be a rational polyhedron. Let $x \in P$ be a vertex of P. Then the encoding size of x is bounded by $O(n^2\phi)$ where ϕ is the facet comlexity of P.*

We say an inequality $c^T x \leq \beta$ is a *separating hyperplane* for polyhedron $P \subseteq \mathbb{R}^n$ and point $x \in \mathbb{R}^n$ if $c^T x \leq \beta$ is valid for P and $c^T x > \beta$. The *separation problem* take a polyhedron P and a point x in P as input and outputs Yes if x is in P and otherwise outputs a separating hyperplane between P and x.

The celebrated Ellipsoid Method of Khachiyan [46] says that solving a linear program reduces to solving its associated separation problem. This provided the first known polynomial time algorithm for linear programming.

Theorem 1.1.5 *[46][40] There exists a polynomial $f(n, \phi)$ in variables n and ϕ such that for any polyhedron $P \subseteq \mathbb{R}^n$ with facet complexity ϕ and for any $c \in \mathbb{R}^n$, the linear optimization problem instance $I = (c^T x, P)$ can be solved in time*

$$O(poly(n, \phi, \sum_{i=1}^{n} \log(c_i), T))$$

where T is the time complexity of solving the separation problem on P with points x such that $\sum_{i=1}^{n} \log(x_i) = O(f(n, \phi))$.

Moreover, if $(c^T x, P)$ has an optimal solution the algorithm finds one contained in a minimal face of P. In particular, if P is pointed then the algorithm returns a vertex optimal solution when one exists.

Many interesting variant problems on polyhedra have algorithms based on the ellispoid which are presented in Grotschel, Lovasz, Schrijver [40]. One such variant of interest to us is the problem of finding a relative interior point in a polyhedron.

Theorem 1.1.6 *[40, Theorem 6.5.5] There exists a polynomial $f(n, \phi)$ in variables n and ϕ such that for any non-empty polyhedron $P \subseteq \mathbb{R}^n$ with facet complexity ϕ a relative interior point of P can be found in time*

$$O(poly(n, \phi, \sum_{i=1}^{n} \log(c_i), T))$$

where T is the time complexity of solving the separation problem on P with points x such that $\sum_{i=1}^{n} \log(x_i) = O(f(n, \phi))$.

Duality and Integrality

Consider the following linear program, which we will call the *primal*

$$\begin{aligned} &\max \sum_{i=1}^{n} c_i x_i \\ &\text{s.t} \sum_{i=1}^{n} a_{i,j} x_i \leq b_j \qquad\qquad \forall j \in [m] \\ &\qquad x \geq 0. \end{aligned}$$

To this primal we will associate a linear program called the *dual*

$$\begin{aligned} &\min \sum_{j=1}^{m} b_j y_j \\ &\text{s.t. } \sum_{j=1}^{m} a_{i,j} y_j \geq c_i && \forall i \in [n] \\ &\qquad\qquad y \geq 0. \end{aligned}$$

Theorem 1.1.7 *[40, Theorem 6.5.14] There exists a polynomial $f(n, \phi)$ in variables n and ϕ such that for any polyhedron $P \subseteq \mathbb{R}^n$ with facet complexity ϕ and for any $c \in \mathbb{R}^n$, the dual of the linear optimization problem instance $I = (c^T x, P)$ can be solved in time*

$$O(poly(n, \phi, \sum_{i=1}^{n} \log(c_i), T))$$

where T is the time complexity of solving the separation problem on P with points x such that $\sum_{i=1}^{n} \log(x_i) = O(f(n, \phi))$.

Notice that any linear program can be rewritten in the form of the primal linear program above. Thus we can take the dual of any linear program. It turns out the dual of the dual is the primal. Moreover it is easy to show the *Weak Duality Theorem* which states that any dual feasible solution provides an upper bound on the optimal value of the primal. In fact we can say something much stronger: the *Strong Duality Theorem* states that when one of the linear programs is feasible, their optimal values are equal.

Theorem 1.1.8 *(Strong Duality) If the primal linear program is feasible then the dual is also feasible and the optimal value of the primal is equal to the optimal value of the dual.*

An interesting corollary of strong duality are the *Complementary Slackness Conditions*. They allow the characterization of an optimal primal solution and optimal dual solution pair.

Corollary 1.1.9 *(Complementary Slackness) Let x be a feasible solution to the primal and let y be a feasible solution to the dual. Then x is optimal for the primal and y is optimal for the dual if and only if the Primal and Dual Complementary Slackness Conditions hold. The Primal Complementary Slackness Conditions state that for all for all $i \in [n]$ either*

$$x_i = 0 \quad or \quad \sum_{j=1}^{m} a_{i,j} y_j = c_i.$$

Similarly the Dual Complementary Slackness Conditions state that for all $j \in [m]$ either

$$y_j = 0 \quad or \quad \sum_{i=1}^{n} a_{i,j} x_i = b_j.$$

Example: We now consider an example related to the matching problem. For any graph $G = (V, E)$ let $P_f(G)$ denote the *fractional matching polytope* of G defined as follows:

$$P_f(G) = \{x \in \mathbb{R}^E : x(\delta(v)) \leq 1 \text{ for all } v \in V, x \geq 0\}.$$

Let $w \in \mathbb{R}^E$ be a vector of edge weights. The following linear program is a *relaxation* of the associated instance of the matching problem called the *fractional matching linear program*:

$$\max\{w^T x : x \in P_f(G)\}.$$

The dual of this linear program is:

$$\min\{\mathbb{1}^T y : y_u + y_v \geq w_{uv} \text{ for all} uv \in E, y \geq 0\}.$$

Solutions to the dual linear program are called *fractional covers*. ■

We say that a polytope is *integral* is every extreme point of that polytope has integer coordinates. For example, on bipartite graphs the fractional matching polytope is integral. That means its extreme points are characteristic vectors of matchings and we can use the ellipsoid method to find max weight matchings in polynomial time on bipartite graphs.

What about finding matchings on non-bipartite graphs? Notice that for C_3 with unit weights, the unique optimal solution is $\frac{1}{2}\chi(E(C_3))$, which is decidedly non-integral. Fortunately we can add constraints to the fractional matching linear program to obtain an integral formulation which is efficiently solvable. By a celebrated result of Edmonds [22] there is a linear programming description for the convex hull of matchings on a general graph.

Example: (Edmonds [22]) Let $G = (V, E)$ be a graph. Let $\mathcal{O}$ be the set of node sets $S \subseteq V$ such that $|S| \geq 3$ and $|S|$ is odd. Let the *matching polytope* of G be

$$P = \text{conv}(\{\chi(M) \in \{0,1\}^E : M \text{ is a matching in } G\}).$$

Then P can be described as

$$\begin{aligned} P = \{x \in \mathbb{R}^E :& \\ & x(\delta(v)) \leq 1 && \forall v \in V \\ & x(E(S)) \leq \frac{|S| - 1}{2} && \forall S \in \mathcal{O} \\ & x \geq 0.\} \end{aligned}$$

Optimizing a linear function over P is called the *matching linear program*. ■

Theorem 1.1.10 *(Edmonds [23]) Given a w-weighted graph $G = (V, E)$ an integral optimal solution to the associated matching linear program, and an optimal solution to its dual, can be found in polynomial time. Hence the weighted matching problem can be solved in polynomial time.*

1.2 Cooperative Game Theory

Game Theory studies the decision-making of rational, self-interested agents in strategic environments. Cooperative Game Theory is the branch of game theory which studies situations where players are able to making binding agreements about the distribution of payoffs outside the rules of the game [65]. For more in depth coverage see the textbook of Chalkiadakis, Georgios and Elkind [13].

Mathematically a cooperative game is defined by a set of players and a function which determines the value (or cost) generated by a coalition of players. Formally a *cooperative game* is an ordered pair (n, ν) where $n \in \mathbb{Z}_+$ is the number of players, and $\nu : 2^{[n]} \to \mathbb{R}$ is the so-called value function of the game. For any $S \subseteq [n]$, $\nu(S)$ indicates the value generated when the players in S collaborate.

Some common classes of cooperative games are: monotone, superadditive, and convex games. A cooperative game (n, ν) is said to be *monotone* if $\nu(S) \leq \nu(T)$ for and $S \subseteq T \subseteq [n]$. The game (n, ν) is called *superadditive* if

$$\nu(S) + \nu(T) \leq \nu(S \cup T)$$

for any $S, T \subseteq [n]$ such that $S \cap T = \varnothing$. The game is said to be *convex* if

$$\nu(S) + \nu(T) \leq \nu(S \cup T) + \nu(S \cap T)$$

for any $S, T \subseteq [n]$. In combinatorial optimization the value function ν of a convex game would be referred to as *supermodular.*

A basic assumption we make in the games we will study is that the value of $\nu(S)$ can be distributed amongst the players in S in any way they see fit. Formally, games with this property are known as *transferable utility games* (TU games).

Example: Let $G = (V, E)$ be a w-weighted graph. Then the matching game on G is the cooperative game (n, ν) where $n = |V|$ and for any $S \subseteq V$, $\nu(S)$ is the maximum w-weight of a matching on $G[S]$. ■

The matching game will serve as the central topic of this thesis. Its generalization to *b-matching games* will also play a large role, so we will define them now.

Example: Let $G = (V, E)$ be a b-valued, w-weighted graph. Then the b-matching games on G is the cooperative game (n, ν) where $n = |V|$ and for any $S \subseteq V$,

$$\begin{aligned} \nu(S) := \max\ & w^T x \\ \text{s.t. } & x(\delta(v)) \le b_v && \forall v \in S \\ & x(\delta(v)) = 0 && \forall v \in V \backslash S \\ & x_e \ge 0 && \forall e \in E \\ & x \in \mathbb{Z}^E. \end{aligned}$$

If we add the constraint that $x_e \le 1$ for all $e \in E$, then we call the b-matching game *simple*.

The underlying combinatorial optimization problem is called a b-matching problem. Notice when $b = \mathbb{1}$ we recover the matching problem. ■

1.2.1 Solution Concepts

Given a cooperative game how should the value of the *grand coalition*, $[n]$, be distributed amongs the players? Cooperative Game Theorists have proposed many different *solution concepts* which provide a mathematical characterization of desireable propreties a outcome should have. A solution concept is an *allocation* $x \in R^n$ (where the i-th coordinates indicates that player i is allocated x_i units of value) with some additional properties. Generally these properties capture one of two ideas: *fairness* and *stability.* Fairness is the idea that the value a player receives should somehow reflect the value of their contribution. Stability is the idea that players should receive value in a way that disincentives defecting from the grand coalition. We will now discuss a some important classical solution concepts. Our discussion is by no means exhaustive. We have concentrated on solution concepts related the nucleolus, which the central object of study and one of the most influential solution concepts in cooperative game theory. We have also chosen to describe the Shapley Value. While this thesis does not present results related to the Shapley Value we would be remiss to omit it from a discussion of solution concepts as it is the arguably the most well-known solution concept in the field.

Shapley Value

The Shapley Value's is driven by the idea of fairness. It was invention by Shapley in 1951 [74]. It is a solution concept which seeks to pin a player's payoff to their marginal contribution. It does this by measuring how much the value of the game increases when player i joins, averaged over the possible orders in which agents can join. We will make this idea formal now.

Let Π_n be the set of permutations of $[n]$. For each $\pi \in \Pi_n$ and each $i \in [n]$, define

$$S_\pi(i) := \{j \in [n] : \pi(j) < \pi(i)\}.$$

Intuitively, if we this of a permutation π as describing the order the players join the game then $S_\pi(i)$ is the set of players who joined the game before i. The *Shapley Value*, $\varphi \in \mathbb{R}^n$ assigns the player their average marginal contribution over all permutations of the player set. For each $i \in [n]$,

$$\varphi_i = \frac{1}{n!} \sum_{\pi \in \Pi} \left(\nu(S_\pi(i) \cup \{i\}) - \nu(S_\pi(i)) \right).$$

Imputations

We now start considering solution concepts built around the notion of stability. The question here is what qualitites should an allocation have to disincentive players from leaving the grand coalition?

Let $x \in \mathbb{R}^n$ be an allocation. The property of individual rationality states that every player is allocation at least the amount of value they can generate on their own. Formally we say x is *individually rational* if $x_i \geq \nu(\{i\})$ for all $i \in [n]$. Another important property is that an allocation only uses as much value as the grand coalition has generated. We say x is *efficient* if $x([n]) = \nu([n])$.

An *imputation* is an allocation which is both individually rational and efficient. For a cooperative game (n, ν) we use $\mathcal{I}(n, \nu)$ to denote the set of imputations of (n, ν). Formally,

$$\mathcal{I}(n, \nu) := \{x \in \mathbb{R}^n : x([n]) = \nu([n]), x_i \geq \nu(\{i\}) \text{ for all } i \in [n]\}.$$

Core and Leastcore

Imputations make the game stable for individuals, but what about coalitions? For an allocation $x \in \mathbb{R}^n$ if it were the case the $x(S) < \nu(S)$ for some $S \subseteq [n]$ then S may have incentive to deviate from the grand coalition and form their own group. The *core* [36] is the set of efficient allocations that forbid this incentive to deviate. We denote the core of (n, ν) by $\mathcal{C}(n, \nu)$, which we formally define as

$$\mathcal{C}(n, \nu) := \{x \in \mathbb{R}^n : x([n]) = \nu([n]), x(S) \geq \nu(S) \text{ for all } S \subseteq [n]\}.$$

Notice that core allocations are imputations.

Observe that the core may sometimes be empty. For example, consider the matching game on the cycle C_3 with unit weight edges. If there existed $x \in R^n$ in the core of this game then we get the clear contradiction

$$2 = \sum_{e \in E(C_3)} x(e) \geq 3.$$

The equality follows since x is efficient, and the inequality follows since $x(e) \geq 1$ for all $e \in E(C_3)$.

When the core is empty we relax the stability constraints. The *ε-core* [75] is the set of efficient allocations such that $x(S) \geq \nu(S) + \varepsilon$ for all $S \subseteq [n]$. Let ε^* be the largest value of ε for which the ε-core is nonempty. Then the ε^*-core has a special name: the *leastcore* [61].

Notice that core emptiness implies that ε^* will be negative, and thus that the ε^*-core need not contain an imputation. In particular allocations in the ε^*-core may not be individually rational. If we want individual rationality we instead consider $P_1(\varepsilon)$:

$$P_1(\varepsilon) := \{x \in \mathcal{I}(n, \nu) : x(S) \geq \nu(S) + \varepsilon \text{ for all } S \subset [n]\}.$$

We let ε_1 be $\max\{\varepsilon : P_1(\varepsilon) \neq \varnothing\}$. The reason for the name for this object will become clear when we discuss the nucleolus. For now, notice that the core, ε-core, and $P_1(\varepsilon)$ are all polyhedra.

We define the *excess* of coaltion $S \subseteq [n]$ with respect to an allocation x to be $x(S) - \nu(S)$ and denote this quantity by $\text{ex}(x, S)$. One can think of the excess as a measure of satisfaction of a coalition. A non-negative excess means S is satisfied with x in the sense that they have as much allocation as they could earn alone. A negative excess means the coalition is unhappy in that same sense. Now we can think of $P_1(\varepsilon_1)$ as the set of imputations which maximize the bottleneck excess, or satisfaction of the worst off coalition.

The following folklore lemma, Lemma 1.2.1, says that for superadditive games, the leastcore and $P_1(\varepsilon_1)$ coincide.

Lemma 1.2.1 *Let (n, ν) be a superadditive cooperative game. If x is an allocation in the ε^*-core (i.e. the leastcore) of (n, ν) then x satisfies individual rationality.*

Proof: If $\varepsilon^* \geq 0$ (i.e. the core is nonempty) then the claim is trivial. So we may assume that $\varepsilon^* < 0$.

Suppose for a contradiction there exists $i \in [n]$ such that $x_i < \nu(\{i\})$. Let $S \subset [n]$ such that $\text{ex}(x, S) = \varepsilon^*$.

If i is not in S then

$$x(S \cup \{i\}) = x(S) + x_i = \nu(S) + \varepsilon^* + x_i < \nu(S) + \nu(\{i\}) + \varepsilon^* \leq \nu(S \cup \{i\}) + \varepsilon^*,$$

contradicting that x is in the ε^*-core. In the chain of inequality above, the last inequality is where superadditivity is needed.

Thus for every $S \subset [n]$ such that $\text{ex}(x, S) = \varepsilon^*$ we have that i is in S. Now we can choose $\delta > 0$, increase x_i by δ and decrease x_j by $\delta/(n-1)$ for each $j \in [n] \backslash \{i\}$ to obtain an allocation x'. The resulting allocation x' is efficient, and yet $\text{ex}(x', S) > \varepsilon^*$ for all $S \subset [n]$. This contradicts the definition of ε^*. ■

Notice that b-matching games, and in particular matching games, are superadditive. So by Lemma 1.2.1, for b-matching games $\varepsilon^* = \varepsilon_1$ and the leastcore is equal to $P_1(\varepsilon_1)$. Since this thesis is primarily concerned with matching and b-matching games, for the remainder of this thesis we will treat the leastcore and $P_1(\varepsilon_1)$ as interchangeable. Alternatively the reader may want to assume we are always working in the setting of superadditive games.

Kernel

The kernel [17] is based around the idea of players being unable to bargain for a fraction of another player's payoff in the allocation. As a measure of power of player i over player j we define the *surplus* of i over j to be

$$S_{i,j}(x) = \max\{\nu(S) - x(S) : S \subset [n], i \in S, j \notin S\}.$$

This quantity is the most i could earn if orchestrated a deviation without j. Player i does so by asking $S \backslash \{i\}$ to join them in deviating, pays each player in $S \backslash \{i\}$ what they received under x, and i keeps the surplus.

If $S_{i,j}(x) = S_{j,i}(x)$ then j can counter i's threat with one of his own. If $S_{i,j}(x) > S_{j,i}(x)$ then the only way j is unharmed is if $x_j = \nu(\{j\})$ as in this case j can earn their payoff on their own. If either of these two situations happened, then we can think of i as not having bargaining power over j.

The *kernel* is the set of imputations for which no player has bargaining power over another and we denote it by $\mathcal{K}(n,\nu)$:

$$\begin{aligned}\mathcal{K}(n,\nu) : \{x \in \mathcal{I}(n,\nu) : \text{for all } i \neq j \in [n], \text{ either:}&\\ S_{i,j}(x) = S_{j,i}(x); \text{ or}&\\ S_{i,j}(x) > S_{j,i}(x), \text{ and } x_j = \nu(\{j\}); \text{ or}&\\ S_{i,j}(x) < S_{j,i}(x), \text{ and } x_i = \nu(\{i\}).\}&\end{aligned}$$

Nucleolus

The nucleolus was first defined by Schmeidler [71]. It is a classical solution concept which builds upon the idea of $P_1(\varepsilon_1)$. Recall that $P_1(\varepsilon_1)$ maximizes the minimum excess over the imputation set. Nucleolus maximizes the second minimum excess subject to being in $P_1(\varepsilon_1)$, and maximizes the third minimum excess subject to that and so on.

Formally consider a cooperative game (n,ν). Let $x \in \mathbb{R}^n$ be an allocation. Enumerate the coalitions as $S_1, S_2, \dots, S_{2^n}$ such that they are sorted in order of non-decreasing excess, i.e. $\text{ex}(x, S_i) \leq \text{ex}(x, S_{i+1})$. Define $\theta(x) \in \mathbb{R}^{2^n}$ to be the vecor where $\theta(x)_i = \text{ex}(x, S_i)$. The *nucleolus* is an allocation which lexicographically maximizes $\theta(x)$ over the imputation set. We denote the nucleolus of (n,ν) by $\eta(n,\nu)$:

$$\eta(n,\nu) := \arg\operatorname{lex}\max\{\theta(x) : x \in \mathcal{I}(n,\nu)\}.$$

Schmeidler [72] proved the nucleolus was unique and that it lied in ther intersection of the kernel and leastcore of the game.

Despite its intricate definition the concept of the nucleolus is surprisingly ancient. Its history can be traced back to a discussion on bankruptcy division in the Babylonian Talmud [2]. Some notable modern applications of the nucleolus include but are not limited to water supply management [1], fair file sharing on peer-to-peer networks [62], resource sharing in job assignment [77], and airport pricing [10].

Current research interest in the nucleolus stems not only from its geometric beauty [61], or several practical applications (e.g., see above or [11, 58]), but from the strange way problems of computing the nucleolus fall in the complexity landscape, seeming to straddle the NP vs P boundary.

1.2.2 Cooperative Combinatorial Optimization Games

Naively to specify a game as input to computational problem one needs to write down $\nu(S)$ for every $S \subseteq [n]$. This is a number of values which is exponential in the number of players, and for inputs of this size computing any reasonable solution concept is trivial. What is interesting is that many games value functions can be implicitly defined in terms optimizing over some combinatorial structure on the player set, just as we did with the matching game.

Notice that to give a matching game as input to a computational problem, it suffices to specify the graph G and edge weights w rather than encode every value of $\nu(S)$.

By modifying the underlying optimization problem coalitions need to solve, we can generate many examples of *combinatorial optimization games*. For instance, we can replace solving the matching linear program on the subgraph induced by a coalition with the fractional matching linear program to obtain the *fractional matching game.* We can also start with other combinatorial optimization problems, such as in *spanning tree games* where the value of $\nu(S)$ is the minimum cost[5] of a spanning tree connecting the coalition to a special vertex identified as the root.

We need not restrict ourselves to players being vertex sets. For example, given a directed graph $D = (V, E)$ with a source node s and sink node t, the *maximum flow game* is the cooperative game with player set E, and the value of a coalition is the value of a maximum st-flow using only arcs in the coalition.

In a similar fashion all manner of combinatorial optimization games can be defined. The question of finding efficient algorithms for computing the nucleolus of such games has garnered much attention from the community. This has lead to polynomial time algorithms, such as for

[5]Our usual perspective is of profit games, but this is an example of cost game, where the coalitions are incurring a cost to achieve some objective and need to decide how to share this cost.

fractional matching, cover, and clique games [15], simple flow games [67], assignment games[78], and unweighted matching games [45]. Such methods worked by analyzing the Maschler Peleg Shapley Scheme for computing the nucleolus which we will discuss in Section 2.1.2. Another approach was taken by Fleiner, Solymosi and Sziklai, who used the concept of dually essential coalitions [79] to compute the nucleolus of a large class of directed acyclic graph games [80] via the characterization set method (Granot, Granot, and Zhu [38] and independently Reinjerse and Potters [68]). Other cases have led to NP-hardness proofs, such as flow games [19], weighted voting games [25], simple b-matching games [6], and spanning tree games [29]. It is interesting to observe that the complexity of computing the nucleolus does not always correspond to the complexity of the underlying combinatorial optimization problem.

1.3 Connections

Before outlining the contribution we will cover in main body of the thesis, we would like to mention some interesting connections between matching games and other adjacent research areas.

Network bargaining games model situations closely related to matching games. They are a model for studying network exchange theory, a field of social science which studies power imbalances during bilateral negotiations between agents in a social network [16]. In a *network bargaining game* we have a b-capacitated w-weighted graph $G = (V, E)$ as in matching games. The difference from matching games is that agents negotiate how they will split the value of their shared edge if a partnership forms rather than the value being distributed globally as in cooperative games. A solution in a network bargaining game is a pair (M, z) where M is a simple b-matching and z is a non-negative vector with entries z_{uv} and z_{vu} for each edge uv in G. Where z_{uv} indicates the share that u earns from the value of edge uv and z_{vu} indicates the share that v earns. If uv is in M then $z_{uv} + z_{vu} = w_{uv}$, and otherwise $z_{uv} = z_{vu} = 0$.

A solution (M, z) to a network bargaining game is said to be *stable* if the profit an agent earns from any contract in M is at least as much as its outside option. An agent's *outside option* is the maximum profit that the agent can earn by forming a mutually beneficial contract with one of its neighbours. A solution is called *balanced* if the difference between z_{uv} and u's outside option is equal to the difference between z_{vu} and v's outside option for every edge uv in M. Kleinberg and Tardos [47] showed how to compute balanced solutions in polynomial time when b is the all-ones vector, and they showed that having a stable solution implies having a balanced solution. Farczadi, Georgiou, and Könemann [32] extended this result to network bargaining games with general vertex capacities. Bateni, Hajiaghayi, Immorlica, and Mahini [4] showed a connection between matching games and network bargaining by proving that stable outcomes coincide with the core and balanced outcomes correspond with the kernel.

In presenting the leastcore and nucleolus, the discussion was in part motivated by the possibility of the core being empty and hence that there was a need for a more sophisticated solution concept. Another approach to dealing with core emptiness in matching games would be to ask how many vertices or edges need to be removed from the underlying graph to produce an instance with a non-empty core? So called *graph stabilization* has been intensely studied in the literature, see for instance [9, 44, 49]. Chandrasekaran [14] wrote an excellent survey on the topic which we recommend for readers looking for details on subject.

Connections have also been made between matching games and stable matchings. Allocations in the core of the corresponding matching game have been found to be in 1-to-1 correspondence with stable matchings with payments, a variant of the Gale and Shapley's [34] famous stable marriage problem. See Koopmans and Beckmann [52], Shapley and Shubik [76], Eriksson and Karlander [27], and Biró, Kern, and Paulusma [7] for details.

1.4 Outline

In this thesis we study the complexity of computing solution concepts for matching games and their generalization to b-matching games, which poses significant additional challenges. In particular we focus on computing the nucleolus. Our work concentrates on computing the nucleolus over the Shapley value for two reasons. The first being that Shapley Value has already been well-studied [3], leading to a #P-completeness result for computing the Shapley Value of unweighted matching games and fully polynomial time randomized approximation scheme (FPRAS). The second reason being that the problem of computing the nucleolus of matching games had been open since at least 1998, when Faigle, Kern, Fekete, and Hochstättler [28] mention the problem in their work on the nucleon, a multiplicative-error analog to the nucleolus which they show is polynomial time computable.

In Chapter 2 we study efficient algorithms for computing the nucleolus of matching games. We describe two classical frameworks for the computing the nucleolus: the Kopelowitz Scheme [53] and the Maschler, Peleg, Shapley (MPS) Scheme [61]. In this chapter we will discuss the long history of related work on this problem. Our main contribution will be Theorem 2.0.1, which states that the nucleolus of any weighted matching game can be computed in polynomial time. We achieve this result by providing a polynomially sized description of each linear program in the MPS Scheme for matching games. This chapter is based on work in a paper with Jochen Koenemann and Kostya Pashkovich [50].

In Chapter 3 we consider the generalization of matching games to b-matching games. In view of Theorem 2.0.1 these problems are significantly harder, as computing the nucleolus of such games is NP-hard in general [6]. We will survey the complexity landscape of computing the nucleolus of b-matching games and strengthen the hardness result. In particular we will show Theorem 3.0.1, which states that deciding if an allocation is the nucleolus of a b-matching game is NP-hard even when every b-value is 3, every weight is 1, and the underlying graph is bipartite. We augment these results with some positive findings when b-values are restricted to $b \leq 2$. This includes a separation oracle for the leastcore of such b-matching games, and an algorithm for computing the nucleolus in some special cases. The main result of this chapter is based onwork with Jochen Koenemann and Felix Zhou that is pending publication.

In Chapter 4 we have two goals. The first is to identify a large class of b-matching games for which computing the nucleolus is easy. We do this in Theorem 4.0.2 which states that the nucleolus of any b-matching game whose underlying graph has bounded treewidth can be computed in polynomial time. The technique is based on the results of our second goal: devise a general framework for computing the nucleolus of a large class of cooperative games. The framework we contribute is summarized by Theorem 4.0.1 which states that for any game where the minimum excess coalition for a given allocation can be found using a dynamic programming technique, the nucleolus can be computed in time proportional to the size of that dynamic program. This technique is inspired by the work of Pashkovich [63] on weighted voting games. The work in this chapter is based on a joint paper with Jochen Koenemann [51].

Chapter 2

Computing the Nucleolus of Matching Games

Beyond being one of the most fundamental problems in combinatorial optimization, starting with the founding work of Kuhn on the Hungarian method for the assignment problem [54], matching problems have historically teetered on the cusp of hardness. For example, prior to Edmonds' celebrated Blossom Algorithm [23, 22] it was not clear whether Maximum Matching belonged to P. For another example, until Rothvoß' landmark result [70] it was thought that the matching polytope could potentially have polynomial extension complexity.

In cooperative game theory, matchings live up to their historical pedigree of representing a challenging problem class. The long standing open problem in this area was whether the nucleolus of a weighted matching game instance can be computed in polynomial time. The question was posed as an important open problem in multiple papers. In 1998, Faigle, Kern, Fekete, and Hochstättler [28] mention the problem in their work on the nucleon. Kern and Paulusma state the question of computing the nucleolus for general matching games as an important open problem in 2003 [45]. In 2008, Deng and Fang [18] conjectured this problem to be NP-hard, and in 2017 Biró, Kern, Paulusma, and Wojuteczky [8] reaffirmed this problem as an interesting open question. Theorem 2.0.1 settles the question, providing a polynomial-time algorithm to compute the nucleolus of a general instance of a weighted cooperative matching game.

Theorem 2.0.1 *Given a graph $G = (V, E)$ and weights $w : E \to \mathbb{R}$, the nucleolus $\eta(|V|, \nu)$ of the corresponding weighted matching game can be computed in polynomial time.*

Prior to our work, the nucleolus was known to be polynomial-time computable only in structured instances of the matching game. Solymosi and Raghavan [78] showed how to compute the nucleolus in an (unweighted) assignment game instance in polynomial time, i.e. on instances of the matching game on bipartite graphs. Kern and Paulusma [45] later provided an efficient algorithm to compute the nucleolus in general unweighted matching game instances. Paulusma [64] extended the work in [45] and gave an efficient algorithm to compute the nucleolus in matching games where edge weights are induced by node potentials. Farczadi [31] finally extended Paulusma's framework further using the concept of *extendible allocations*. We note also that it is easy to compute the nucleolus in weighted instances of the matching game with non-empty core. For such instances, the leastcore has a simple compact description that does not include constraints for coalitions of size greater than 2. Thus it is relatively straightforward to adapt the iterative algorithm of Maschler [61] to a polynomial-time algorithm for computing the nucleolus, as we will describe towards the end of Section 2.1. Such an algorithm for computing the nucleolus would rely on the ellipsoid method. Circumventing this, Biró, Kern, and Paulusma [7] gave a combinatorial algorithm for computing the nucleolus of matching games with non-empty core.

In Section 2.1 we will describe classical techniques for computing the nucleolus of a cooperative game. These techniques proceed by a hierarchy of linear programs, where the solution of each linear program in the sequence depends on the solution of the previous linear program in the sequence. When the core of a matching game is non-empty, it is well-known and straightforward to compute the nucleolus using the latter of these schemes, the Maschler Peleg Shapley (MPS) Scheme, as we will discuss.

The first linear progam in both schemes for computing the nucleolus which we will describe is

$$P_1 := \max\{\varepsilon : x \in P_1(\varepsilon)\}. \tag{P_1}$$

The optimal value of (P_1) is ε_1. For matching games it is well known how to solve this linear program.

Our approach to proving Theorem 2.0.1 is to provide a compact (polynomially sized) description of each feasible region polytope in the MPS Scheme. While there are a linear number of linear programs in the sequence, their naive implementation requires an exponential number of constraints. While it is known how to separate over the feasible region of (P_1), the challenge lies in solving all successive linear programs in the sequence. Previous results in [45, 64] made use of the *unweighted* nature of node-weighted instances of matching games, and were able to employ the *Edmonds-Gallai* structure theorem to derive compact formulations for the LPs in the MPS hierarchy. We do not know how to extend this line of work beyond node-weighted instances. In our work, we identify a minimal family of tight excess constraints of (P_1) corresponding to coalitions whose vertices are saturated by so called *universal matchings*. A matching is universal if it saturates a coalition of vertices that is tight for all allocations in the leastcore. As we will show, universal matchings are the optimal matchings for carefully chosen cost functions derived from so called *universal allocations* (see Section 2.2.4).

From there we rely on well-known characterizations of extreme points of matching polyhedra, and the fact that these are defined by laminar families of blossom constraints. Ultimately, the above allows us to obtain a decomposition of the input graph into the edges on either the inside or outside of blossoms. The structure of the associated optimum face of the matching polytope (e.g., see Schrijver [73]) elucidates the structure of excess over all coalitions in the matching game. Our proof uses a critical insight into the symmetric nature of exchange on the nodes of a blossom as we move between leastcore allocations (see Lemma 2.2.8).

We present the details of the leastcore LP in Section 2.2. There we introduce the concept of *universal matchings* which are fundamental to our approach, and give a compact formulation for the leastcore linear program, (P_1). We also present our main technical lemma, Lemma 2.2.8, which provides a crucial symmetry condition on the values allocations can take over the vertices of blossoms in the graph decomposition we use to describe the compact formulation. In Section 2.3 we describe the successive linear programs in Maschler's Scheme and provide a compact formulation for each one in a matching game. We show how this formulation can be used in Maschler's framework to compute the nucleolus in Section 2.3.

2.1 Schemes For Computing the Nucleolus

2.1.1 Kopelowitz Scheme

The first scheme for computing the nucleolus is due to Kopelowitz [53]. It is defined as follows.

Consider a cooperative game (n, ν). Define a sequence of linear programs (K_i) for $i = 1, \dots, N$. The first linear program K_1 will be equal to (P_1) as defined earlier. The feasible region of each linear program will be a subset of $\mathbb{R}^n \times \mathbb{R}$. The optimal value of K_i will be denoted ε_i^K, and the set of $x \in \mathbb{R}^n$ such that (x, ε) is feasible for K_i will be denoted $K_i(\varepsilon)$.

The remaining linear programs in the sequence are defined recursively

$$\begin{aligned} &\max\ \varepsilon \\ &\text{s.t.}\ \ \operatorname{ex}(x, S) \geq \varepsilon && \forall S \subseteq [n], S \notin \bigcup_{j=1}^{i-1} \operatorname{Tight}(K_j(\varepsilon_j^K)) \\ &\qquad x \in K_{i-1}(\varepsilon_{i-1}^K). \end{aligned}$$

Here for each $j = 1, \dots, N$,

$$\operatorname{Tight}(K_j(\varepsilon_j^K)) = \{S \subseteq [n] : \operatorname{ex}(x, S) = \varepsilon_j^K, \text{for all } x \in K_j(\varepsilon_j^K)\}.$$

We call coalitions in the above set, the *tight* coalitions for K_j.

The sequence terminates, i.e. we obtain K_N, when the feasible region of the current linear program is a singleton. In other words, when $\bigcup_{j=1}^{i-1} \operatorname{Tight}(K_i(\varepsilon_i^K)) = 2^{[n]}$, the set of all coalitions.

It is immediate that N is finite, since at least one coalition is add to the tight set in every round of Kopelowitz Scheme. It is also clear from the definitions that $K_N(\varepsilon_N^K)$ contains $\eta(n, \nu)$, the nucleolus of the game. The issue is that it may take an exponential number of rounds for Kopelowitz Scheme to terminate, as the following example shows.

Example: This example from Greco, Malizia, Palopoli, Scarcello [39] shows that there exist cooperative games where Kopelowitz Scheme runs in an exponential number of rounds.

For any $n \geq 3$, let (n, ν) be the cooperative game defined as follows. Let $m = n-2$ and fix an arbitrary ordering over the subsets of $\{W \subseteq [m] : 2 \leq |W| < m\}$, labelling the sets $W_1, \ldots, W_h$ where $h = 2^m - m - 2$.

For any $S \subset [n]$, we define

$$\nu(S) := \begin{cases} m+2, & \text{if } S = [n] \\ m, & \text{if } S = [m] \\ |W_i| - 1 + 2^{-i}, & \text{if } S = W_i, \text{ for } i \in [h] \\ 2, & \text{if } S = [n]\backslash[m] \\ 1, & \text{if } S = \{j\} \text{ for some } j \in [m] \\ 0, & \text{if } S = \{m+1\} \text{ or } S = \{m+2\} \\ -2, & \text{otherwise.} \end{cases}$$

This definition may appear complex, but it is in service of creating a simple effect: for each $i \in [h]$ there is a round of Kopelowitz Scheme where W_i is the only coalition added to the tight sets.

First observe that if $x \in K_1(\varepsilon_1^K)$ then x is an imputation by definition, and so $x_j \geq 1$ for all $j \in [m]$ and $x_{m+1}, x_{m+2} \geq 0$ by individual rantionality. Further, by efficiency $x([n]) = m+2$. Since $x([m]) \geq m + \varepsilon_1^K$, and $x(\{m+1, m+2\}) \geq 2 + \varepsilon_1^K$, this implies that $\varepsilon_1^K = 0$. Thus we see that

$$K_1(\varepsilon_K^1) = \{x \in \mathbb{R}^n : x_1 = \cdots = x_m = 1, x_{m+1} \geq 0, x_{m+2} \geq 0, x_{[n]\backslash[m]} = 2\}.$$

Therefore $\text{Tight}(K_1(\varepsilon_1^K)) = \{\{1\}, \ldots, \{m\}, [m], \{m+1, m+2\}\}$.

Now, for each $i \in [h]$,

$$\text{ex}(x, W_i) = x(W_i) - \nu(W_i) = |W_i| - |W_i| + 1 - 2^{-i} = 1 - 2^{-i}.$$

Since the value of x_j is fixed to 1 after the first round for each $j \in [m]$, each subsequent round of Kopelowitz Scheme adds one coalition from $W_1, \ldots, W_h$ to the tight sets in that order. Thus taking at least h rounds, i.e. a number of rounds on the order of an exponential in n, to find the nucleolus. ■

2.1.2 Maschler, Peleg, Shapley Scheme

In the last subsection we saw that Kopelowitz Scheme takes an exponential number of rounds to find the nucleolus of a cooperative game in the worst case. The scheme we present in this subsection improves upon that bound dramtically, taking a linear number of rounds in the worst case. It is the well-known *Maschler Peleg Shapley (MPS) Scheme* [53, 61].

Notice how in the worst case example for the Kopelowitz Scheme, an exponential number of rounds were needed, but in each of those rounds no change was being made to the total amount allocated to the coalitions entering the tight set? The key idea in the MPS Scheme is to only consider coalitions in subsequent rounds for which the total value assigned to a coalition is not yet fixed over the potential allocations which are still feasible. Toward this end, for a polytope $Q \subseteq \mathbb{R}^n$, we define $\text{Fix}(Q)$ as follows

$$\text{Fix}(Q) := \{S \subseteq [n] : \exists c_S \in \mathbb{R}, x(S) = c_S \text{ for all } x \in Q\}.$$

We call the coalitions in $\text{Fix}(Q)$ the *fixed* coalitions. Observe that for the polytopes appearing Kopelowitz Scheme, $\text{Tight}(K_i(\varepsilon_i^K)) \subseteq \text{Fix}(K_i(\varepsilon_i))$.

Consider a cooperative game (n, ν). Similar to how we defined Kopelowitz Scheme, we define a sequence of linear programs (P_i) for $i = 1, \ldots, N$ whose feasible regions are subsets of $\mathbb{R}^n \times R$. The first linear program will be equal to (P_1) as defined earlier. The optimal value of P_i will be denoted ε_i, and the set of $x \in \mathbb{R}^n$ such that (x, ε) is feasible for P_i will be denoted $P_i(\varepsilon)$.

The remaining linear programs in the sequence are defined recursively

$$\begin{aligned} \max\ & \varepsilon & & \quad (P_i) \\ \text{s.t.}\ & \text{ex}(x, S) \geq \varepsilon & & \forall S \subseteq [n], S \notin \text{Fix}(P_{i-1}(\varepsilon_{i-1})) \\ & x \in P_{i-1}(\varepsilon_{i-1}). & & \end{aligned}$$

As with the Kopelowitz Scheme, the sequence terminates when the feasible region of the current linear program is a singleton. In other words, when $\text{Fix}(P_i(\varepsilon_i)) = 2^{[n]}$.

To see that the MPS Scheme terminates in $O(n)$ rounds, consider a coalition $S \subseteq [n]$ and observe that if $\chi(S)$ is in the linear span of the incidence vectors of coalitions in $\text{Fix}(P_i(\varepsilon_i))$ then S is in $\text{Fix}(P_i(\varepsilon_i))$. Therefore in each round i, there exists $S \subseteq [n]$ such that $x(S) - \nu(S) = \varepsilon_i$ and $\chi(S)$ is not in the linear span of incidence vectors of coaltions in $\text{Fix}(P_{i-1}(\varepsilon_{i-1}))$. Hence after at most n rounds the incidence vectors of the coalitions in the tight set span $\mathbb{R}^n$, at which point the next polytope in the sequence has a unique solution, the nucleolus.

When the core is non-empty, it is easy to compute the nucleolus of a matching game in polynomial time using the MPS Scheme. We will explain how by giving the proof of this well-known theorem. Although more technically complex, our result for the core empty case will follow the same high-level structure and so it is worth going over in detail.

Theorem 2.1.1 *([64]) Let $G = (V, E)$ be a w-weighted graph and let (n, ν) be the matching game on G. Suppose that the core of (n, ν) is non-empty. Then $\eta(n, \nu)$ can be computed in polynomial time in $|E|$ and the encoding size of w.*

Proof: First notice that since the core non-empty, $P_1(\varepsilon_1) = \mathcal{C}(n, \nu)$, the core of the game. Indeed, core non-emptiness implies $\varepsilon_1 \geq 0$. To see that $\varepsilon_1 \leq 0$ observe that if $\varepsilon_1 > 0$ then $x(M) > w(M)$ for all matchings M. In particular this would hold for the maximum weight matching in G, implying that $x(V) > \nu(V)$, a contradiction to efficiency.

We claim that the core has a polynomially sized descripition of the form

$$\begin{aligned}\mathcal{C}(n, \nu) = \{x \in \mathbb{R}^n : &\operatorname{ex}(x, e) \geq 0, && \forall e \in E\\ &x(V) = \nu(V)\\ &x \geq 0\}.\end{aligned}$$

If this claim holds then we can use any polynomial time linear programming algorithm, such as the ellipsoid method for instance, to compute allocations in the core in polynomial time and thus solve the first linear program in the MPS Scheme.

To see the claim holds it suffices to show for any imputation x, $x(uv) \geq w(uv)$ for all $uv \in E$ if and only if $x(S) \geq \nu(S)$ for all $S \subseteq V$. The 'if' direction is immediate since $\nu(\{u, v\}) = w(uv)$. For the 'only if' direction, let $S \subseteq V$ and let $M \subseteq E(G[S])$ be a matching such that $w(M) = \nu(S)$.

Since $x \geq 0$ by individual rationality, $x(S) \geq x(V(M))$ and therefore, by summing $x(e) \geq w(e)$ over the edges of M,

$$x(S) \geq w(M) = \nu(S).$$

Thus the claim holds.

Now we need to give a compact description of the higher round polytopes in the MPS Scheme. We claim that for any $\varepsilon \geq 0$, and $i \geq 2$,

$$\begin{aligned}P_i(\varepsilon) = \hat{P}_i(\varepsilon) := \{x \in \mathbb{R}^n : &\operatorname{ex}(x, e) \geq \varepsilon, && \forall e \in E \backslash \operatorname{Fix}(P_{i-1}(\varepsilon_{i-1}))\\ &x_v \geq \varepsilon, && \forall v \in V, \{v\} \notin \operatorname{Fix}(P_{i-1})(\varepsilon_{i-1})\\ &x \in P_{i-1}(\varepsilon_{i-1})\}\end{aligned}$$

The fact that $P_i(\varepsilon)$ is contained in $\hat{P}_i(\varepsilon)$ is immediate since $\hat{P}_i(\varepsilon_i)$ is a relaxation of $P_i(\varepsilon)$. For the reverse inclusion consider $x \in \hat{P}_i(\varepsilon)$. It remains to show that $\operatorname{ex}(x, S) \geq \varepsilon$ for all $S \subseteq [n], S \notin \operatorname{Fix}(P_{i-1}(\varepsilon_{i-1}))$.

Let $S \subseteq [n]$ such that $S \notin \operatorname{Fix}(P_{i-1}(\varepsilon_{i-1}))$. Let $M \subseteq E(G[S])$ be a matching such that $\nu(S) = w(M)$. Since S is not fixed, either there exists $v \in S$ such that $\{v\}$ is not fixed or there exists $e \in M$ such that e is not fixed.

If there exists $v \in S \backslash V(M)$ such that $\{v\} \notin \operatorname{Fix}(P_{i-1}(\varepsilon_{i-1}))$ then since x is in the core,

$$\operatorname{ex}(x, S) \geq x_v + x(V(M)) - w(M) \geq x_v \geq \varepsilon$$

as desired. Otherwise, there exists $e \in M$ such that $e \notin \operatorname{Fix}(P_{i-1}(\varepsilon_{i-1}))$. In this case, since x is in the core,

$$\operatorname{ex}(x, S) \geq \operatorname{ex}(x, V(M)) = \operatorname{ex}(x, e) + \operatorname{ex}(x, V(M-e)) \geq \varepsilon.$$

This proof also shows the optimal value of $\max\{\varepsilon : \hat{P}_i(\varepsilon) \neq \emptyset\}$ is equal to ε_i. Thus we can replace $P_{i-1}(\varepsilon_{i-1})$ with $\hat{P}_{i-1}(\varepsilon_{i-1})$ in the description of $\hat{P}_i(\varepsilon_i)$ and thus obtain a compact description of each linear program in the MPS Scheme for matching games with non-empty core. ■

From the compact formulation of the core of a matching game, one can see that $\varepsilon_1 = 0$ if and only if the value of a maximum weight matching in G with weights w equals the value of a maximum weight fractional matching. This follows since $x \in P_1(\varepsilon_1)$ is a fractional weighted node cover of value $\nu(V)$ when $\varepsilon_1 = 0$.

2.2 Leastcore Formulation

2.2.1 Leastcore and Core of Matching Games

The leastcore linear program (P_1) can be rewritten equivalently as

$$\begin{aligned} \max\ & \varepsilon \\ \text{s.t. } & x(M) \geq w(M) + \varepsilon \qquad \text{for all} \quad M \in \mathcal{M} \\ & x(V) = \nu(V) \\ & x \geq \mathbb{0}\,, \end{aligned} \tag{P_1}$$

where $\mathcal{M}$ is the set of all matchings M on G, and $x(M)$ is a shorthand for $x(V(M))$. To see this, since $x \geq 0$ observe that for any $S \subseteq V$,

$$x(S) - \nu(S) \geq x(M) - w(M)$$

where M is a matching in $G[S]$ for which $w(M) = \nu(S)$.

The separation problem for the linear program (P_1) can be reduced to finding a maximum weight matching in the graph G with edge weights $w(uv) - x_u - x_v$, $uv \in E$. Since the maximum weight matching can be found in polynomial time [22], we know that the linear program (P_1) can be solved in polynomial time as well via the ellipsoid method.

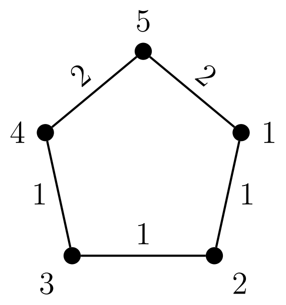

Figure 2.1: A matching game with empty core

When the core is non-empty we saw in Theorem 2.1.1 how to compute the nucleolus. Our contribution is to give an algorithm for computing the nucleolus when the core is empty. As the following example shows, this situation can occur when very simple structures are present in our input graph. We also give a calculation of the nucleolus of this game as a concrete example.

Example:

Consider the graph in Fig. 2.1. This graph $G = (V, E)$ is a 5-cycle with two adjacent edges 15 and 45 of weight 2, and the remaining three edges of weight 1. Since the maximum weight matching value is $\nu(G) = 3$, but the maximum weight fractional matching value is $7/2$, the core of this game is empty. The allocation x^* defined by

$$x^*(1) = x^*(2) = x^*(3) = x^*(4) = \frac{2}{5} \quad \text{and} \quad x^*(5) = \frac{7}{5}$$

lies in the leastcore. Each edge has the same excess, $-1/5$, and any coalition of four vertices yields a minimum excess coalition with excess $-2/5$. Hence the leastcore value of this game is $\varepsilon_1 = -2/5$.

In fact, we can see that x^* is the nucleolus of this game. To certify this we can use the result of Schmeidler [71] that the nucleolus lies in the intersection of the leastcore and the prekernel. For this example, the prekernel condition is that for all $i \neq j \in V$,

$$\max_{S \subseteq V \setminus \{j\}} x(S \cup \{i\}) - \nu(S \cup \{i\}) = \max_{S \subseteq V \setminus \{i\}} x(S \cup \{j\}) - \nu(S \cup \{j\}).$$

This condition reduces to the condition that the excess values of non-adjacent edges are equal. Since G is an odd cycle, this implies that all edges have equal excess, i.e.

$$\text{ex}(x, 12) = \text{ex}(x, 23) = \text{ex}(x, 34) = \text{ex}(x, 45) = \text{ex}(x, 15).$$

Combining the four equations above with the leastcore condition that $x(V) = \nu(G)$ we obtain a system of equations with the unique solution x^*. Hence the intersection of the leastcore and prekernel is precisely $\{x^*\}$, and so by Schmeidler, x^* is the nucleolus. ■

Now we will work towards providing a polynomial-size description of (P_1). For the remainder of this chapter we assume that the cooperative matching game (n, ν) determined by w-weighted graph G has an empty core, as computing the nucleolus is otherwise well-known to be solvable in polynomial time [64].

2.2.2 Universal Matchings, Universal Allocations

For each $x \in P_1(\varepsilon_1)$ we say that a matching $M \in \mathcal{M}$ is an x-tight matching whenever $\text{ex}(x, M) = \varepsilon_1$. Note that $\text{ex}(x, M)$ is a shorthand for $\text{ex}(x, V(M))$. We denote by $\mathcal{M}^x$ the set of x-tight matchings.

A *universal matching* $M \in \mathcal{M}$ is a matching which is x-tight for all $x \in P_1(\varepsilon_1)$. We denote the set of universal matchings in G by $\mathcal{M}_{uni}$. A *universal allocation* $x^* \in P_1(\varepsilon_1)$ is a leastcore point whose x^*-tight matchings are precisely the set of universal matchings, i.e. $\mathcal{M}^{x^*} = \mathcal{M}_{uni}$.

Lemma 2.2.1 *There exists a universal allocation $x^* \in P_1(\varepsilon_1)$.*

Proof: Indeed, it is straightforward to show that every x^* in the relative interior of $P_1(\varepsilon_1)$ is a universal allocation. If the relative interior is empty then $P_1(\varepsilon_1)$ is a single point, which trivially contains a universal allocation. For the sake of completeness we now provide a combinatorial proof that universal allocations exist.

Let $\mathcal{M}^x$ denote the set of x-tight matchings for some $x \in P_1(\varepsilon_1)$. That is $\mathcal{M}^x := \{M \in \mathcal{M} : \text{ex}(x, M) = \varepsilon_1\}$. We claim that there exists $x^* \in P_1(\varepsilon_1)$ such that

$$\mathcal{M}^{x^*} = \bigcap_{x \in P_1(\varepsilon_1)} \mathcal{M}^x .$$

Let $x^* \in P_1(\varepsilon_1)$ be chosen to minimize $|\mathcal{M}^{x^*}|$. Suppose for a contradiction that x^* is not universal. Then there exists $x \in P_1(\varepsilon_1)$ and $M \in \mathcal{M}^{x^*} \backslash \mathcal{M}^x$.

Let $\bar{x} := \frac{1}{2}(x^* + x)$. Since $P_1(\varepsilon_1)$ is a convex set, $\bar{x} \in P_1(\varepsilon_1)$. Furthermore, $\mathcal{M}^{\bar{x}} = \mathcal{M}^{x^*} \cap \mathcal{M}^x$, and thus $\mathcal{M}^{\bar{x}} \subseteq \mathcal{M}^{x^*} \backslash \{M\}$ contradicting the minimality of $|\mathcal{M}^{x^*}|$. ■

Lemma 2.2.1 tells us universal allocations exist. The following lemma, Lemma 2.2.2, tells us how to compute them in polynomial time. This lemma implies Lemma 2.2.1, but the pure existence proof is more elegant and gives the idea behind the algorithm which proves Lemma 2.2.2.

Lemma 2.2.2 *A universal allocation $x^* \in P_1(\varepsilon_1)$ can be computed in polynomial time.*

Proof: A point x^* in the relative interior of $P_1(\varepsilon_1)$ can be found in polynomial time using the ellipsoid method 1.1.6. Since any allocation x^* from the relative interior of $P_1(\varepsilon_1)$ is a universal allocation, this implies the statement of the lemma.

In Section 2.2.3 we provide a self-contained proof of this Lemma. ■

2.2.3 Computing a Universal Allocation

For the sake of completeness, we prove Lemma 2.2.2 by providing an algorithm for computing a universal allocation x^*. Consider the following algorithm:

```
1:  Compute $x \in P_1(\varepsilon_1)$.
2:  $F$ ← face of matching polytope maximizing function defined by $w(uv) - x(u) - x(v)$, $uv \in E$.
3:  Compute $M \in \mathcal{M}$, such that $\chi(M)$ is a vertex of $F$.
4:  Initialize $K \leftarrow \varnothing$.
5:  loop
6:      $y \leftarrow \arg\max\{y(M) : y \in P_1(\varepsilon_1)\}$.
7:      if $y(M) > x(M)$ then
8:          $x \leftarrow \frac{1}{2}(x + y)$
9:          Let $F$ be the face of the matching polytope maximizing the linear function defined by
            weights $w(uv) - x(u) - x(v)$, $uv \in E$.
10:     else {$M$ is a universal matching}
11:         $K \leftarrow K \cup \{\chi(M)\}$
12:         Compute $M$, such that $\chi(M) \notin \text{span}(K)$ is a vertex of $F$. If no such $M$ exists, break
            Loop.
13:     end if
14: end loop
15: return $x$.
```

Theorem 2.2.3 *The allocation x^* returned in Step 15 of the above algorithm is a universal allocation.*

Proof: Suppose that x^* is not a universal allocation, then there exists a non-universal x^*-tight matching M; i.e., there exists $y \in P_1(\varepsilon_1)$ such that M is not y-tight.

Since $\chi(M) \in \text{span}(K)$, we know that

$$\chi(M) = \sum_{j=1}^{p} \alpha_j \chi(M_j)\,,$$

for some $\alpha_j \in \mathbb{R}$, $j \in [p]$, where $K = \{\chi(M_1), \ldots, \chi(M_p)\}$. Recall, that all matchings $M_1, \ldots, M_p$ are universal matchings, hence

$$x^*(M_j) = y(M_j) = \varepsilon_1 + w(M_j)$$

for all $j \in [p]$. It follows that

$$\varepsilon_1 + w(M) = x^*(M) = \sum_{j=1}^{p} \alpha_j x^*(M_j) = \sum_{j=1}^{p} \alpha_j y(M_j) = y(M)\,,$$

contradicting the fact that M is x^*-tight but not y-tight. ■

Theorem 2.2.4 *The algorithm terminates in polynomial time.*

Proof: Indeed, if $y(M) > x(M)$ in Step 7 of the algorithm then Step 9 leads to a dimension reduction for face F. This dimension reduction follows since the vertices of the old F correspond to the set of x-tight matchings, whereas the vertices of the new F correspond to the set of $\frac{1}{2}(x+y)$-tight matchings. Since $y(M) > x(M)$, the $\frac{1}{2}(x+y)$-tight matchings are a strict subset of the x-tight matchings, and hence the new F is a strict subface of the old F.

If on the other hand $y(M) \leq x(M)$ then the dimension of K as defined in Step 11 increases because of our choice of M in Step 12. The upper bound on the dimensions of the linear space $\text{span}(K)$ and the face F is $|E|$, while the lower bound on their dimensions is 0.

Since we can separate over $P_1(\varepsilon_1)$ in polynomial time we can solve the optimization problem in Step 12 in polynomial time. Moreover, since we can separate over F in polynomial time, we can decide if F has a vertex outside the linear space $\text{span}(K)$ in polynomial time by solving two optimization problems for each of the (at most $|E|$) equations defining $\text{span}(K)$: maximizing and minimizing the corresponding (normal to $\text{span}(K)$) vector over F, and checking if both optimal values are zero. Since each step of the algorithm runs in polynomial time, this implies that the algorithm terminates in polynomial time. Hence, the statement of the theorem follows. ■

Given a non-universal allocation x and a universal allocation x^*, we observe that $\mathcal{M}^{x^*} \subset \mathcal{M}^x$ and so $\theta(x^*)$ is strictly lexicographically greater than $\theta(x)$. Thus the nucleolus is a universal allocation. We emphasize that $\mathcal{M}^{x^*} = \mathcal{M}_{uni}$ is invariant under the (not necessarily unique) choice of universal allocations x^*. Henceforth we fix a universal allocation $x^* \in P_1(\varepsilon_1)$.

2.2.4 Description of the Convex Hull of Universal Matchings

By the definition of universal allocation x^*, a matching M is universal if and only if it is x^*-tight. Thus, M is a universal matching if and only if its characteristic vector lies in the optimal face of the matching polytope corresponding to (the maximization of) the linear objective function assigning weight $-\text{ex}(x^*, uv) = w(uv) - x^*(u) - x^*(v)$ to each edge $uv \in E$. Let $\mathcal{O}$ be the set of node sets $S \subseteq V$ such that $|S| \geq 3$, $|S|$ is odd. Edmonds [22] gave a linear description of the matching polytope of G as the set of $y \in \mathbb{R}^E$ satisfying:

$$\begin{array}{rcll} y(\delta(v)) & \leq & 1 & \text{for all} \quad v \in V \\ y(E(S)) & \leq & (|S|-1)/2 & \text{for all} \quad S \in \mathcal{O} \\ y & \geq & 0\,. & \end{array}$$

Thus, a matching $M \in \mathcal{M}$ is universal if and only if it satisfies the constraints

$$\begin{array}{rcll} M \cap \delta(v) & = & 1 & \text{for all} \quad v \in W \\ M \cap E(S) & = & (|S|-1)/2 & \text{for all} \quad S \in \mathcal{L} \\ M \cap \{e\} & = & 0 & \text{for all} \quad e \in F\,, \end{array} \tag{2.1}$$

where W is some subset of V, $\mathcal{L}$ is a subset of $\mathcal{O}$, and F is a subset of E. Using an uncrossing argument, as in [57, Pages 141-150], we may assume that the collection of sets $\mathcal{L}$ is a laminar family of node sets; i.e., for any two distinct sets $S, T \in \mathcal{L}$, either $S \cap T = \varnothing$ or $S \subseteq T$ or $T \subseteq S$. For completeness we include the most important lemma behind the uncrossing here.

Lemma 2.2.5 *Let $S, T \in \mathcal{O}$ such that $S \cap T \neq \varnothing$. Consider the face of the matching polytope of G defined by* $\operatorname{conv}\{\chi(M) : M \in \mathcal{M}_{uni}\}$. *Let y be a point in such a face. If*

$$y(E(S)) = \frac{|S|-1}{2} \quad \text{and} \quad y(E(T)) = \frac{|T|-1}{2}$$

which is to say that S and T are tight constraints of the matching polytope with respect to y, then either

(i) the sets $S \cap T, S \cup T$ are tights sets in $\mathcal{O}$ and

$$\chi(E(S)) + \chi(E(T)) = \chi(E(S \cap T)) + \chi(E(S \cup T)),$$

(ii) or the sets $S \backslash T, T \backslash S$ are tight sets in $\mathcal{O}$, the degree constraints for each vertex in $S \cap T$ are tight, and

$$\chi(E(S)) + \chi(E(T)) = \chi(S \backslash T) + \chi(T \backslash S) + \sum_{v \in S \cap T} \chi(\delta(v)).$$

Proof: First suppose that $|S \cap T|$ is odd. In this case we will show (i). We have that $|S \cup T|$ is odd, since $|S| + |T| = |S \cap T| + |S \cup T|$. Thus we have

$$\begin{aligned} \frac{|S|-1}{2} + \frac{|T|-1}{2} &= y(E(S)) + y(E(T)) \\ &\leq y(E(S \cap T)) + y(E(S \cup T)) \qquad\qquad \text{(by supermodularity)} \\ &\leq \frac{|S \cap T|-1}{2} + \frac{|S \cup T|-1}{2} \\ &= \frac{|S|-1}{2} + \frac{|T|-1}{2}. \end{aligned}$$

Therefore each inequality above holds with equality. By the second inequality holding at equality both $S \cap T$ and $S \cup T$ are tight. The first inequality holding at equality implies there are no edges between $S \backslash T$ and $T \backslash S$. Thus

$$\chi(S) + \chi(T) = \chi(S \cap T) + \chi(S \cup T).$$

Now suppose $|S \cap T|$ is even. In this case will show (ii). We have that $|S \backslash T|$ is odd and $|T \backslash S|$ is odd since $|S|$ and $|T|$ are odd. Thus we have

$$\begin{aligned}
\frac{|S|-1}{2} + \frac{|T|-1}{2} &= y(E(S)) + y(E(T)) \\
&\leq y(E(S \backslash T)) + y(E(T \backslash S)) + \sum_{v \in S \cap T} y(\delta(v)) \\
&\leq \frac{|S \backslash T|-1}{2} + \frac{|T \backslash S|-1}{2} + |S \cap T| \\
&= \frac{|S|-1}{2} + \frac{|T|-1}{2}.
\end{aligned}$$

The second inequality above follows from the degree and odd set constraints. The second inequality holding at equality implies that $S \backslash T$, $T \backslash S$, and $\delta(v)$ are tight for each $v \in S \cap T$. The first inequality holding at equality implies there are no edges between $S \cap T$ and $V \backslash (S \cup T)$. Therefore

$$\chi(S) + \chi(T) = \chi(S \backslash T) + \chi(T \backslash S) + \sum_{v \in S \cap T} \chi(\delta(v)).$$

■

We will now show a crucial lemma demonstrating that no node is covered by every universal matching.

Lemma 2.2.6 *For every node $v \in V$ there exists $M \in \mathcal{M}_{uni}$ such that v is exposed by M. Hence, $W = \varnothing$.*

Proof: Assume for a contradiction that there exists a node $v \in V$ such that $v \in W$.

First, note that there always exists a non-universal matching $M \in \mathcal{M} \setminus \mathcal{M}_{uni}$ since otherwise the empty matching would be universal, and thus

$$0 = x^*(\varnothing) = w(\varnothing) + \varepsilon_1,$$

implying that the core of the given matching game instance is non-empty.

Suppose first that there exists a node $u \in V$ exposed by some matching $M' \in \mathcal{M}_{uni}$ such that $x^*_u > 0$. Our strategy is to shift a small amount of allocation from u to v while preserving feasibility. Shifting by any sufficiently small positive δ will work. In what follows we explicitly define a value for δ for concreteness. We define

$$\delta_0 := \min\{\text{ex}(x^*, M) - \varepsilon_1 : \quad M \in \mathcal{M} \setminus \mathcal{M}_{uni}\}.$$

Recall that $\mathcal{M}_{uni}$ is the set of maximum weight matchings in G with respect to the node weights $w(uv) - x^*(uv)$, $uv \in E$, i.e. $\mathcal{M}_{uni}$ is the set of x^*-tight matchings. Moreover, recall that $\text{ex}(x^*, M) = \varepsilon_1$ for $M \in \mathcal{M}_{uni}$. Thus, we have $\delta_0 > 0$.

We define $\delta := \min\{\delta_0, x^*_u\} > 0$ and a new allocation x' as follows:

$$x'_r := \begin{cases} x^*_r + \delta, & \text{if } r = v \\ x^*_r - \delta, & \text{if } r = u \\ x^*_r, & \text{otherwise.} \end{cases}$$

Since all universal matchings contain v, the excess with respect to x' of any universal matching is no smaller than their excess with respect to x^*. Therefore, by our choice of δ, (x', ε_1) is a feasible, and hence optimal, solution for (P_1). But M' is not x'-tight, since M' covers v and exposes u. This contradicts that M' is a universal matching.

Now consider the other case: for all $u \in V$ if u is exposed by a universal matching then $x^*_u = 0$. Then, for every universal matching $M \in \mathcal{M}_{uni}$ we have

$$\varepsilon_1 = \mathrm{ex}(x^*, M) = x^*(V) - w(M) = \nu(G) - w(M).$$

Since $\nu(G)$ is the maximum weight of a matching in G with respect to the weights w, we get that $\varepsilon_1 \geq 0$. Thus x^* is in the core, contradicting our assumption that the core is empty. ■

2.2.5 Description of Leastcore

We denote inclusion-wise maximal sets in the family $\mathcal{L}$ as $S^*_1, S^*_2, \ldots, S^*_k$. We define the edge set E^+ to be the set of edges in G such that at most one of its nodes is in S^*_i for every $i \in [k] := \{1, \ldots, k\}$, i.e.

$$E^+ := E \setminus \big(\bigcup_{i=1}^{k} E(S^*_i)\big).$$

Lemma 2.2.7 *For every choice of $v_i \in S^*_i$, $i \in [k]$, there exists a universal matching $M \in \mathcal{M}_{uni}$ such that the node set covered by M is as follows*

$$\bigcup_{i=1}^{k} S^*_i \setminus \{v_i\}\,.$$

Proof: By Lemma 2.2.6, we know that for every $i \in [k]$ there exists a universal matching $M_{v_i} \in \mathcal{M}_{uni}$ such that v_i is exposed by M_{v_i}. Now, for every $i \in [k]$, let us define

$$M_i := E(S^*_i) \cap M_{v_i}.$$

Since M_i satisfies all laminar family constraints in $\mathcal{L}$ for subsets of S_i^* we have that

$$\bigcup_{i=1}^{k} M_i$$

is a matching satisfying all the constraints (2.1), and hence is a universal matching covering the desired nodes. ■

Representative Universal Matching For each $i \in [k]$ fix a unique *representative* node $v_i^* \in S_i^*$. By Lemma 2.2.7, there exists a universal matching M^* covering precisely $\bigcup_{i\in[k]} S_i^* \backslash \{v_i^*\}$. For any $x \in P_1(\varepsilon_1)$ and $S \subseteq V$ we use $\text{diff}(x, S)$ to denote

$$\text{diff}(x, S) := x(S) - x^*(S)\,.$$

For single nodes we use the shorthand $\text{diff}(x, v) = \text{diff}(x, \{v\})$. We now prove the following crucial structural result on allocations in the leastcore.

Lemma 2.2.8 *For every leastcore allocation x, i.e. for every $x \in P_1(\varepsilon_1)$, we have that*

(i) for all $i \in [k]$, for all $u \in S_i^*$: $\text{diff}(x, u) = \text{diff}(x, v_i^*)$,

(ii) for all $e \in E^+$: $\text{ex}(x, e) \geq 0$.

Proof: Consider $u \in S_i^*$, and note that we may use Lemma 2.2.7 to choose a universal matching M_u covering precisely

$$S_i^* \backslash \{u\} \cup \bigcup_{j\neq i} S_j^* \backslash \{v_j^*\}.$$

Hence we have $V(M_u) \cup \{u\} = V(M^*) \cup \{v_i^*\}$, and since M^* and M_u are universal, $x(M^*) = x^*(M^*)$ and $x(M_u) = x^*(M_u)$. Using these observations we see that

$$\begin{aligned}\text{diff}(x, u) &= x(u) - x(M_u) - (x^*(u) - x^*(M_u))\\ &= x(v_i^*) - x(M^*) - (x^*(v_i^*) - x^*(M^*))\\ &= \text{diff}(x, v_i^*).\end{aligned}$$

showing (i).

Now we prove (ii). Consider $e \in E^+$ where $e = \{u, v\}$. Since $e \notin E(S_i^*)$ for all $i \in [k]$, we can choose a universal matching M exposing u and v by Lemma 2.2.7. Thus $M \cup \{e\}$ is also a matching. Notice M is x-tight, and so we have

$$\text{ex}(x, e) = \underbrace{\text{ex}(x, M \cup \{e\})}_{\geq \varepsilon_1} - \underbrace{\text{ex}(x, M)}_{=\varepsilon_1} \geq 0$$

as desired. ■

2.2.6 Optimal Matchings of Restricted Cardinality

This section will study the way that minimum excess matchings of a particular cardinality can can be found within $S_1^*, \dots, S_k^*$. The observed structure will be essential for arguing correctness of our compact formulation for the leastcore in Section 2.2.7.

It is well-known [73], that for any $t \in \mathbb{N}$ and for any graph H,

$$\operatorname{conv}\{\chi(M) : M \in \mathcal{M}(H), |M| = t\}$$

has the following linear description:

$$\begin{aligned} P_t(H) := \{x \in \mathbb{R}^E :& \\ & x(\delta(v)) \le 1 && \text{for all} \quad v \in V(H) \\ & x(E(U)) \le \frac{|U|-1}{2} && \text{for all} \quad U \in \mathcal{O}(H) \\ & x(E(H)) = t \\ & x \ge 0\}. \end{aligned}$$

For any given $c \in R^{E(H)}$ we denote by $P_t^c(H)$ the set of vertices x of the above polytope maximizing $c^T x$, i.e. the optimal solutions to the linear program $\max\{c^T x : x \in P_t(H)\}$. Since the vertices are integral

$$P_t^c(H) = \{\chi(M) : M \text{ is a maximum } c\text{-weight matching of size } t\}.$$

The dual to this linear optimization problem is to minimize

$$\sum_{v \in V(H)} y_v + \sum_{U \in \mathcal{O}(H)} \frac{|U|-1}{2} z_U + t\gamma$$

where (y, z, γ) is in $D_{t,c}(H)$ defined as follows:

$$\begin{aligned} D_{t,c}(H) := \{(y, z, \gamma) \in& \mathbb{R}^{V(H)} \times \mathbb{R}^{\mathcal{O}(H)} \times \mathbb{R} : \\ & y_u + y_v + \gamma + \sum_{U \in \mathcal{O} : uv \subseteq U} z_U \ge c(uv) && \text{for all} \quad uv \in E(H) \\ & y, z \ge 0\}. \end{aligned}$$

Let $D_t^c(H)$ denote the set of optimal solutions to

$$\min\{\sum_{v \in V(H)} y_v + \sum_{U \in \mathcal{O}(H)} \frac{|U|-1}{2} z_U + t\gamma : (y, z, \gamma) \in D_{t,c}(H)\}.$$

Lemma 2.2.9 *Suppose* $2 \leq t \leq \left\lfloor \frac{|V(H)|}{2} \right\rfloor$. *Let* $x \in P_t^c(H)$ *and* $(y, z, \gamma) \in D_t^c(H)$. *If the support of* z *is laminar and* $y = 0$ *then there exists* $e \in \operatorname{supp}(x)$ *such that*

$$x - \chi(e) \in P_{t-1}^c(H)$$

and there exists $(0, z', \gamma') \in D_{t-1}^c(H)$ *with the support of* z' *laminar.*

Proof: Let $\mathcal{L} = \operatorname{supp}(z)$ be the laminar family defined by the support of z. Let $S_1, \dots, S_\ell \in \mathcal{L}$ be the top level sets of $\mathcal{L}$ (i.e. containment maximal sets), ordered so that

$$0 < z_{S_1} \leq z_{S_2} \leq \cdots \leq z_{S_\ell}.$$

Since $y = 0$, if there exists $e \in \operatorname{supp}(x) \cap (E(H) \setminus \bigcup_{i \in [\ell]} E(S_i))$ then by complementary slackness, $x - \chi(e) \in P_{t-1}^c(H)$ and $(0, z, \gamma) \in D_{t-1}^c(H)$. Thus we may assume that $x(uv) = 0$ for all $uv \in E(H) \setminus \bigcup_{i \in [\ell]} E(S_i)$. Let $uv \in E(S_1)$ such that $x_{uv} = 1$ and S_1 is a minimal set in $\mathcal{L}$ containing uv. Complementary slackness assures us that such uv exists. Let $x' = x - \chi(uv)$. Define $z' \in \mathbb{R}^{\mathcal{O}(H)}$ as follows

$$z'_U = \begin{cases} z_U - z_{S_1}, & \text{if } U = S_i \text{ for some } i \in [\ell] \\ z_U, & \text{otherwise.} \end{cases}$$

Define $\gamma' = \gamma + z_{S_1}$. We will show $x' \in P_{t-1}^c(H)$ and $(0, z', \gamma') \in D_{t-1}^c(H)$. First we verify feasibility. Indeed, since x is a matching with t edges, x' is a matching with $t - 1$ edges and primal feasibility is satisfied. For dual feasibility, observe by our choice of S_1 that $z' \geq 0$, and for any edge $uv \in \bigcup_{i \in [\ell]} E(S_i)$ the net effect on the left hand side of the dual constraint associated with e is 0. For $uv \in E(H) \setminus \bigcup_{i \in [\ell]} E(S_i)$, the left hand side of the dual constraint associated with e increased by z_{S_1}.

Clearly since $\operatorname{supp}(z)$ is laminar, and $\operatorname{supp}(z') \subseteq \operatorname{supp}(z) \setminus \{S_1\}$, we have that $\operatorname{supp}(z')$ is laminar. It remains to verify optimality, which we will show via complementary slackness. Consider some $uv \in E(H)$ for which $x'_{uv} > 0$. Then $uv \in E(S_i)$ for exactly one i. Hence

$$\gamma' + \sum_{U \in \mathcal{O}(G): uv \subseteq U} z'_U = \gamma + z_{S_1} + (-z_{S_1}) + \sum_{U \in \mathcal{O}(G): uv \subseteq U} z_U = c(uv)$$

where the last equality follows from complementary slackness for x and $(0, z, \gamma)$. Lastly, let $U \in \mathcal{L}$ such that $z'_U > 0$. By our choice of z', $U \neq S_1$ and so

$$x'(E(U)) = x(E(U)) = \frac{|U| - 1}{2}$$

where the last equality follows from complementary slackness for x and $(0, z, \gamma)$. ■

Lemma 2.2.10 *Let $x \in P_1(\varepsilon_1)$ and let $M \in \mathcal{M}$ be a matching such that $M \subseteq \bigcup_{i\in[k]} E(S_i^*)$. Then there exists $M' \subseteq M^*$ such that $\mathrm{ex}(x, M') \leq \mathrm{ex}(x, M)$ and for all $i \in [k]$, $|M' \cap E(S_i^*)| = |M \cap E(S_i^*)|$.*

Proof: Let $i \in [k]$ and let $H = G[S_i^*]$. Let $t = |M^* \cap E(H)|$, and let $c \in \mathbb{R}^{E(H)}$ be defined by

$$c(uv) := w(uv) - x(uv) \quad \text{for all } uv \in E(H).$$

Let $(y, z, \gamma) \in D_t^c(H)$. Since $c^T\chi(M) = -\mathrm{ex}(x, M)$ for all $M \in \mathcal{M}(H)$, by Lemma 2.2.6 and complementary slackness, we have $y = 0$. Now we use standard uncrossing techniques (see for ex. [57, Pages 141-150]) to obtain $(0, z, \gamma) \in D_t^c(H)$ with $\mathrm{supp}(z)$ laminar.

Note that $\chi(M^* \cap E(H))$ is in $P_t^c(H)$, otherwise we can replace $M^* \cap E(H)$, within H, with an optimal matching in $P_t^c(H)$ to obtain a matching with lower excess than M^*, contradicting the universal tightness of M^*.

Apply Lemma 2.2.9 inductively to obtain $M_i' \subseteq M^* \cap E(H)$ such that $\chi(M_i') \in P^c_{|M' \cap E(H)|}(H)$. Since all S_i^* are disjoint, the matching $M' = \bigcup_{i\in[k]} M_i'$ is the desired matching. ■

2.2.7 Compact Formulation

Recall that x^* is a fixed universal allocation in $P_1(\varepsilon_1)$. Let $E^* \subseteq E$ denote the union of universal matchings, i.e. $E^* = \cup_{M\in\mathcal{M}_{uni}} M$. Since a matching is universal if any only if it is x^*-tight it is not hard to compute E^*. To test if an edge uv is in E^*, one simply computes a maximum weight matching on the graph obtained by deleting u and v from G, $G - u - v$, with respect to the weights $-\mathrm{ex}(x^*, e)$ for all $e \in E(G - u - v)$. Then $uv \in E^*$ if and only if the value of this maximum weight matching is $-\varepsilon_1 + \mathrm{ex}(x^*, uv)$.

We now define linear program ($\overline{P}_1$).

$$\begin{aligned} \max \quad & \varepsilon & & & (\overline{P}_1)\\ \text{s.t.} \quad & \mathrm{diff}(x, u) = \mathrm{diff}(x, v_i^*) & & \text{for all} \quad u \in S_i^*,\ i \in [k] & (2.2)\\ & \mathrm{ex}(x, e) \leq 0 & & \text{for all} \quad e \in E^* & \\ & \mathrm{ex}(x, e) \geq 0 & & \text{for all} \quad e \in E^+ & \\ & \mathrm{ex}(x, M^*) = \varepsilon & & & \\ & x(V) = \nu(V) & & & \\ & x \geq 0\,. & & & \end{aligned}$$

Let $\overline{\varepsilon}_1$ be the optimal value of the linear program ($\overline{P}_1$). We now show that $\overline{P_1}(\overline{\varepsilon}_1)$ is indeed a compact description of the leastcore $P_1(\varepsilon_1)$.

Theorem 2.2.11 *We have $\varepsilon_1 = \overline{\varepsilon}_1$ and $P_1(\varepsilon_1) = \overline{P}_1(\overline{\varepsilon}_1)$.*

Proof: First, we show that $P_1(\varepsilon_1) \subseteq \overline{P}_1(\varepsilon_1)$. Consider $x \in P_1(\varepsilon_1)$. By Lemma 2.2.8(i) we have

$$\mathrm{diff}(x, u) = \mathrm{diff}(x, v_i^*) \qquad \text{for all} \quad u \in S_i^*,\ i \in [k].$$

Lemma 2.2.8(ii) shows that $\mathrm{ex}(x, e) \geq 0$ for all $e \in E^+$, and $\mathrm{ex}(x, M^*) = \varepsilon_1$ holds by the universality of M^*. It remains to show that

$$\mathrm{ex}(x, e) \leq 0 \quad \text{for all} \quad e \in E^*.$$

Suppose for contradiction there exists $e \in E^*$ such that $\mathrm{ex}(x, e) > 0$. By the definition of E^*, there exists a universal matching M' containing e. Since M' is universal, $\mathrm{ex}(x, M') = \varepsilon_1$. But by our choice of e,

$$\mathrm{ex}(x, M' \setminus \{e\}) < \mathrm{ex}(x, M') = \varepsilon_1$$

contradicting that x is in $P_1(\varepsilon_1)$. Thus we showed that (x, ε_1) is feasible for ($\overline{P}_1$), i.e. we showed that $P_1(\varepsilon_1) \subseteq \overline{P}_1(\varepsilon_1)$.

To complete the proof we show that $\overline{P}_1(\overline{\varepsilon}_1) \subseteq P_1(\overline{\varepsilon}_1)$. Let x be an allocation in $\overline{P}_1(\overline{\varepsilon}_1)$. Due to the description of the linear program (P_1), it is enough to show that for every matching $M \in \mathcal{M}$ we have

$$\mathrm{ex}(x, M) \geq \overline{\varepsilon}_1\,.$$

Since $\mathrm{ex}(x, e) \geq 0$ for all $e \in E^+$, it suffices to consider only the matchings M, which are unions of matchings in the graphs $G[S_i^*]$, $i \in [k]$. Let $t_i := |M \cap E(S_i^*)|$. By Lemma 2.2.10 applied to x^* there exists $M' \subseteq M^*$ such that

$$\mathrm{ex}(x^*, M) \geq \mathrm{ex}(x^*, M')$$

and

$$|M' \cap E(S_i^*)| = t_i, \quad \text{for all } i \in [k].$$

Then due to constraints (2.2) in ($\overline{P}_1$) we have

$$\begin{aligned}
\mathrm{ex}(x, M) &= \sum_{i=1}^{k} 2t_i\, \mathrm{diff}(x, v_i^*) + \mathrm{ex}(x^*, M) \\
&= \mathrm{diff}(x, M) + \mathrm{ex}(x^*, M) \\
&= \mathrm{diff}(x, M') + \mathrm{ex}(x^*, M') \\
&\geq \mathrm{ex}(x, M') \\
&\geq \mathrm{ex}(x, M^*) \\
&= \overline{\varepsilon}_1,
\end{aligned}$$

where the last inequality follows since $M' \subseteq M^*$ and $\mathrm{ex}(x, e) \leq 0$ for all $e \in E^*$.

Thus, we showed that $P_1(\varepsilon_1) \subseteq \overline{P}_1(\varepsilon_1)$ and $\overline{P}_1(\overline{\varepsilon}_1) \subseteq P_1(\overline{\varepsilon}_1)$. Recall, that ε_1 and $\overline{\varepsilon}_1$ are the optimal values of the linear programs (P_1) and ($\overline{P}_1$) respectively. Thus, we have $\varepsilon_1 = \overline{\varepsilon}_1$ and $P_1(\varepsilon_1) = \overline{P}_1(\overline{\varepsilon}_1)$. ■

2.3 Computing the Nucleolus

The last section presented a polynomial-size formulation for the leastcore LP (P_1). In this section we complete our polynomial-time implementation of MPS Scheme by showing that the j-th linear program, (P_j), in the MPS Scheme has the following compact reformulation:

$$\begin{array}{llcll} \max & \varepsilon & & & \\ \text{s.t.} & \text{ex}(x,e) & \geq & \varepsilon - \varepsilon_1 & \text{for all} \quad e \in E^+,\, e \notin \text{Fix}(\overline{P}_{j-1}(\overline{\varepsilon}_{j-1})) \\ & x(v) & \geq & \varepsilon - \varepsilon_1 & \text{for all} \quad v \in V,\, v \notin \text{Fix}(\overline{P}_{j-1}(\overline{\varepsilon}_{j-1})) \\ & \text{ex}(x,e) & \leq & \varepsilon_1 - \varepsilon & \text{for all} \quad e \in E^*,\, e \notin \text{Fix}(\overline{P}_{j-1}(\overline{\varepsilon}_{j-1})) \\ & x & \in & \overline{P}_{j-1}(\overline{\varepsilon}_{j-1}), & \end{array} \qquad (\overline{P}_j)$$

where $\overline{\varepsilon}_j$ is the optimal value of the linear program ($\overline{P}_j$)

Theorem 2.3.1 *For all $j = 1, \ldots, j^*$, we have $\varepsilon_j = \overline{\varepsilon}_j$ and $P_j(\varepsilon_j) = \overline{P}_j(\overline{\varepsilon}_j)$.*

Proof: We proceed by induction. For $j = 1$ the statement holds due to Theorem 2.2.11. Let us show that the statement holds for each $j = 2, \ldots, j^*$, assuming that the statement holds for $j - 1$.

By induction, $P_{j-1}(\varepsilon_{j-1}) = \overline{P}_{j-1}(\overline{\varepsilon}_{j-1})$. We let $\mathcal{F} := \text{Fix}(P_{j-1}(\varepsilon_{j-1})) = \text{Fix}(\overline{P}_{j-1}(\overline{\varepsilon}_{j-1}))$ to ease presentation.

First we show $P_j(\varepsilon_j) \subseteq \overline{P}_j(\varepsilon_j)$. Let $x \in P_j(\varepsilon_j)$. First consider an edge $e \in E^+ \backslash \mathcal{F}$. By Lemma 2.2.7 there exists a universal matching $M \in \mathcal{M}_{uni}$ exposing the endpoints of e. Moreover, since $V(M) \in \text{Fix}(P_1(\varepsilon_1)) \subseteq \mathcal{F}$ we have $V(M \cup \{e\}) \notin \mathcal{F}$. Thus we see that

$$\text{ex}(x,e) = \underbrace{\text{ex}(x, M \cup \{e\})}_{\geq \varepsilon_j} - \underbrace{\text{ex}(x, M)}_{= \varepsilon_1} \geq \varepsilon_j - \varepsilon_1.$$

Now consider $v \in V \backslash \mathcal{F}$. Similar to the previous argument, but via Lemma 2.2.6, we can find a universal matching M exposing v. Thus, since M is universal, $V(M) \cup \{v\} \notin \mathcal{F}$. We also have $\nu(V(M) \cup \{v\}) \geq w(M)$, so

$$x(v) = x(M \cup \{v\}) - w(M) - \text{ex}(x, M) \geq \varepsilon_j - \varepsilon_1.$$

Finally consider $e = \{u, v\} \in E^*\backslash\mathcal{F}$. Since $e \in E^*$ there exists a universal matching $M \in \mathcal{M}_{uni}$ such that e is in M. Since $V(M) \in \mathrm{Fix}(P_1(\varepsilon_1)) \subseteq \mathcal{F}$ and $\{u, v\} \notin \mathcal{F}$, we see that $M \setminus \{e\} \notin \mathcal{F}$. Thus we have

$$\mathrm{ex}(x, e) = \mathrm{ex}(x, M) - \mathrm{ex}(x, M\backslash\{e\}) \leq \varepsilon_1 - \varepsilon_j$$

as desired.

So we have shown $P_j(\varepsilon_j) \subseteq \overline{P}_j(\varepsilon_j)$, and thus it remains to show $\overline{P}_j(\overline{\varepsilon}_j) \subseteq P_j(\overline{\varepsilon}_j)$. Let $x \in \overline{P}_j(\overline{\varepsilon}_j)$ and let $S \subset V$ such that $S \notin \mathcal{F}$. We need to show

$$x(S) - \nu(S) \geq \overline{\varepsilon}_j.$$

Since $S \notin \mathcal{F}$, there exists $v \in S$ such that $\{v\} \notin \mathcal{F}$. Let M be a maximum w-weight matching in $G[S]$, i.e. $w(M) = \nu(S)$. Either $v \in S\backslash V(M)$ or $v \in V(M)$. We proceed by case distinction.

Case 1: $v \in S\backslash V(M)$. Since $x \geq 0$ we have $x(S) \geq x(V(M) \cup \{v\})$, and it suffices to prove that the right-hand side exceeds the weight of M by at least $\overline{\varepsilon}_j$. By assumption, $x \in \overline{P}_j(\overline{\varepsilon}_j) \subseteq \overline{P}_1(\overline{\varepsilon}_1) = P_1(\varepsilon_1)$, and hence

$$\mathrm{ex}(x, M) \geq \varepsilon_1. \tag{2.3}$$

Together with $x(v) \geq \overline{\varepsilon}_j - \varepsilon_1$ this yields the desired inequality.

Case 2: $v \in V(M)$. The same argument as before applies if there is $u \in S\backslash V(M)$ with $\{u\} \notin \mathcal{F}$. Therefore, we focus on the case where $\{u\} \in \mathcal{F}$ for all $u \in S\backslash V(M)$. Then M contains an edge f such that $f \notin \mathcal{F}$. Again using $x \geq 0$, it suffices to show that $x(V(M)) \geq \overline{\varepsilon}_j$. We further distinguish cases: either $f \in M \cap E^+$ or $f \in M\backslash E^+$.

Case 2a: $f \in M \cap E^+$. Here, the desired inequality follows from $\mathrm{ex}(x, M\backslash\{f\}) \geq \varepsilon_1$ due to $x \in P_1(\varepsilon_1)$ and from $\mathrm{ex}(x, f) \geq \overline{\varepsilon}_j - \varepsilon_1$.

Case 2b: $f \in M\backslash E^+$. If $M \cap E^+$ has an edge that is not in $\mathcal{F}$ then we use the same argument as in Case 2a. So we may assume that all of the edges in $M \cap E^+$ are in $\mathcal{F}$. Now recall that $x \in P_1(\varepsilon_1)$, and hence $\mathrm{ex}(x, e) \geq 0$ for all $e \in E^+$ by Lemma 2.2.8.(ii). Thus, we may assume that $V(M) \notin \mathcal{F}$ and $M \cap E^+ = \varnothing$.

By Lemma 2.2.10, applied to x, there exists $M' \subseteq M^*$ such that $|M' \cap E(S_i^*)| = |M \cap E(S_i^*)|$

for all $i \in [k]$ and $\mathrm{ex}(x, M) \geq \mathrm{ex}(x, M')$. Observe that

$$\begin{aligned}
x(M') &= \sum_{i=1}^{k} x(M' \cap E(S_i^*)) && \text{(since } M' \subseteq M^*\text{)} \\
&= \sum_{i=1}^{k} \sum_{e \in M' \cap E(S_i^*)} x(e) + x^*(e) - x^*(e) \\
&= x^*(M') + \sum_{i=1}^{k} |V(M') \cap S_i^*| \cdot \mathrm{diff}(x, v_i^*) && \text{(by Lemma 2.2.8)} \\
&= x^*(M') + \sum_{i=1}^{k} |V(M) \cap S_i^*| \cdot \mathrm{diff}(x, v_i^*) && \text{(since } |M' \cap E(S_i^*)| = |M \cap E(S_i^*)|\text{)} \\
&= x^*(M') + x(M) - x^*(M).
\end{aligned}$$

That is, since $x^*(M') - x^*(M)$ is constant, $x(M')$ is a constant difference from $x(M)$. Hence $V(M) \in \mathcal{F}$ if and only if $V(M') \in \mathcal{F}$. Since $V(M) \notin \mathcal{F}$ we have $V(M') \notin \mathcal{F}$. Thus $V(M^* \backslash M') \notin \mathcal{F}$ and hence there exists $e \in M^* \backslash M'$ such that $e \notin \mathcal{F}$. Observe that e is in E^*. Hence,

$$\begin{aligned}
\mathrm{ex}(x, M) &\geq \mathrm{ex}(x, M') \\
&= \mathrm{ex}(x, M^*) - \mathrm{ex}(x, e) - \mathrm{ex}(x, M^* \backslash (M' \cup \{e\})) \\
&\geq \varepsilon_1 - (\varepsilon_1 - \overline{\varepsilon}_j) \\
&= \overline{\varepsilon}_j,
\end{aligned}$$

where the last inequality follows since $\mathrm{ex}(x, M^* \backslash (M' \cup \{e\})) \leq 0$ and $x \in \overline{P}_j(\overline{\varepsilon}_j)$, $e \in E^* \backslash \mathcal{F}$. ■

Supposing we have a separation oracle for $P_j(\varepsilon_j)$, we can decide, for any $S \subseteq V$, if $S \in \mathrm{Fix}(P_j(\varepsilon_j))$ in polynomial time. To do so, using equivalence of separation and optimization we solve two linear programs: $\max\{x(S) : x \in P_j(\varepsilon_j)\}$ and $\min\{x(S) : x \in P_j(\varepsilon_j)\}$, and test if their optimal values are equal. Note that their optimal values are equal if and only if $S \in \mathrm{Fix}(P_j(\varepsilon_j))$.

With Theorem 2.3.1 we can replace each linear program (P_j) with ($\overline{P}_j$) in the MPS Scheme. We already know how to execute the first round of MPS Scheme in polynomial time via Section 2.2.1. Therefore consider the j-th round of the MPS Scheme, for $j > 1$, and suppose inductively that we can optimize over $\overline{P}_{j-1}(\overline{\varepsilon}_{j-1})$. Since the universal allocation x^*, the node sets S_i^*, $i \in [k]$, the edge set E^+, and the edge set E^* can all be computed in polynomial time, and we can test if a coalition is fixed over $\overline{P}_{j-1}(\overline{\varepsilon}_{j-1})$ in polynomial time, we have shown that the j-th round of the MPS Scheme can be executed in polynomial time for cooperative matching games with empty core. Hence we can compute the nucleolus of such games in polyomial time. Therefore we have shown Theorem 2.0.1.

Chapter 3

Results on b-Matching Games

In Chapter 2 we gave an algorithm for computing the nucleolus of a cooperative matching game with empty core. This resolved the question of the complexity of computing the nucleolus of a matching game. What remains to study is the complexity of computing the nucleolus of b-matching games.

The game-theoretic interpretation of a matching game is a model of a situation where players represented by vertices enter into partnerships with their neighbours to generate value. From this perspective simple b-matching games are well motivated. They model situations where players have the capacity to enter into more than one partnership simultaneously.

In [8] Biró, Kern, and Paulusma showed that separating core of simple b-matching games can be done in polynomial time when $b_v \leq 2$ for all $v \in V$. In Section 3.2 we generalize this result to the leastcore constraints, and hence to the first linear program (P_1) of the MPS Scheme, capturing the case where the core is empty.

On the hardness side when $b = 3 \cdot \chi(V)$ separating the core is NP-hard even when the underlying graph is unweighted and bipartite [8]. In [6] Biró, Kern, Pálvölgyi, and Paulusma showed that computing the nucleolus of simple b-matching games is NP-hard even for unweighted graphs. This result uses a gadget which does not produce a bipartite graph. In Theorem 3.0.1 we generalize this result to unweighted bipartite bounded-degree graphs.

Theorem 3.0.1 *The problem of deciding whether an allocation is equal to the nucleolus of an unweighted bipartite 3-matching game is* NP*-hard, even in graphs of maximum degree 7.*

Since the core is always non-empty for bipartite b-matching games [20], this result implies hardness of computing the nucleolus when the core of a simple b-matching game is non-empty. Section 3.1 covers the proof of Theorem 3.0.1. It combines the approach in [8] for hardness of core separation on bipartite graphs with a careful analysis of multiple rounds of Kopelowitz Scheme inspired by Deng, Fang, and Sun [19].

Our technique uses an intermediate reduction to a variant of cubic subgraph known as *Two from Cubic Subgraph*. Let G be an arbitrary graph. The TWO FROM CUBIC SUBGRAPH problem is to decide if G contains a subgraph H such that there are $u \neq v \in V(H)$ satisfying

$$\deg_H(w) = \begin{cases} 2, & w \in \{u, v\} \\ 3, & \text{else} \end{cases}$$

for all $w \in V(H)$. We say that H is a *Two from Cubic Subgraph.* We say that vertices u and v are *lacking* and we say the edges incident to u or v in G which are not present in H are *lacking edges*.

In Section 3.1.1 we prove Theorem 3.0.2 which states that the Two from Cubic Subgraph problem is NP-complete. Observe that TWO FROM CUBIC SUBGRAPH is in NP as we can efficiently verify a YES instance when the prover gives us a Two from Cubic Subgraph in the input graph.

Theorem 3.0.2 TWO FROM CUBIC SUBGRAPH *is* NP*-complete, even in bipartite graphs of maximum degree 4.*

Notice that The CUBIC SUBGRAPH problem, which is used in the core separation hardness result of [8], was shown to be NP-complete by Plesnik [66]. The hardness of TWO FROM CUBIC SUBGRAPH requires significant technical innovation over the hardness of CUBIC SUBGRAPH. Our gadget is more complex, and its analysis more involved.

The gap in our knowledge of computing the nucleolus of b-matching games now is when $b_v \leq 2$ for all $v \in V$. In Section 3.3, we attempt to chip away at this gap with efficient algorithms to compute the nucleolus in two relevant cases when $b \leq 2$. Section 3.3.1 explores the scenario when only a constant number of vertices satisfy $b_v = 2$ and Section 3.3.2 studies the case when we relax the constraints to allow for non-simple b-matchings.

3.1 Hardness

We consider simple b-matching games for $b \equiv 3$ and uniform weights. The goal of the this section is to prove Theorem 3.0.1.

Hereinafter, $G = (V, E)$ is a bipartite instance of TWO FROM CUBIC SUBGRAPH. We assume that $E \neq \varnothing$ so that $|V| \geq 2$.

Take $G^* := (V^*, E^*)$ to be the following bipartite gadget graph depicted in Section 3.1:

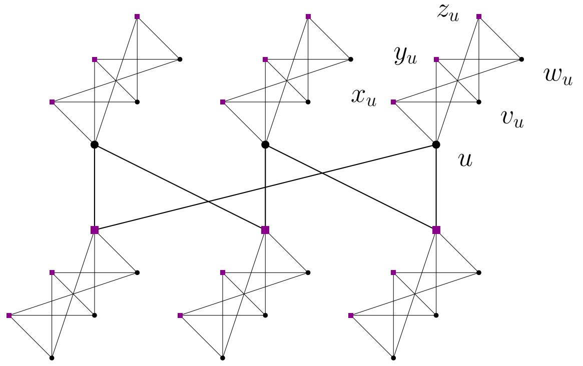

Figure 3.1: The gadget graph from [8].

For each original vertex $u \in V$, create 5 new vertices v_u, w_u, x_u, y_u, z_u. Then, define

$$V^* := V \cup \{v_u, w_u, x_u, y_u, z_u : u \in V\}. \tag{3.1}$$

To obtain E^* from E, we add edges until $(\{u, v_u, w_u\}, \{x_u, y_u, z_u\})$ is a $K_{3,3}$ subgraph for every $u \in V$.

In Figure 3.1, the bigger vertices with bolded edges indicate the original graph and the smaller vertices with thinner edges were added to obtain the gadget graph. The square and circular vertices depict a bipartition of G^*. Observe that the maximum degree of G^* is the maximum degree of G plus 3.

For each $u \in V$, let

$$T_u := \{v_u, w_u, x_u, y_u, z_u\} \tag{3.2}$$
$$V_u := T_u \cup \{u\}. \tag{3.3}$$

We say that T_u are the gadget vertices of u and V_u is the complete gadget of u.

Let $\Gamma = (|V^*|, \nu)$ be the unweighted 3-matching game on G^*.

In Lemma 3.1.2, we show that if no two from cubic subgraphs of G exists, then the nucleolus is precisely $x^* = \frac{3}{2}\chi(V^*)$. Conversely, we prove in Lemma 3.1.3 that the existence of a two from cubic subgraph implies that x^* cannot be the nucleolus. The proof of Theorem 3.0.1 will follow from Lemma 3.1.2, Lemma 3.1.3, and the hardness of TWO FROM CUBIC SUBGRAPH to be proven in Section 3.1.1. Remark that the degree bound follows from the degree bound in Theorem 3.0.2 and the fact that our gadget graph increases the maximum degree of the original graph by 3.

Lemma 3.1.1 *Let M be a maximum 3-matching in G^*. Let $\mathcal{C}$ be the set of connected components of $G^*[M]$. Then for all core allocations x and every component $C \in \mathcal{C}$,*

$$x(C) = \nu(C).$$

$\lvert S \cap \{u, v_u, w_u\} \rvert$	$\lvert S \cap \{x_u, y_u, z_u\} \rvert$	$\nu(S)$	$\mathrm{ex}(x^*, S)$
0	1	0	$\frac{3}{2}$
0	2	0	3
0	3	0	$\frac{9}{2}$
1	0	0	$\frac{3}{2}$
1	1	1	2
1	2	2	$\frac{5}{2}$
1	3	3	3
2	0	0	3
2	1	2	$\frac{5}{2}$
2	2	4	2
2	3	6	$\frac{3}{2}$
3	0	0	$\frac{9}{2}$
3	1	3	3
3	2	6	$\frac{3}{2}$

Table 3.1: The excess computation for Lemma 3.1.2 when $x^* = \frac{3}{2}\chi(V^*), S \subsetneq V_u$.

Proof: Observe that

$$x(V^*) \geq x(M) = \sum_{C \in \mathcal{C}} x(C) \geq \sum_{C \in \mathcal{C}} \nu(C) = \sum_{C \in \mathcal{C}} |M \cap E(C)| = |M| = \nu(V^*)$$

with the first inequality following since $x \geq 0$ and the second inequality following since x is in the core. ■

Lemma 3.1.2 *If G does not contain a two from cubic subgraph, the uniform allocation $x^* = \frac{3}{2}\chi(V^*)$ is the nucleolus of Γ.*

Proof: We argue using the Kopelowitz Scheme. Put (K_k) as the k-th LP in the Kopelowitz Scheme.

We check through computation that

$$\mathrm{ex}(x^*, S) \geq \frac{3}{2} \qquad \forall S \subsetneq V_u, \forall u \in V. \tag{3.4}$$

See Table 3.1.

Let ε_1^K be the optimal objective value to (K_1). We claim that $\varepsilon_1^K = 0$. By core non-emptiness, we have $\varepsilon_1^K \geq 0$. Moreover, using Lemma 3.1.1, since $E \neq \varnothing$, we can choose $u \in V$ so that $V_u \subsetneq V$ is a coalition for which

$$\mathrm{ex}(x, V_u) = 0$$

for all core allocations x. Thus $\varepsilon_1^K = 0$ and the set of optimal solutions to (K_1) is precisely the core.

Let $\mathcal{S}_1 = \mathrm{Tight}(K_1(\varepsilon_1^K))$. We now claim that

$$\mathcal{S}_1 = \{S \subseteq N^* : S = \bigcup_{u \in S \cap N} V_u\}. \tag{3.5}$$

These are the unions of complete gadgets.

Let $(x, 0)$ be an optimal solution to (K_1), i.e. let $x \in \mathcal{C}(\Gamma)$ be a core allocation. Clearly, if S is a union of complete gadgets, then $\mathrm{ex}(x, S) = 0$ due to Lemma 3.1.1. This shows the reverse inclusion in Equation 3.5

We now want to show that if S is not a union of complete gadgets then S is not in $\mathcal{S}_1$. Notice that for any coalition $S \subseteq V^*$, $\nu(S) \leq \frac{3}{2}|S| = x^*(S)$ by the definition of a 3-matching, so $(x^*, 0)$ is an optimal solution to (K_1).

We claim that

$$\forall S \notin \mathcal{S}_1, \mathrm{ex}(x^*, S) \geq \frac{3}{2}. \tag{3.6}$$

This shows that if S is not a union of complete gadgets, then there is some optimal solution of (K_1) for which S does not attain equality in (K_1) and hence $S \notin \mathcal{S}_1$. Thus the inclusion in Equation 3.5 would hold.

Inequality 3.6 is true if $S = \{u\}$ for some $u \in V^*$. If $S \subseteq V$ with $|S| \geq 2$, then $\nu(S) \leq \frac{3}{2}|S| - 2$. Otherwise, the edges of a maximum 3-matching in $G[S]$ induce a two from cubic subgraph. Thus

$$\mathrm{ex}(x^*, S) \geq \frac{3}{2}|S| - \left(\frac{3}{2}|S| - 2\right) \geq 2.$$

It remains to consider the case when there is some $u \in V$ such that $T_u \cap S \neq \varnothing$. The argument here is similar to the reduction employed in [8]: If $S \cap V_u = V_u$, then

$$\mathrm{ex}(x^*, S \setminus V_u) \leq \mathrm{ex}(x^*, S) - 9 + 9 = \mathrm{ex}(x^*, S).$$

We can remove as many of these complete gadgets from S as possible to obtain some coalition S'.

If $S' = \varnothing$, then $S \in \mathcal{S}_1$ by definition. In addition, if $S' \subseteq V$, there is again nothing to prove. Thus there must be some $u' \in S'$ such that $|T_{u'} \cap S'| \geq 1$ and $S' \cap V_{u'} \neq V_{u'}$.

If $|S' \cap T_{u'}| \leq 4$, then

$$\mathrm{ex}(x^*, S' \setminus T_{u'}) \leq \mathrm{ex}(x^*, S') - \frac{3}{2}|T_{u'}| + |E^*(S' \cap T_{u'} \cup \{u'\})| \leq \mathrm{ex}(x^*, S').$$

Finally, if $|S' \cap T_{u'}| = 5$, we are required to have $u' \notin S'$. So

$$\mathrm{ex}(x^*, S' \setminus T_{u'}) \leq \mathrm{ex}(x^*, S') - 5 \cdot \frac{3}{2} + 6 \leq \mathrm{ex}(x^*, S').$$

We may thus again repeatedly remove vertices of $V^* \setminus V$ until we arrive back at the base case of $S' \subseteq V$.

Thus Inequality 3.6 holds as all other coalitions have strictly greater excess with respect to x^*.

To complete the proof, we now argue that $\varepsilon_2^K = \frac{3}{2}$. Observe $\nu(V^*) = \frac{3}{2}|V^*|$ implies that

$$\min_{a \in V^*} x(a) \leq \frac{3}{2}$$

for any allocation x in the core and thus also for feasible solutions to (K_2) as well as the nucleolus. It follows that $\varepsilon_2 \leq \frac{3}{2}$. But Inequality 3.6 shows that this upper bound is attained by x^*.

For all feasible solutions x to (K_2), $x(a) \geq \frac{3}{2}$ for all $a \in V^*$. But $x(a) \leq \frac{3}{2}$ for all $a \in V^*$ as well, or else $x(V^*) > \frac{3}{2}|V^*|$ and x would not be an allocation. Since all the singleton coalitions attain equality in (K_2), we have that Kopelowitz Scheme terminates after two rounds and $x^* \equiv \frac{3}{2}$ is the nucleolus. ■

Lemma 3.1.3 *If G contains a two from cubic subgraph, then the nucleolus of the gadget graph is not $x^* = \frac{3}{2}\chi(V^*)$.*

Proof: We will show that $x^* = \frac{3}{2}\chi(V^*)$ is not an optimal solution to (K_2). Recall that the nucleolus is necessarily an optimal solution to each linear program in the Kopelowitz Scheme. Hence this would show that x^* not the nucleolus.

Let us introduce a parameter as follows:

$$\Delta := \begin{cases} 0, & \text{if } G \text{ contains a cubic subgraph} \\ 1, & \text{if } G \text{ contains a two from cubic subgraph but no cubic subgraph.} \end{cases}$$

$\lvert S \cap \{u\} \rvert$	$\lvert S \cap \{v_u, w_u\} \rvert$	$\lvert S \cap \{x_u, y_u, z_u\} \rvert$	$\nu(S)$	$\mathrm{ex}(x_\delta, S)$
0	0	1	0	$\frac{3}{2} - \frac{\delta}{5}$
0	0	2	0	$3 - \frac{2\delta}{5}$
0	0	3	0	$\frac{9}{2} - \frac{3\delta}{5}$
0	1	0	0	$\frac{3}{2} - \frac{\delta}{5}$
0	1	1	1	$2 - \frac{2\delta}{5}$
0	1	2	2	$\frac{5}{2} - \frac{3\delta}{5}$
0	1	3	3	$3 - \frac{4\delta}{5}$
0	2	0	0	$3 - \frac{2\delta}{5}$
0	2	1	2	$\frac{5}{2} - \frac{2\delta}{5}$
0	2	2	4	$2 - \frac{4\delta}{5}$
0	2	3	6	$\frac{3}{2} - \delta$
1	0	0	0	$\frac{3}{2} + \delta$
1	0	1	1	$2 + \frac{4\delta}{5}$
1	0	2	2	$\frac{5}{2} + \frac{3\delta}{5}$
1	0	3	3	$3 + \frac{2\delta}{5}$
1	1	0	0	$3 + \frac{4\delta}{5}$
1	1	1	2	$\frac{5}{2} + \frac{3\delta}{5}$
1	1	2	4	$2 + \frac{2\delta}{5}$
1	1	3	6	$\frac{3}{2} + \frac{\delta}{5}$
1	2	0	0	$\frac{9}{2} + \frac{3\delta}{5}$
1	2	1	3	$3 + \frac{2\delta}{5}$
1	2	2	6	$\frac{3}{2} + \frac{\delta}{5}$

Table 3.2: The excess computation for x_δ from Lemma 3.1.3 when $S \subsetneq V_u$.

Let $V' \subseteq V$ be the vertices in a cubic subgraph or the vertices of a two from cubic subgraph if no cubic subgraph exists. Then

$$\mathrm{ex}\left(x^*, V'\right) = \frac{3}{2}|V'| - \left(\frac{3}{2}|V'| - \Delta\right) = \Delta. \tag{3.7}$$

In particular, the minimum excess over all coalitions in $\mathcal{S}$ is at most Δ.

For $0 < \delta < \frac{1}{2}$, define

$$x_\delta(a) := \begin{cases} \frac{3}{2} + \delta, & \text{if } a \in V \\ \frac{3}{2} - \frac{\delta}{5}, & \text{if } a \in V^* \setminus V \end{cases}$$

We check by computation in Table 3.2 that

$$\forall u \in V, \forall S \subsetneq V_u, \mathrm{ex}(x_\delta, S) \geq \frac{3}{2} - \delta \tag{3.8}$$

The coalitions with minimum excess among such coalitions are $S = T_u$ for some $u \in V$.

As in Lemma 3.1.2, let $\mathcal{S}_1 = \mathrm{Tight}(K_1(\varepsilon_1^K))$. We claim that $\varepsilon_1^K = 0$ and

$$\mathcal{S}_1 = \{S \subseteq V^* : S = \bigcup_{u \in S \cap N} V_u\}.$$

The fact that $\varepsilon_1 = 0$ is clear from the proof of Lemma 3.1.2. Moreover, it is clear that the unions of complete gadgets must attain equality in (K_1). We need only show that $\mathrm{ex}(x_\delta, S) > 0$ if S is not a union of complete gadgets. This would show that if S is not a union of complete gadgets, then there is some core allocation (in particular x_δ) for which S does not attain equality in (K_1).

If $S = \{a\}$ for some $a \in V^*$, then

$$\mathrm{ex}(x_\delta, S) \geq \frac{3}{2} - \frac{\delta}{5} > 0.$$

When $S \subseteq V$. We have

$$\begin{aligned} \mathrm{ex}(x_\delta, S) &\geq \left(\frac{3}{2} + \delta\right)|S| - \left(\frac{3}{2}|S| - \Delta\right) \\ &= \delta|S| + \Delta \\ &> 0. \end{aligned}$$

Suppose now that there is some $u \in V$ such that $S \cap T_u \neq \varnothing$. Once again, if $S \cap V_u = V_u$,

$$\mathrm{ex}(x_\delta, S \setminus V_u) \leq \mathrm{ex}(x_\delta, S) - 9 + 9 = \mathrm{ex}(x_\delta, S).$$

We can thus remove all complete gadgets from S to obtain another coalition S'. If $S' = \varnothing$, then $S \in \mathcal{S}_1$. Similar to before, if $S' \subseteq V$, we are back at the base case.

Pick some $u' \in S'$ such that $|S' \cap T_{u'}| \geq 1$. Observe that

$$\nu(S') \leq \nu(S' \setminus T_{u'}) + \nu(S' \cap V_{u'}).$$

This is because any maximum 3-matching on S' is a disjoint union of 3-matchings on $S' \setminus T_{u'}$ and $S' \cap V_{u'}$.

Suppose $u' \in S'$. We must have $|S' \cap T_{u'}| \leq 4$.

$$\begin{aligned}
\operatorname{ex}(x_\delta, S') &= x(S' \setminus T_{u'}) + x(S' \cap V_{u'}) - x(u') - \nu(S') \\
&\geq x(S' \setminus T_{u'}) - \nu(S' \setminus T_{u'}) + \left[x(S' \cap V_{u'}) - x(u')\right] - \nu(S' \cap V_{u'}) \\
&\geq \operatorname{ex}(x_\delta, S' \setminus T_{u'}) + |S \cap T_{u'}| \left(\frac{3}{2} - \frac{\delta}{5}\right) - |E^*(S' \cap V_{u'})| \\
&\geq \operatorname{ex}(x_\delta, S' \setminus T_{u'}) - \frac{4}{5}\delta.
\end{aligned}$$

Suppose now that $u' \notin S'$. In this case,

$$\begin{aligned}
\operatorname{ex}(x_\delta, S') &= x(S' \setminus T_{u'}) + x(S' \cap T_{u'}) - \nu(S') \\
&\geq x(S' \setminus T_{u'}) - \nu(S' \setminus T_{u'}) + x(S' \cap T_{u'}) - \nu(S' \cap T_{u'}) \\
&= \operatorname{ex}(x_\delta, S' \setminus T_{u'}) + \operatorname{ex}(x_\delta, S' \cap T_{u'}) \\
&\geq \operatorname{ex}(x_\delta, S' \setminus T_{u'}) + \left(\frac{3}{2} - \delta\right) && \text{by Inequality 3.8}
\end{aligned}$$

By repeatedly removing vertices of $V^* \setminus V$, we see that

$$\begin{aligned}
\operatorname{ex}(x_\delta, S') &\geq \operatorname{ex}(x_\delta, S' \cap V) + \sum_{u \in S' \cap V : S' \cap T_u \neq \varnothing} \left[-\frac{4}{5}\delta\right] + \sum_{u \in V \setminus S' : S' \cap T_u \neq \varnothing} \left[\frac{3}{2} - \delta\right] \\
&\geq \delta |S' \cap V| + \Delta - |S' \cap V| \left(\frac{4}{5}\delta\right) + |\{u \in V \setminus S' : S' \cap T_u \neq \varnothing\}| \left(\frac{3}{2} - \delta\right) \\
&= \frac{\delta}{5}|S' \cap V| + \Delta + |\{u \in V \setminus S' : S' \cap T_u \neq \varnothing\}| \left(\frac{3}{2} - \delta\right) \\
&\geq \frac{\delta}{5} + \Delta.
\end{aligned}$$

The last inequality follows from the assumption that $S' \neq \varnothing$. In particular, at least one of $S' \cap V$ or $\{u \in V \setminus S' : S' \cap T_u \neq \varnothing\}$ is non-empty.

This shows that $\varepsilon_1^K = 0$ is indeed the optimal solution to (K_1). Moreover, $\mathcal{S}_1$ is again the union of complete gadgets.

As an immediate corollary to the proof,

$$\varepsilon_2^K \geq \frac{\delta}{5} + \Delta > \Delta.$$

Recall there was a coalition $V' \subseteq V$ satisfying Inequality 3.7. It follows that $x^* \equiv \frac{3}{2}$ is not an optimal solution to (K_2) and therefore cannot be the nucleolus. ■

3.1.1 Two from Cubic Subgraph

In this subsection, we prove that Two from Cubic Subgraph, from which we reduce to nucleolus testing is NP-hard.

Let $X = \{a_1, a_2, \ldots, a_{3k}\}$ be a ground set and let $\mathcal{S}$ be a collection of 3-element subsets of X of the form $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$. In the Exact Cover by 3-Sets problem, the input is $(X, \mathcal{S})$ and the decision is to decide if there is a subcollection $Y \subseteq \mathcal{S}$ where element $a_i \in X$ is included in exactly one subset $S_j \in Y$. We call such a Y an *exact cover*.

It is well known that Exact Cover by 3-Sets is NP-hard, even if every element of the ground set belongs to exactly three subsets [37]. We reduce Exact Cover by 3-Sets to Two from Cubic Subgraph. We construct a gadget graph from an instance of Exact Cover by 3-Sets as follows.

Step I

Let $(X, \mathcal{S})$ be an instance of the Exact Cover by 3-Sets where every element of the ground set belongs to exactly three subsets. Create the bipartite graph G_0 with bipartition $X \dot\cup \mathcal{S}$, where

$$a_i S_j \in E(G_0) \iff a_i \in S_j.$$

The problem is reformulated as follows: Does there exist a subgraph with vertex set $X \dot\cup \mathcal{S}'$ such that every vertex of X has degree 1 and each vertex of $\mathcal{S}'$ has degree 3? Notice we require the entire ground set to be included in the subgraph vertex set.

Step II

Add $7k$ new vertices to G_0

$$b_1, b_2, \ldots, b_{7k}$$

as follows. Each $b_i, 1 \leq i \leq 3k$ is adjacent to a_i and b_{3k+i}. If $i > 1$, then b_i is also adjacent to b_{3k+i-1}. Otherwise, $i = 1$ and b_i is also adjacent to b_{6k}. Finally, each $b_{6k+j}, 1 \leq j \leq k$, is adjacent to $b_{3k+3j-2}, b_{3k+3j-1}, b_{3k+3j}$.

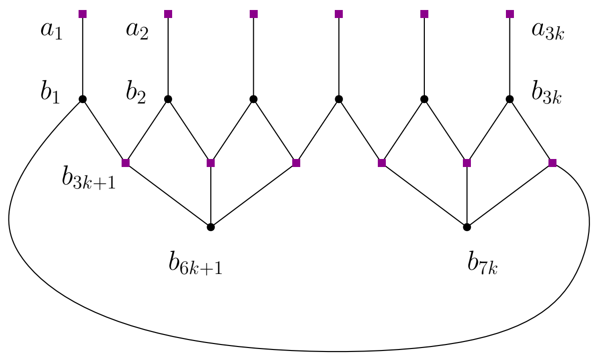

Figure 3.2: A subgraph of G_1 for $k = 2$, depicting the changes in Step I.

See Figure 3.2. The square and circular vertices depict a bipartition of G_1.

Define the following tripartition of the newly added vertices

$$B_1 := \{b_i : 1 \le i \le 3k\} \tag{3.9}$$
$$B_2 := \{b_{3k+i} : 1 \le i \le 3k\} \tag{3.10}$$
$$B_3 := \{b_{6k+i} : 1 \le i \le k\} \tag{3.11}$$

Let us refer to this graph as G_1.

Step III

At this step, we diverge from the work in [66]. In Plesnik's hardness proof of Cubic Subgraph, the author substituted a complete bipartite graph at each a_i so the resulting graph has a cubic subgraph if and only if it has the desired subgraph in Step I. We proceed by using a grid-like substitution at each a_i into two copies of G_1.

Let G_2 be the graph obtained after the following substitution: At each vertex a_i, we substitute the following gadget.

Let $\{S_{i,j} \in \mathcal{S} : j = 1, 2, 3\}$ be the set of 3-element subsets containing a_i. Add vertices

$$u_{i,j}, w_{i,j}, c_{u_{i,j}}, c_{w_{i,j}} \quad \text{for } j = 1, 2, 3.$$

For $j = 1, 2, 3$, create the edges

$$w_{i,j-1}u_{i,j}, u_{i,j}w_{i,j}, w_{i,j}S_{i,j},$$
$$u_{i,j}c_{u_{i,j}}, w_{i,j}c_{w_{i,j}}, c_{u_{i,j}}c_{w_{i,j}}.$$

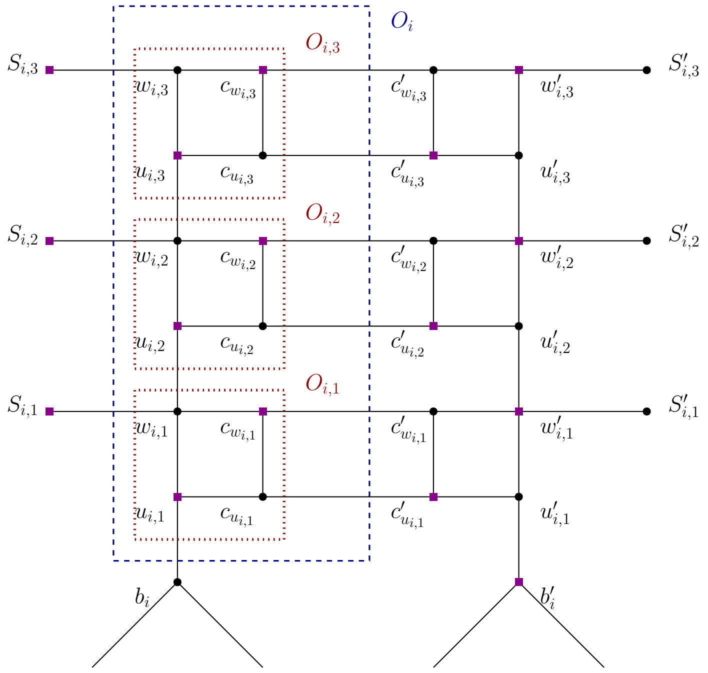

Figure 3.3: The substitution at what used to be a_i, a'_i.

Here we understand $w_{i,0} = b_i$.

Next, create a copy G'_2 of G_2. For each $v \in G_2$, we let v' denote the corresponding copy in G'_2. Also, add the edges $c_{w_{i,j}}c'_{w_{i,j}}, c_{u_{i,j}}c'_{u_{i,j}}$ for $1 \le i \le 3k, 1 \le j \le 3$.

See Figure 3.3. The square and circular vertices depict a bipartition of G.

Define

$$O_{i,j} := \{u_{i,j}, w_{i,j}, c_{u_{i,j}}, c_{w_{i,j}}\}. \tag{3.12}$$

Let us call the vertices

$$O_i := \bigcup_{j=1}^{3} O_{i,j} \tag{3.13}$$

the *ore* of $b_i \in B_1$ and similarly for $b'_i \in B'_1$.

Let this new graph be G. In summary we have the vertex sets $B := B_3 \cup B_2 \cup B_1, S$ and $O_i, 1 \le i \le 3k$ coming from the substitution in G_1. Similarly $B', S', O'_i, 1 \le i \le 3k$ are the vertex sets from G'_2. Let $A = B \cup S \cup \bigcup_{1 \le i \le 2k} O_i$ and $A' = B' \cup S' \cup \bigcup_{1 \le i \le 3k} O'_i$ be the vertices which originated from G_2, G'_2 respectively, so $V(G) = A \dot\cup A'$.

We are now ready to show Theorem 3.0.2 by proving that G has a Two from Cubic Subgraph if and only if $(X, \mathcal{S})$ has an exact cover. Since the construction of G from $(X, \mathcal{S})$ can be done in polynomial time, this will suffice to show the theorem.

First we will show in Lemma 3.1.4 the sufficient direction of the if and only if statement.

Lemma 3.1.4 *If there is an exact cover of X by 3-sets then G contains a two from cubic subgraph.*

Proof: Let $Y \subseteq \mathcal{S}$ be the exact cover of X.

We will describe the vertices from A of a cubic subgraph and take the exact copy within A'.

First, take all vertices $b \in B$.

Fix $b_i \in B_1$. There is exactly one $S_{i,j} \in Y$ such that $a_i \in S_{i,j}$. Let P_i be the unique $b_i S_{i,j}$-path in the ore of b_i. Put

$$Q_i := \bigcup_{j : O_{i,j} \cap V(P_i) \neq \varnothing} O_{i,j} \tag{3.14}$$

Then we take

$$B \cup \left(\bigcup_i Q_i \right) \cup Y \tag{3.15}$$

as well as their copies in A' to be the vertex set of our cubic subgraph. ■

To prove the necessary direction of the if and only if statement we need to study the structure of the gadget graph G. The following easy proposition follows from inspection of O_i. It allows us to extrapolate from certain vertices being in a Two from Cubic Subgraph that other vertices must be in that subgraph as well.

Proposition 3.1.5 *Let $\bar{G}$ be a graph. Let H be a two from cubic subgraph of $\bar{G}$ and let $v \in V(H)$. Let $h \in [3]$ and let $g \in \{0, 1, \ldots, h\}$.*

Suppose there are edge disjoint paths $P_1, \ldots, P_h$ in $\bar{G}$ all starting at v and each ending at w_i for $i = 1, \ldots, h$ respectively. Further suppose that for all $i \in [h]$, each vertex in $V(P_i) \backslash \{w_i\}$ is of degree at most 3 in $\bar{G}$.

If H has at most g lacking vertices in $\bigcup_{i=1}^{h} P_i$ then there exist $h - g$ unique paths P_i, $i \in [h]$ such that every vertex of P_i is in H.

Proof:

Let L be the set of lacking vertices in H which lie on a path in $P_1, \ldots, P_h$. For each vertex $u \in L$ there exists at most one edge in $\delta_{\bar{G}}(u) \backslash E(H)$ since u is of degree at most 3 in $\bar{G}$. Let E_L be the set of such edges. Hence $|E_L| \leq g$.

Now observe that for every $i \in [h]$, if $E(P_i) \cap E_L = \emptyset$ then $P_i \subseteq H$. This follows inductively from the observation that since v is of degree at most 3 in $\bar{G}$ the edge from v to its neighbour on P_i must be present in H. We repeat this observation at every subsequent vertex of P_i until we see that $P_i \subseteq H$.

We complete the proof by the pigeonhole principle: at least $h - g$ paths P_i must satisfy $E(P_i) \cap E_L = \emptyset$. ■

We use Proposition 3.1.5 to draw two conclusions. The first is that if there are vertices in a Two from Cubic Subgraph contained in B or S then there is an ore with vertices in the Two from Cubic Subgraph and there is no lacking vertex in that ore. This is the content of Lemma 3.1.6. Notice that by symmetry this lemma could also apply to B' or S'. The second conclusion we draw using Propagation 3.1.5 is that if there is an ore with vertices in a Two from Cubic Subgraph then one of B, S, B', or S' has vertices in the Two from Cubic Subgraph.

Lemma 3.1.6 *If $B \cap V(H) \neq \varnothing$ or $S \cap V(H) \neq \varnothing$ then there exists $i \in [3k]$ such that $(O_i \cup O_i') \cap V(H) \neq \varnothing$ and there is no lacking vertex of H in $O_i \cup O_i'$.*

Proof: The case $S \cap V(H) \neq \varnothing$ is a simple application of the pigeonhole principle, so it remains to consider the case where $B \cap V(H) \neq \varnothing$.

Suppose $B \cap V(H) \neq \varnothing$. Let $b_i \in B \cap V(H)$. Without loss of generality we have three cases: $i = 1, 3k+1, 6k+1$. In each case we will construct three edge disjoint paths from b_i ending in vertices amongst $u_{i,1}, \dots, u_{3k,1}$. Then using Proposition 3.1.5 with $G = \bar{G}$, and the pigeonhole principle on where lacking vertices can lie, we obtain the desired result.

Consider the first case, $b_i = b_1$. We construct 3 edge disjoint paths starting from b_1. The first path is $b_1, u_{1,1}$. The second path is $b_1, b_{3k+1}, b_2, u_{2,1}$. The third path is $b_1, b_{6k}, b_{3k}, u_{3k,1}$.

Consider the second case, $b_i = b_{3k+1}$. We construct 3 edge disjoint paths starting from b_{3k+1}. The first path is $b_{3k+1}, b_1, u_{1,1}$. The second path is $b_{3k+1}, b_2, u_{2,1}$. The third path is $b_{3k+1}, b_{6k+1}, b_{3k+2}, b_3, u_{3,1}$.

Consider the third case, $b_i = b_{6k+1}$. We construct 3 edge disjoint paths starting from b_{6k+1}. The first path is $b_{6k+1}, b_{3k+1}, b_1, u_{1,1}$. The second path is $b_{6k+1}, b_{3k+2}, b_2, u_{2,1}$. The third path is $b_{6k+1}, b_{3k+3}, b_3, u_{3,1}$. ■

Lemma 3.1.7 *If there exists $i \in [3k]$ such that $(O_i \cup O_i') \cap V(H) \neq \varnothing$ then*

$$V(H) \cap \left(S \cup S' \cup B \cup B'\right) \neq \varnothing.$$

Proof: Let $i \in [3k]$ such that $(O_i \cup O_i')) \cap V(H) \neq \varnothing$. Let $v \in (O_i \cup O_i')) \cap V(H) \neq \varnothing\}$. Let $F = \{S_{i,1}w_{i,1}, S_{i,2}w_{i,2}, S_{i,1}'w_{i,1}', S_{i,2}'w_{i,2}'\}$. If

$$F \cap E(H) \neq \varnothing$$

then we are done. So suppose that $F \cap E(H) = \varnothing$.

Let $\bar{G} = G - F$ be the graph obtained from G by deleting edges in F. Now H is a two from cubic subgraph of $\bar{G}$. By inspection we can construct 3 edge disjoint paths in G' starting from v and ending in vertices in $\{S_{i,3}, S'_{i,3}, b_i, b'_i\}$, satisfying the hypothesis of Proposition 3.1.5. Therefore by Proposition 3.1.5 the result holds. ■

Lemma 3.1.6 and Lemma 3.1.7 when taken together imply that a Two from Cubic Subgraph of G always has vertices of some ore, and in particular the vertices of that ore are not lacking. By inspecting an ore we obtain the easy Proposition 3.1.8 which says that vertices of an ore in a Two from Cubic Subgraph are forced to choose certain edges if their is no lacking vertex in that ore. We can use Proposition 3.1.8 with our conclusion from combining Lemma 3.1.6 and Lemma 3.1.7 to see that any Two from Cubic Subgraph must simultaneously have vertices in the ores, B, B', S, and S'. This is the content of Lemma 3.1.9.

Proposition 3.1.8 *Let H be a two from cubic subgraph of G. Let O_i be the ore of $b_i \in B$ for some $i \in [3k]$. If $V(H) \cap V(O_i) \neq \varnothing$ and no lacking vertex of H is in O_i then H contains exactly one edge from*

$$\{S_{i,1}w_{i,1}, S_{i,2}w_{i,2}S_{i,3}w_{i,3}\},$$

and the edge $u_{i,1}b_i$.

Lemma 3.1.9 *Let H be a non-empty two from cubic subgraph of G. For each*

$$K \in \{B, S, \bigcup_{i=1}^{3k} O_i, B', S', \bigcup_{i=1}^{3k} O'_i\}$$

we have that

$$V(H) \cap K \neq \varnothing.$$

Proof: Since H is non-empty we may assume without loss of generality that $V(H) \cap A \neq \varnothing$. Therefore by Lemma 3.1.7 and Lemma 3.1.6 there exists $i \in [3k]$ such that $(O_i \cup O'_i) \cap V(H) \neq \varnothing$ and $O_i \cup O'_i$ contains no lacking vertex of H.

Thus by Proposition 3.1.8 applied to each of O_i and O'_i we obtain the desired result. ■

Using Lemma 3.1.9 we can restrict our attention to proving that G having a Two from Cubic Subgraph implies an exact cover in the setting where all the sets mentioned in the lemma intersect with the Two from Cubic Subgraph. This is the content of Lemma 3.1.12. To prove this lemma we will need Lemma 3.1.10 and Corollary 3.1.11 which follow.

Lemma 3.1.10 *If $V(H) \cap B \neq \varnothing$ and there is at most one lacking vertex in $V(H) \cap B$ then $B \subseteq V(H)$.*

Proof: Since $B \cap V(H) \neq \varnothing$ let $v \in B \cap V(H)$. Notice that B is connected and every vertex of B lies on a cycle. Hence B is 2-connected and thus 2-edge-connected, i.e. there are two edge disjoint paths between v and every other vertex in B using only edges in B. Thus by Proposition 3.1.5 the result holds. ∎

When every vertex of B is in a Two from Cubic Subgraph H then the only way an ore O_i cannot have a vertex in H is if b_i is lacking and specifically lacks edge $u_{i,1}b_i$. This observation yields the following Corollary to Lemma 3.1.10.

Corollary 3.1.11 *If $V(H) \cap B \neq \varnothing$ and there is at most one lacking vertex in $V(H) \cap B$ then there is at most one $i \in [3k]$ such that $O_i \cap V(H) = \varnothing$.*

Furthermore if $V(H) \cap B \neq \varnothing$ and there is no lacking vertex in $V(H) \cap B$ B has no lack then $O_i \cap V(H) \neq \varnothing$ for all $i \in [3k]$.

Lemma 3.1.12 *Let H be a two from cubic subgraph of G. If*

$$V(H) \cap K \neq \varnothing \quad \text{for each } K \in \{B, S, \bigcup_{i=1}^{3k} O_i, B', S', \bigcup_{i=1}^{3k} O'_i\}$$

then there is an exact cover of X by 3-sets.

Proof: Without loss of generality we have two cases: either every lacking vertex of H is in A' or exactly one lacking is in A and one lacking vertex is in A'.

Consider the first case, that every lacking vertex is in A'. Since $V(H) \cap B \neq \varnothing$ and there is no lacking vertex in A, by Lemma 3.1.10, $B \subseteq V(H)$ and further every ore O_i for $i \in [3k]$ has a non-trivial intersection with $V(H)$.

Now by Proposition 3.1.8 for every $i \in [3k]$ there is a $j(i) \in [3]$ such that $S_{i,j(i)}w_{i,j(i)} \in E(H)$. Choose $Y = \cup_{i \in [3k]} S_{i,j(i)}$. By construction there is exactly one set in Y containing a_i for each $i \in [3k]$ and moreover there is at most one such set. Therefore Y is an exact cover of X by 3-sets as desired.

Now we need to consider the second case, that exactly one lacking vertex of H is in each of A and A'. Let $\ell \in V(H) \cap A$ be the lacking vertex of H which is in A. For this case we need to consider three distinct subcases: $\ell \in S$, $\ell \in B$, and $\ell \in \bigcup_{i \in [3k]} O_i$. Notice the proof from the first case works without modification in the first subcase, and in fact provides a contradiction showing this subcase is impossible. It remains to study the latter two subcases.

First consider the subcase where $\ell \in B$. By Corollary 3.1.11 there is at most one $i_0 \in [3k]$ such that O_{i_0} is disjoint from $V(H)$. Thus we can obtain a unique set $S_{i,j(i)}$ covering each a_i for $i \in [3k] \backslash \{i_0\}$ by the construction from the proof of the first case. Let Y be the union of those covering sets as in the first case. If Y is not an exact cover then Y covers every element of the ground set except for a_{i_0}. Then

$$3k - 1 = |X| - 1 = 3|Y|,$$

a contradiction since k and $|Y|$ are integers.

Finally consider the subcase where $\ell \in O_{i_0}$ for some $i_0 \in [3k]$. Since $V(H) \cap B \neq \varnothing$, and there is no lacking vertex in $V(H) \cap B$, by Corollary 3.1.11, $V(H) \cap O_i \neq \varnothing$ for each $i \in [3k]$. Since every ore in A except O_{i_0} has no lacking vertex, by Proposition 3.1.8, for every $i \in [3k] \backslash \{i_0\}$, there exists a unique $S_{i,j(i)}$ covering a_i by construction from the proof of the first case. Let Y be the union of those covering sets as in the first case. If Y is not an exact cover then Y covers every element of the ground set except for a_{i_0}. Then

$$3k - 1 = |X| - 1 = 3|Y|,$$

a contradiction as in the previous subcase. Therefore Y is an exact cover as desired. ■

Lemma 3.1.9 and Lemma 3.1.12 show that if G has a Two from Cubic Subgraph then $(X, \mathcal{S})$ has an exact cover. Since we showed the converse in Lemma 3.1.4 this means we have shown Theorem 3.0.2.

3.2 Leastcore Separation for Simple b-Matching Games

It has been shown that core separation, and hence leastcore separation, is NP-hard [8] for simple b-matching games when we allow $b_v \geq 3$ for arbitrary nodes $v \in V$, even for unweighted bipartite graphs. In Section 2.2.1 of Chapter 2 we saw that the separation of (P_1) is polynomial time solvable when $b_v = 1$ for all $v \in V$. Thus we restrict our attention to b vectors such that $b_v \leq 2$ for all $v \in V$, and consider simple b-matching games in this setting. Hence we will resolve the complexity of computing (P_1) for simple b-matching games.

Given a b-valued w-weighted graph $G = (V, E)$ where $b_v \leq 2$ for all $v \in V$, let (n, ν) be the associated simple b-matching game. The leastcore linear program (P_1) for (n, ν) can equivalently be stated as

$$\begin{aligned} \max\ & \varepsilon \\ \text{s.t. } & x(M) \geq w(M) + \varepsilon \qquad \text{for all} \quad M \in \mathcal{M}_b \\ & x(V) = \nu(V) \\ & x \geq \mathbb{0}\,, \end{aligned} \tag{P_1^b}$$

where $\mathcal{M}_b$ is the set of simple b-matchings in G.

3.2.1 Max-Weight Min-Cost Matching

To separate the leastcore linear program for a given point $(p, \varepsilon) \in \mathbb{R}^V \times \mathbb{R}$ it suffices to be able to solve

$$\begin{aligned} \max\ & w(M) - p(V(M)) \\ \text{s.t. }\ & M \in \mathcal{M}_b \end{aligned} \tag{P_2^b}$$

and verify that this value is at most $-\varepsilon$.

The challenge with adapting the leastcore separation algorithm for matching games to b-matching games is deciding what the weight of edges adjacent to b-value 2 nodes should be. If we simply take weights $\bar{w}(uv) = w(uv) - p(u) - p(v)$, for all $uv \in E$, as we would for matching games, then the cost $p(u)$ for nodes u with $b_u = 2$ would be double counted in b-matchings which cover u with 2 edges. On the other hand if we only subtracted $\frac{1}{2}p(u)$ from weights of edges incident with b-value 2 node u, the cost $p(u)$ would only be half accounted for in b-matchings which only use one edge incident to u. To put it succinctly, we do not know a priori if the optimal matching will use one edge or two edges incident to any given b-value 2 node and hence do not known what the appropriate reduced weights should be.

To overcome the aforementioned difficulty, we will construct an auxiliary graph instance on which we can control the number of edges incident to b-value 2 nodes and still solve (P_2^b).

Construct graph $\bar{G} = (\bar{V}, \bar{E})$ by starting with graph G. Now, for each $v \in V$ such that $b_v = 2$ add nodes v_1, v_2 to $\bar{G}$ and create a cycle vv_1v_2. Let T be the set of all such cycles. Additionally for each $v \in V$ such that $b_v = 2$, add node v' to $\bar{G}$ and create edge vv'. Now $\bar{G}$ consists of graph G, triangles T, and edges vv' for each $v \in V$ such that $b_v = 2$.

For each $v \in V$ set $\bar{b}_v = b_v$. Set $\bar{b}_{v_1} = \bar{b}_{v_2} = 2$ for each $vv_1v_2 \in T$. Set $\bar{b}_{v'} = 1$ for each $v \in V$ such that $b_v = 2$. For each $e \in \bar{E}$ set

$$\bar{c}(e) = \begin{cases} 2, & \text{if } e = v_1v_2 \text{ for some } vv_1v_2 \in T \\ 1, & \text{otherwise.} \end{cases}$$

Finally for each $uv \in \bar{E}$ set

$$\bar{w}(uv) = \begin{cases} w(uv) - \frac{p(u)}{b_u} - \frac{p(v)}{b_v}, & \text{if } uv \in E \\ -\frac{p(u)}{2} & \text{if } b_v = 2 \text{ and } u = v' \\ 0, & \text{otherwise.} \end{cases}$$

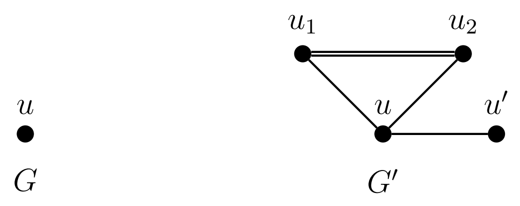

Figure 3.4: The transformation of a b-value 2 vertex in G to the corresponding gadget in $\bar{G}$.

The problem of finding a maximum $\bar{w}$-weight, $\bar{c}$-capacitated, $\bar{b}$-matching in $\bar{G}$ which fully matches each $\bar{b}$-value 2 node can be solved in polynomial time by computing a vertex solution to the following polynomial time solvable linear program [73]:

$$\begin{aligned} \max\ & \bar{w}^T x && && \text{(Q)} \\ \text{s.t. }\ & x(\delta(v)) = \bar{b}_v && \forall v \in \bar{V} : \bar{b}_v = 2 \\ & x(\delta(v)) \leq \bar{b}_v && \forall v \in \bar{V} : \bar{b}_v = 1 \\ x(E(U)) + x(F) & \leq \frac{\bar{b}(U) + \bar{c}(F) - 1}{2} && \forall U \subseteq \bar{V}, F \subseteq \delta(U) : \bar{b}(U) + \bar{c}(F) \text{ is odd} \\ & 0 \leq x \leq \bar{c}. \end{aligned}$$

Note that the vertices of the feasible region of (Q) are all integral. We will show the optimal value of (Q) is equal to the value of an optimal solution to (P_2^b).

Lemma 3.2.1 *For each x feasible for* (Q)*, for each $vv_1v_2 \in T$ either $x(v_1v_2) = 2$ or*

$$x(vv_1) = x(vv_2) = x(v_1v_2) = 1.$$

Proof: First observe that $x(v_1v_2) \geq 1$, for otherwise

$$x(\delta(v_1)) \leq x(vv_1) \leq \bar{c}(vv_1) = 1 < 2 = \bar{b}_{v_1}$$

and so x is not feasible for (Q). Now suppose for a contradiction that the claim is false. Then $x(v_1v_2) = 1$ and, without loss of generality, $x(vv_1) = 0$. But then

$$x(\delta(v_1)) = x(vv_1) + x(v_1v_2) = 1 < 2 = \bar{b}_{v_1},$$

contradicting that x is a feasible for (Q). ■

For an edge set $F \subseteq \bar{E}$, and a node $v \in \bar{V}$, define

$$\delta_F(v) := \{e \in F : v \in e\},$$

and define $d_F(v) := |\delta_F(v)|$.

Lemma 3.2.2 *Let $\bar{M}$ be the $\bar{c}$-capacitated $\bar{b}$-matching defined by the support of a vertex solution to (Q). Let M be the restriction of $\bar{M}$ to the edges of G. Let $v \in \{v \in V : b_v = 2\}$. The following hold:*

1. *If $d_M(v) = 0$ then vv_1, vv_2, and v_1v_2 are edges of $\bar{M}$.*
2. *If $d_M(v) = 1$ then vv' is an edge of $\bar{M}$.*
3. *If $d_M(v) = 2$ then v_1v_2 appears twice in the multiset $\bar{M}$.*

Proof: The third claim is immediate from Lemma 3.2.1. The first claim follows from Lemma 3.2.1 and the necessity that v has two incidence edges in $\bar{M}$. The second claim follows from the same reasoning as the first claim. ■

Theorem 3.2.3 *The optimal values of* (P_2^b) *and* (Q) *are equal.*

Proof: Let $M \in \mathcal{M}_b$ be an optimal solution to (P_2^b). Then M is a node-disjoint union of a family of cycles, which we will denote by $\mathcal{C}$, and a family of paths which we will denote by $\mathcal{P}$. Let

$$B := \{v \in V : v \text{ is an endpoint of some path } P \in \mathcal{P} \text{ and } b_v = 2\}.$$

Then $\bar{w}(M)$ overcounts $w(M) - p(V(M))$ by precisely $p(B)/2$. Formally,

$$\begin{aligned}\bar{w}(M) &= \sum_{P\in\mathcal{P}} \bar{w}(P) + \sum_{C\in\mathcal{C}} \bar{w}(C)\\ &= \frac{p(B)}{2} + \sum_{P\in\mathcal{P}} (w(P) - p(V(P))) + \sum_{C\in\mathcal{C}} (w(C) - p(C))\\ &= w(M) - p(V(M)) + \frac{p(B)}{2}.\end{aligned}$$

Let $E_B = \{vv' : v \in B\}$. Then $\bar{w}(E_B) = -p(B)/2$. So we have that

$$\bar{w}(M \cup E_B) = \bar{w}(M) + \bar{w}(E_B) = w(M) - p(V(M)).$$

So if we let $\bar{M}_0 = M \cup E_B$ then $\bar{M}$ is a $\bar{c}$-capacitated, $\bar{b}$-matching in $\bar{G}$ with $\bar{w}$-weight equal to the optimal value of (P_2^b). It remains to add edges to $\bar{M}_0$ to create a feasible solution to (Q) decreasing its $\bar{w}$ value.

Observe that every node that is covered by $\bar{M}_0$ is covered by precisely its $\bar{b}$-value number of edges. Further observe that each pair of dummy nodes v_1, v_2 for each triangle $vv_1v_2 \in T$ is $\bar{M}_0$ exposed. Construct $\bar{M}$ from $\bar{M}_0$ by adding the following edges for each $v \in V$ such that $b_v = 2$. Add edges $\{vv_1, vv_2, v_1v_2\}$ if v is $\bar{M}_0$ exposed; if v is $\bar{M}_0$ covered then add edge v_1v_2 twice to $\bar{M}$ (recalling that $\bar{M}$ is a multiset). It is not hard to see the resulting multiset $\bar{M}$ is feasible for (Q). The edges added to $\bar{M}_0$ to obtain $\bar{M}$ all have cost 0 and hence $\bar{M}$ is a solution to (Q) of weight exactly that of an optimal solution to (Q). Thus the value of (P_2^b) is at most that of (Q).

Now suppose that x is an optimal extreme point solution to (Q). Let $\bar{M}$ be the $\bar{c}$-capacitated, $\bar{b}$-matching in $\bar{G}$ obtained from the support of x. Let M be the restriction of $\bar{M}$ to the edges of G. Let $M' := \bar{M} \backslash E(T)$ be obtained from $\bar{M}$ by deleting all edges vv_1, vv_2, v_1v_2 such that $v \in V(\bar{M})$. Observe that $\bar{w}(\bar{M}) = \bar{w}(M')$, and the restriction of M' to edges of G is equal to M. Now we have,

$$\bar{w}(M) = w(M) - p(V(M)) + \sum_{v \in V(M): b_v = 2, d_M(v) = 1} \frac{1}{2} p(v)$$

by our choice of weights $\bar{w}$. But by Lemma 3.2.2, each node $v \in V$ such that $b_v = 2$ and $d_M(v) = 1$ is also matched to their corresponding node v' in M' in addition to their partner in M. Further, such vv' edges are the only edges in $M' \backslash M$. Hence by the choice of weight $\bar{w}$,

$$\bar{w}(M') = \bar{w}(M) - \sum_{v \in V(M): b_v = 2, d_M(v) = 1} \frac{1}{2} p(v) = w(M) - p(V(M)).$$

Hence the value of (Q) is at most the value of (P_2^b) and thus the result holds. ■

Since we can solve (Q) in polynomial time, this Theorem implies we can separate over the leastcore of simple b-matching games where $b_v \leq 2$ for all $v \in V$, and hence we can compute a leastcore allocation of such games. Consequently, via the Faigle, Kern, and Kuipers algorithm [30] we can compute points in the intersection of leastcore and kernel in polynomial time for b-matching games with $b \leq 2$. If the leastcore intersect kernel is unique for such an instance we can compute the nucleolus.

3.3 Efficient Algorithms in Special Cases

In the case of b-matching games where $b_v \leq 2$ for all $v \in V$, computing the nucleolus is a major open problem in the area. As we discussed in Section 3.2, b-values higher than one pose a significant challenge in studying the minimum excess coalitions. Even the case of $b \leq 2\chi(V)$-matching games on unweighted bipartite graphs poses a challenge compared to their matching game counterparts. Leastcore allocations no longer correspond to optimal dual solutions to the fractional b-matching problem. Moreover it is no longer clear whether it is possible to obtain a compact formulation of every linear program in the MPS Scheme in this, as was done in Theorem 2.1.1. The analogous linear programs for $b \leq 2\chi(V)$-matching games would have a constraint for every path whose internal vertices have b-value 2 and for every cycle whose vertices have b-value 2, yielding exponentially sized linear programs. This is not an issue for finding core allocations [8] but for higher rounds when avoiding fixed coalitions is involved things become more challenging.

In this section we will discuss some special cases of $b \leq 2\chi(V)$-matching games where the nucleolus can be efficiently computed. We will show in Section 3.3.1 that the nucleolus is easy to compute when there are few vertices of with b-value 2. In Section 3.3.2 we will treat the non-simple case in bipartite graphs when every vertex has b-value 2. These are not groundbreaking results, but they do give us an occasion to discuss the method of characterization sets [38]. This is an alternative technique for computing the nucleolus by showing that we need only concern ourselves this a polynomially sized subset of coalitions.

Let (n, ν) be a cooperative game. Let $\mathcal{S}$ be a collection of subsets of $[n]$, let $x \in \mathbb{R}^n$, and write

$$\theta^{\mathcal{S}}(x) \in \mathbb{R}^{\mathcal{S}} \tag{3.16}$$

to denote containing the excess values $\text{ex}(x, S)$ for $S \in \mathcal{S}$ in non-decreasing order of excess. In other words $\theta^{\mathcal{S}}(x)$ is the projection of $\theta(x)$ onto $\mathbb{R}^{\mathcal{S}}$.

We say $\mathcal{S}$ is a *characterization set* for the nucleolus of the cooperative game (n, ν) if the lexicographic maximizer of $\theta^{\mathcal{S}}(x)$ is a singleton that is equal to $\eta(n, \nu)$. Intuitively, the nucleolus of the game restricted to $\mathcal{S}$ is the same as the nucleolus of (n, ν).

Observe that if $\mathcal{S}$ is a characterization set for (n, ν) then for $S \in 2^{[n]} \setminus \mathcal{S}$, we can drop the constraint corresponding to S from the Kopelowitz Scheme when computing the nucleolus.

Proposition 3.3.1 *Let (n, ν) be a cooperative game with non-empty core. Suppose $\mathcal{S}$ is a polynomial sized characterization set for the nucleolus of (n, ν).*

Then the nucleolus of (n, ν) is polynomial time computable.

Proof: Let $\mathcal{S}$ be a characterization set of the nucleolus of some game (n, ν). Consider the following modification of the ℓ-th iteration of Kopelowitz Scheme (K'_ℓ) (with optimal value ε'_ℓ) where we only have constraints corresponding to coalitions in the characterization set $\mathcal{S}$ instead of every coalition.

$$\begin{aligned} &\max \varepsilon && && (K'_\ell) \\ &x(S) = \nu(S) - \varepsilon'_k && \forall 0 \leq k < \ell, \forall S \in \mathcal{S}_k \\ &x(S) \geq \nu(S) - \varepsilon && \forall S \in \mathcal{S} \cup \{\varnothing, N\} \setminus \bigcup_{k=0}^{\ell-1} \mathcal{S}_k \end{aligned}$$

Here $\mathcal{S}_k$ denotes the tight coalitions at the k-th round of this modified Kopelowitz Scheme.

The tweaked Kopelowitz Scheme computes the lexicographic maximizer of $\theta^{\mathcal{S}}$ by definition. Since $\mathcal{S}$ is polynomially sized, each linear program in the scheme can be solved in polynomial time (for instance, by the ellipsoid method). ■

The following theorem gives a sufficient condition for identifying a characterization set.

Theorem 3.3.2 (Granot, Granot, and Zhu[38]) *Let (n, ν) be a cooperative game with non-empty core.*

Suppose $\mathcal{S}$ is a collection of subsets of $[n]$ which is a characterization set for the nucleolus of (n, ν). if for every $S \in 2^{[n]} \setminus \mathcal{S}$ there exists a non-empty subcollection $\mathcal{S}_S$ of $\mathcal{S}$ such that

(i) For all $T \in \mathcal{S}_S$ and core allocations x, $\mathrm{ex}(x, T) \le \mathrm{ex}(x, S)$.

(ii) There are scalars $\lambda_T \in \mathbb{R}$ such that the characteristic vector $\chi_S \in \{0, 1\}^N$ of S satisfies

$$\chi_S = \sum_{T \in \mathcal{S}_S \cup \{N\}} \lambda_T \chi_T. \tag{3.17}$$

We can obtain a characterization set for b-matching games as a corollary to this theorem.

Corollary 3.3.3 *Let (n, ν) be a not necessarily simple weighted b-matching game with non-empty core. Define*

$$\mathcal{S} := \{S \subseteq [n] : \textit{For all maximum } b\textit{-matchings } M \textit{ of } G[S],\ (S, M) \textit{ is connected}\}$$

Then $\mathcal{S}$ is a characterization set for the nucleolus of Γ.

Proof: Fix $S \in 2^{[n]} \setminus \mathcal{S}$. Suppose M is a maximum b-matching of $G[S]$. Let $T_1, T_2, \ldots, T_k$ be the components of (S, M) for $k \ge 2$. Suppose x is a core allocation.

Since $x(S) = \sum_{i=1}^k x(T_i)$ and $\nu(S) = \sum_{i=1}^k \nu(T_i)$, we have

$$\sum_{i=1}^k \mathrm{ex}(S_i, x) = \mathrm{ex}(S, x). \tag{3.18}$$

In particular, condition (ii) of Theorem 3.3.2 is satisfied.

But all excesses are non-negative as x is a core allocation, hence each $\mathrm{ex}(S_i, x) \le \mathrm{ex}(S, x)$ and condition (i) of Theorem 3.3.2 is also satisfied, as desired. ■

Lemma 3.3.4 *Let (n, ν) be a not necessarily simple weighted b-matching game with non-empty core. Suppose*

$$\mathcal{S} := \{S \in 2^{[n]} : \textit{For all maximum } b\textit{-matchings } M \textit{ of } G[S],\ (S, M) \textit{ is connected}\}.$$

is polynomially sized.

Then the nucleolus of (n, ν) is polynomial time computable.

Proof: Apply Proposition 3.3.1 and Corollary 3.3.3. ■

Notice that for matching games with non-empty core the characterization set $\mathcal{S}$ from Corollary 3.3.3 is E unioned with the singleton coalitions. Thus Lemma 3.3.4 provides a quick and easy proof of Theorem 2.1.1.

3.3.1 Simple b-Matching Games

Theorem 3.3.5 *Let G be a bipartite graph with bipartition $N = A\dot{\cup}B$ and $k \geq 0$ a constant.*

Suppose $b_v = 2$ for all $v \in A$ but $b_v = 2$ for at most k vertices of B, then the nucleolus of the simple b-matching game on G can be computed in polynomial time.

Proof: By Lemma 3.3.4, it suffices to show that any component of a b-matching in some arbitrary induced subgraph $G[S]$ has at most $2k + 3$ vertices. If we show this, then the set

$$\mathcal{S} := \{S \in \mathcal{S} : \text{For all maximum } b\text{-matchings } M \text{ of } G[S],\ (S, M) \text{ is connected}\}$$

is polynomially sized since it is contained in the subsets of $V(G)$ of size at most $2k + 3$.

Let C be a component of (S, M) for some $S \subseteq N$ and maximum b-matching M of $G[S]$.

If C is a cycle, then exactly half the vertices of C are from B with $b_v = 2$. It follows that $|C| \leq 2k$.

Suppose now that C is some path. By deleting at both endpoints and one more vertex, we may assume that every other vertex in the path are from B with $b_v = 2$. Thus $|C| \leq 2k + 3$ as required. ■

This result can be modified for the case where at most $O(\log(n + m))$ vertices in total have $b_v = 2$.

3.3.2 Non-Simple 2-Matching Games

In the case where we allow for edges to be included multiple times in a 2-matching, we leverage core non-emptiness and the non-existence of odd cycles to compute the nucleolus in polynomial time.

Consider the following LP formulation of the maximum non-simple b-matching from [73].

$$\begin{aligned}
\max\ & w^T y && && (P_b)\\
\text{s.t. } & y(\delta(v)) \leq b_v && \forall v \in V &&\\
& y(E[U]) \leq \left\lfloor \frac{b(U)}{2} \right\rfloor && \forall U \subseteq V, b(U) \text{ is odd} && (3.19)\\
& y \geq 0 && &&
\end{aligned}$$

Observe that for the case $b \equiv 2$, $b(U)$ is never odd and hence there are no constraints of the form (3.19).

Thus the dual of non-simple 2-matching games is simplified to the following.

$$\begin{aligned} \min\ & 2\sum_{v\in V} x_v && \\ \text{s.t.}\ & x_u + x_v \geq w(uv) && \forall uv \in E \\ & x \geq 0 && \end{aligned} \tag{D_b}$$

Lemma 3.3.6 *Let G be an arbitrary graph with edge weights $w : E \to \mathbb{R}$.*

The core of the weighted non-simple 2-matching game on G is non-empty.

Proof: Let $\bar{x}$ be an optimal solution to (D_b). Since $2\bar{x}(N) = \nu(N)$ by the integrality of (P_b), $2\bar{x}$ is an allocation.

Fix a coalition $\varnothing \neq S \subsetneq N$. Define ($D_S$) as the dual to the non-simple 2-matching LP (P_b) on $G[S]$. Write $2\bar{x} \mid_S$ as the restriction of $2\bar{x}$ to entries indexed by vertices of S.

Observe that $2\bar{x} \mid_S$ is feasible in (D_S), thus $\nu(S) \leq 2\bar{x}(S) = 2\bar{x} \mid_S (S)$ by weak duality and

$$2\bar{x}(S) - \nu(S) \geq 0.$$

By the arbitrary choice of S, $2\bar{x}$ is a core allocation and consequently, the core is non-empty.
■

Notice that since $b = 2\chi(V)$, we did not need to assume G to be bipartite.

Lemma 3.3.7 *For any bipartite graph, there is a maximum weighted non-simple 2-matching consisting only of parallel edges.*

Proof: Let M be a maximum non-simple 2-matching in G. Observe that the components of $(V(M), M)$ are parallel edges, even cycles, and paths. Moreover, any path contains at least 2 edges, or else adding that single edge a second time to our matching can only increase the weight of the matching.

We argue that all vertex solutions to (P_b), such as $\chi(M)$, contain no even cycles nor paths containing at least 2 edges. Hence an optimal vertex solution has the desired properties.

Case I: Even Cycles Suppose $C \subseteq M$ for some even cycle C. Enumerate the edges

$$C : e_1, e_2, e_3, \ldots, e_k$$

for some $k \equiv 0 \mod 2$.

For $i = 0, 1$, put

$$M^{(i)} := M \setminus C \cup \{e_j, e_j : j \equiv i \mod 2\}.$$

That is, double up every other edge of C.

Then

$$\chi(M) = \frac{1}{2}\chi(M^{(0)}) + \frac{1}{2}\chi(M^{(1)})$$

and so $\chi(M)$ was not a vertex solution.

Case II: Paths of Length At Least 2 Let $P \subseteq M$ be a path of length at least 2. Enumerate the edges

$$P : e_1, e_2, \dots, e_k.$$

Similarly to the previous case, define for $i = 0, 1$

$$M^{(i)} := M \setminus P \cup \{e_j, e_j : j \equiv i \mod 2\}.$$

Then

$$\chi(M) = \frac{1}{2}\chi(M^{(0)}) + \frac{1}{2}\chi(M^{(1)})$$

and so $\chi(M)$ was not a vertex solution. ■

We are now ready to prove the result.

Theorem 3.3.8 *Let G be a bipartite graph.*

The nucleolus of the non-simple 2-matching game on G can be computed in polynomial time.

Proof: By Lemma 3.3.4, it suffices to show that if $|S| \geq 3$, then there is a 2-matching in $G[S]$ with multiple components.

But this is precisely what we proved in Lemma 3.3.7, concluding the proof. ■

Unfortunately, Lemma 3.3.7 does not hold when the graph is non-bipartite, even when we restrict ourselves to uniform edge weights. Indeed, consider the simple triangle. The maximum non-simple 2-matching has size 3. However, when we restrict ourselves to matchings composed of only parallel edges, the maximum matching we can obtain has cardinality 2.

Similarly, Lemma 3.3.7 does not in general hold when there are some vertices v where $b_v = 1$. Consider the path of 3 edges where the endpoints have $b_v = 1$ while the internal vertices have $b_v = 2$. The maximum non-simple 2-matching has size 3. However, if we only allow parallel edges, the maximum matching we can obtain again has cardinality 2.

Chapter 4

A Dynamic Programming Framework For Computing the Nucleolus

This chapter studies the relationship between the nucleolus, finding the minimum excess coalition with respect to an alloation, congruency-constrained optimization, and dynamic programming. Our first result unifies these areas and provides a general method for computing the nucleolus.

Theorem 4.0.1 *For any cooperative game (n, ν), if the minimum excess coalition problem on (n, ν) can be solved in time T via an integral dynamic program then the nucleolus of (n, ν) can be computed in time polynomial in T.*

This framework is inspired by work done by Elkind and Pasechnik [26], and work done by Pashkovich [63] on computing the nucleolus of weighted voting games. Section 4.1 will discuss the necessary background on weighted voting games. That section will also discuss the question of how many rounds Kopelowitz Scheme needs to compute the nucleolus of a weighted voting game, an interesting puzzle in the area.

Pashkovich [63] showed how to reduce the problem of computing the nucleolus for weighted voting games to a congruency-constrained optimization problem. Pashkovich then shows how to solve this congruency-constrained optimization problem for this specific class of games via a dynamic program. In Section 4.2.2 we abstract his reduction to the setting of computing the nucleolus of general combinatorial optimization games.

Our main technical achievement is showing that adding congruency constraints to dynamic programs modelled by a directed acyclic hypergraph model inspired by the work of Campbell, Martin, and Rardin [12] adds only a polynomial factor to the computational complexity. This is the content of Theorem 4.3.7, which is instrumental in demonstrating Theorem 4.0.1. Our formal model of dynamic programming, where solutions correspond to directed hyperpaths in a directed acyclic hypergraph, is described in Section 4.3. Proving Theorem 4.3.7 requires new techniques beyond [63]. These new techniques are new needed to handled the extra complexity of hyperarcs in the hypergraph abstraction. The series of lemmas in Section 4.3.1 take the reader through these techniques for manipulating directed acyclic hypergraph dynamic programs.

We show how Theorem 4.0.1 not only generalizes previous work on computing the nucleolus, but extends our capabilities to new classes of combinatorial optimization games that were not possible with just the ideas in [63]. In Section 4.4 we will discuss applications. We will begin by discusing how our result generalizes the weighted voting games result, and we will show how to compute the nucleolus of b-matching games on graphs of bounded treewidth.

Theorem 4.0.2 *For any cooperative b-matching game on a graph whose treewidth is bounded by a constant, the nucleolus can be computed in polynomial time.*

We will also show how our results give an easier algorithm for computing the nucleolus of b-matchng games on bipartite graphs when one side has b-values all 1 than the one known in the literature [4].

By the complexity discussion in Chapter 3, it is necessary to impose some structure on b-matching games to compute their nucleolus in polynomial time unless $\mathsf{P} = \mathsf{NP}$. In Sectino 3.3 we studied bipartite graphs with limitations on the b-values. In Theorem 4.0.2 we impose the structure of bounded treewidth.

To achieve this result we give a dynamic program for computing the minimum excess coalition of a b-matching game in Lemma 4.4.4 then apply Theorem 4.0.1. This dynamic program necessarily requires the use of dynamic programming on hypergraphs instead of just simple graphs, motivating the increased complexity of our model over previous work.

4.1 Weighted Voting Games

A *weighted voting game* is a cooperative game (n, ν) compactly represented by a weight vector $w \in \mathbb{Z}^n$ and a threshold $T \in \mathbb{Z}$. For any coalition of players, $S \subseteq [n]$, the value function is defined to satisfy

$$\nu(S) = \begin{cases} 1, & \text{if } w(S) \geq T \\ 0, & \text{otherwise.} \end{cases}$$

The intuition is that of a collection of voters who have different weights w_i indicating their voting power who are voting on a motion which needs a threshold T of total voter weight to pass.

Elkind and Pasechnik [26] gave an algorithm to solve each linear program in Kopelowitz Scheme for weighted voting games in time pseudopolynomial in the input size for the compact representation of the game. Their claim was that this gave a pseudopolynomial time algorithm for computing the nucleolus of weighted voting games. Unfortunately, as we saw in Section 2.1.1, it is possible for Kopelowitz Scheme to run in an exponential number of rounds in the number of players and so their proof was not sufficient to conclude that there is a pseudopolynomial time algorithm for computing the nucleolus of weighted voting games. Note that computing the nucleolus of a weighted voting game is known to be NP-hard [25].

Notice that the example given in Section 2.1.1 is very different from a weighted voting game in the structure of its value function. Whereas weighted voting games only assign coalitions values in $\{0, 1\}$, the exponential rounds example given assigns coalitions values amongst an exponential number of different values. It then fixes the value allocated to most players in the nucleolus to 1 and uses the many varying coalition values to argue an exponential number of different coalition excesses with respect to the nucleolus (and hence an exponential number of rounds in Kopelowitz Scheme). Such a technique would not work for weighted voting games where only two different values for a coalition are possible.

It is an open question how many rounds Kopelowitz Scheme needs in the literature to compute the nucleolus of a weighted voting game. In Corollary 4.1.2 we prove that when the player weights are drawn from a constant number of distinct weights, Kopelowitz Scheme only needs a polynomial number of iterations to compute the nucleolus.

Lemma 4.1.1 *Let x^* be the nucleolus of a weighted voting game $(n, \nu))$ determined by weights $w \in \mathbb{Z}^n$ and threshold $T \in \mathbb{Z}$. If $w_i \geq w_j$ for some $i, j \in [n]$ then $x^*_i \geq x^*_j$.*

Proof: Suppose not, i.e. there exist $i, j \in [n]$ such that $w_i \geq w_j$ but $x^*_i < x^*_j$.

Let x' be the allocation obtained from x^* by exchanging the values of on i and j, more precisely

$$x' := x^* - \left(x^*_i e_i + x^*_j e_j\right) + \left(x^*_j e_i + x^*_j e_j\right),$$

where $e_i = \chi(\{i\})$ and $e_j = \chi(\{j\})$ are the i-th and j-th standard basis vectors. Then x' is individually rational and efficient, so it is an imputation

Consider the sequence of polytopes

$$K_1(\varepsilon^K_1), \ldots, K_N(\varepsilon^K_N)$$

generated by Kopelowitz Scheme to compute the nucleolus of (n, ν). Since the nucleolus is unique, x' is not the nucleolus. Thus there exists k such $x' \notin K_k(\varepsilon^K_k)$. Choose such k as small as possible.

Then there exists $S \subseteq [n]$ such that $x'(S) - \nu(S) < \varepsilon_k$ but $x^*(S) - \nu(S) = \varepsilon_k$. By the construction of x' we have $j \in S$ and $i \notin S$. Now observe that

$$\begin{aligned} x^*(S) - \nu(S) &= x^*(S\Delta\{i,j\}) - x_i^* + x_j^* - \nu(S) \\ &> x^*(S\Delta\{i,j\}) - \nu(S) && \text{(since } x_j^* > x_i^*\text{)} \\ &\geq x^*(S\Delta\{i,j\}) - \nu(S\Delta\{i,j\}) && \text{(since } w(S\Delta\{i,j\}) \geq w(S)\text{)} \end{aligned}$$

Thus $x^*(S\Delta\{i,j\}) - \nu(S\Delta\{i,j\}) < \varepsilon_k^K$. So by the feasibility of x^* for $K_k(\varepsilon_k^K)$, there exists $\ell < k$ such that

$$x^*(S\Delta\{i,j\}) - \nu(S\Delta\{i,j\}) = \varepsilon_\ell,$$

and moreover by Kopelowitz Scheme $S\Delta\{i,j\}$ is *universal at level* ℓ: for all $x \in K_\ell(\varepsilon_\ell^K)$, $\mathrm{ex}(x, S\Delta\{ij\}) = \varepsilon_\ell^K$.

But now observe that

$$\begin{aligned} x'(S\Delta\{i,j\}) - \nu(S\Delta\{i,j\}) &= x^*(S) - \nu(S\Delta\{i,j\}) \\ &= x^*(S\Delta\{i,j\}) - x_i^* + x_j^* - \nu(S\Delta\{i,j\}) \\ &> x^*(S\Delta\{i,j\}) - \nu(S\Delta\{i,j\}) \\ &= \varepsilon_\ell. \end{aligned}$$

Since $x' \in K_\ell(\varepsilon_\ell^K)$ by the minimality of k, this contradicts that $S\Delta\{i,j\}$ is universal at level ℓ. ■

Corollary 4.1.2 *Consider an instance of a weighted voting game (n, ν) with weight vector $w \in Z^n$ and threshold $T \in \mathbb{Z}$. If $|\{w_i : i \in [n]\}| = O(1)$ then Kopelowitz Scheme computes the nucleolus of this instance in a polynomial number of rounds.*

Proof: Consider the nucleolus x^* of the instance. By Lemma 4.1.1, if $w_i = w_j$ then $x_i^* = x_j^*$, so we have

$$|\{x_i^* : i \in [n]\}| \leq |\{w_i : i \in [n]\}| = O(1).$$

Let $D := \{x_i^* : i \in [n]\}$. We want to upper bound $|\{x^*(S) : S \subseteq [n]\}|$. The value of $x^*(S)$ is uniquely determined by the number of players i in S such that $x_i^* = d$ for each $d \in D$. Formally

$$x^*(S) = \sum_{d \in D} s_d d$$

where s_d is the number of players i in S for which $x_i^* = d$. There are at most n players i such that $x_i^* = d$. Thus there are at most $n^{|D|}$ possible values for $x^*(S)$. Thus there are at most $n^{|D|}$ values for $x^*(S) - \nu(S)$. Let ε_ℓ^K is the optimal value of the ℓ-th round linear program in Kopelowitz Scheme, and let N be the number of rounds Kopelowitz Scheme runs for.

Observe that $\varepsilon_1 < \varepsilon_2 < \cdots < \varepsilon_N$, and each ε_ℓ value is equal to $x^*(S) - \nu(S)$ for some S since x^* is universal at level ℓ. Therefore N is at most $n^{|D|}$. Since $|D| = O(1)$ this implies Kopelowitz Scheme runs in a polynomial number of rounds. ■

It is not hard to see that (n, ν) is completely determined by (w, T). In this case (w, T) is a compact representation of the weighted voting game (n, ν). Even though they may appear simple at first, weighted voting games can have a lot of modelling power. In fact the voting system of the European Union can be modelled by a combination of weighted voting games [5]. In [25] Elkind, Goldberg, Goldberg, and Wooldridge show that the problem of computing the nucleolus of a weighted voting game is NP-hard, in fact even the problem of testing if there is a point in the leastcore of a weighted voting game that assigns a non-zero payoff to a given player is NP-complete. Pashkovich [63] later followed up with an algorithm based on the MPS Scheme which solves $O(n)$ linear programs, each in pseudopolynomial time, and thus computes the nucleolus of a weighted voting game in pseudopolynomial time.

Pashkovich's result crucially relies on the existence of a well-structured dynamic program for knapsack cover problems which runs in pseudopolynomial time. Theorem 4.0.1 and Section 4.4 place Pahskovich's algorithm in the context of a general framework for computing the nucleolus of cooperative games where a natural associated problem has a dynamic program: the *minimum excess coalition problem*.

In the minimum excess coalition problem the given input is a compact representation of a cooperative game (n, ν) with characterization set $\mathcal{S}$, and an imputation x. The goal is to output a coalition $S \subseteq \mathcal{S}$ which minimizes excess, i.e. $x(S) - \nu(S)$, with respect to x.

4.2 Computing the Nucleolus Via Congruency-Constrained Optimization

4.2.1 Relaxed MPS Scheme

Since the MPS Scheme ends after at most n linear program solves, the run time of the method is dominated by the time it takes to solve (P_i). To use the Ellipsoid Method [46, 59] to implement the MPS Scheme we need be able to separate over the constraints corresponding to all coalitions in $\text{Fix}(P_{i-1}(\varepsilon_{i-1}))$ in each iteration. There can be an exponential number of such constraints in general, and some structure on the underlying cooperative game would need to be observed in order to separate these constraints efficiently. This requirement can be relaxed somewhat, and still retain the linear number of iterations required to compute the nucleolus. At the same time we would like to incorporate the use of characterization set techniques (see Section 3.3) in our relaxation of the MPS Scheme.

Let (n, ν) be a cooperative game and let $\mathcal{S}$ be a characterization set for (n, ν). We will define a sequence of linear programs $Q_1, Q_2, \dots, Q_N$ where the unique optimal solution to Q_N is the nucleolus of (n, ν). With each linear program Q_i there will be an associated set of vectors V_i contained in the set of incidence vectors of $\text{Fix}(Q_i)$. The feasible solutions to Q_i will lie in $\mathbb{R}^n \times \mathbb{R}$.

In keeping with the notion we used for $P_i(\varepsilon_i)$, for each linear program Q_i we let $\bar{\varepsilon}_i$ be the optimal value of Q_i and let

$$Q_i(\bar{\varepsilon}_i) := \{x \in \mathbb{R}^n : (x, \bar{\varepsilon}_i) \text{ is feasible for } Q_i\}.$$

We will describe the linear programs $\{Q_i\}_i$ inductively. The first linear program is the least-core linear program of (n, ν) restricted to coalitions in $\mathcal{S}$. That is to say Q_1 is equal to

$$\begin{aligned} &\max\ \varepsilon && && (Q_1)\\ &\text{s.t. } x(S) \geq \nu(S) + \varepsilon && \forall S \in \mathcal{S} : \chi(S) \notin \operatorname{span}(V_0) \\ &\qquad x \in \mathcal{I}(n, \nu). \end{aligned}$$

where $V_0 = \{\chi([n])\}$.

Let $V_1 \subseteq \mathbb{R}^n$ be obtained from V_0 by adding the incidence vector of one coalition in $\mathcal{S} \cap \operatorname{Fix}(Q_1(\bar{\varepsilon}_1)) \setminus \operatorname{span}(V_0)$. Now given Q_{i-1} and V_{i-1} we describe Q_i as follows

$$\begin{aligned} &\max\ \varepsilon && && (Q_i)\\ &\text{s.t. } x(S) \geq \nu(S) + \varepsilon && \forall S \in \mathcal{S} : \chi(S) \notin \operatorname{span}(V_{i-1}) \\ &\qquad x \in Q_{i-1}(\bar{\varepsilon}_{i-1}). \end{aligned}$$

Now we choose $v \in \mathcal{S} \cap \operatorname{Fix}(Q_i(\bar{\varepsilon}_i)) \setminus \operatorname{span}(V_{i-1})$ and set $V_i := V_{i-1} \cup \{v\}$. By the optimality of $\bar{\varepsilon}_i$, v always exists as long as $Q_i(\bar{\varepsilon}_i)$ has affine dimension at least 1. If $Q_i(\bar{\varepsilon}_i)$ has affine dimension 0 we terminate the procedure and conclude that $Q_i(\bar{\varepsilon}_i)$ is a singleton containing the nucleolus.

Lemma 4.2.1 *([63]) When the Relaxed MPS Scheme is run on a cooperative game (n, ν) yielding a hierarchy of linear programs $Q_1, \dots, Q_N$, with optimal values $\bar{\varepsilon_1}, \dots, \bar{\varepsilon_N}$ respectively, the set $Q_N(\bar{\varepsilon_N})$ is a singleton containing the nucleolus of (n, ν). Moreover N is at most n.*

Proof: The claim that N is at most n follows since the dimension of V_i increases by 1 in every round and the dimension of the ambient space is n. It remains to argue that the relaxed MPS Scheme computes the nucleolus.

By construction $Q_N(\bar{\varepsilon})$ is a singleton since the set has affine dimension 0. To see that it contains the nucleolus we claim that in fact every set $Q_i(\bar{\varepsilon}_i)$ contains the nucleolus. Let $x^* = \eta(n, \nu)$.

Suppose for a contradiction there exists i such that $Q_i(\bar{\varepsilon}_i)$ does not contain the x^*. Choose i as small as possible. Let $x' \in Q_i(\bar{\varepsilon}_i)$. Then there exists $S \in \mathcal{S} \setminus \operatorname{span}(V_{i-1})$ such that

$$\bar{\varepsilon}_1 = \operatorname{ex}(x', S) < \operatorname{ex}(x^*, S).$$

Moreover, using the minimality of i, we can choose S so that for any $S' \in \mathcal{S}$ such that $\operatorname{ex}(x^*, S') < \operatorname{ex}(x^*, S)$ we have $\operatorname{ex}(x^*, S') = \operatorname{ex}(x', S')$.

But then $\theta^{\mathcal{S}}(x')$ is lexicographically smaller than $\theta^{\mathcal{S}}(x^*)$, which contradicts that x^* is the nucleolus of (n, ν) and $\mathcal{S}$ is a characterization set of (n, ν).

■

4.2.2 The Linear Subspace Avoidance Problem

Motivated by the desire to design a separation oracle for the constraints of (Q_i) we initiate a general study of combinatorial optimization problems whose feasible region avoids a linear subspace. Recall from Section 1.1 that a combinatorial optimization problem is an optimization problem of the form

$$\max\{f(x) : x \in \mathcal{X}\} \tag{P}$$

where $\mathcal{X} \subseteq \{0,1\}^n$ is known as the feasible region, and $f : \mathcal{X} \to \mathbb{R}$ is the objective function. Normally (P) is presented via a compact representation. For example in the shortest path problem on a directed graph, $\mathcal{X}$ is the family of paths in a directed graph D and $f(x)$ is a linear function. The entire feasible set $\mathcal{X}$ is uniquely determined by the underlying directed graph D, and f is determined by weights on the arcs of D. When giving as input D and the arc weights, the problem is completely determined without specifying every one of the exponentially many paths in $\mathcal{X}$.

For compactly represented cooperative games the minimum excess coalition problem can be phrased as a problem of the form (P). Simply take $\mathcal{X}$ to be the set of incidence vectors of subsets of $\mathcal{S}$ and take $f(x)$ to be $x(S) - \nu(S)$.

Now consider a linear subspace $\mathcal{L} \subseteq \mathbb{R}^E$. For our combinatorial optimization problem (P), the associated *linear subspace avoidance problem* is

$$\max\{f(x) : x \in \mathcal{X} \backslash \mathcal{L}\} \tag{$P_{\mathcal{L}}$}$$

Even when (P) can be solved in polynomial time with respect to its compact representation and $\mathcal{L}$ is given through a basis, ($P_{\mathcal{L}}$) can be NP-hard.

Lemma 4.2.2 *($P_{\mathcal{L}}$) is NP-hard in general even when (P) can be solved in polynomial time with respect to its compact representation and $\mathcal{L}$ is given through a basis. In fact, even deciding if the feasible region ($P_{\mathcal{L}}$) is non-empty is NP-complete.*

Proof: We begin by considering the TWO DISJOINT DIRECTED PATHS (TDDP) problem. This well-known NP-complete problem [33] gives as input a directed graph G with two source nodes s_1, s_2 and two sink nodes t_1, t_2, and the problem is to decide if there exists a pair of arc disjoint paths, one from s_1 to t_1 and the other from s_2 to t_2.

Add an arc (t_1, s_2) if it does not already exist. Call the new graph G'. Observe that there are arc disjoint $s_1 - t_1$ and $s_2 - t_2$ paths in G if and only if there is an $s_1 - t_2$ path in G' using the arc (s_2, t_1) in G'. Let $\mathcal{X}$ be the set of incidence vectors of $s_1 - t_2$ paths in G'. The graph G' with s_1, s_2, t_1, t_2 labelled may serve as a compact presentation of $\mathcal{X}$. Let $c \in \mathbb{R}^{E(G')}$ be the all-ones vector and let $f(x) := -c^T x$. Furthermore, the corresponding problem (P) can be solved in polynomial time with respect to the encoding size of G' and c.

Suppose we have an oracle which can solve the corresponding instance of $(P_{\mathcal{L}})$ in polynomial time for any linear subspace $\mathcal{L} \subseteq \mathbb{R}^{E(G')}$. Consider the particular linear subspace

$$\mathcal{L} := \{x \in \mathbb{R}^{E(G')} : x_{(t_1, s_2)} = 0\}.$$

Observe that there an $s_1 - t_2$ path in G' using arc $t_1 - s_2$ if and only if $\mathcal{X} \backslash \mathcal{L}$ is nonempty. Using our oracle for $(P_{\mathcal{L}})$ we can decide in polynomial time if this is the case. ■

Observe that when we formulate the minimum excess coalition problem for a cooperative game (n, ν) with characterization set $\mathcal{S}$ as a problem of the form (P) and we take $\mathcal{L} = \text{span}(V_{i-1})$ then $(P_{\mathcal{L}})$ is the separation problem for (Q_i), the i-th linear program in the relaxed MPS Scheme. This discussion yields the following easy lemma

Lemma 4.2.3 *If (P) is a minimum excess coalition problem of a cooperative game (n, ν) on characterization set $\mathcal{S}$ and one can solve the associated $(P_{\mathcal{L}})$ for any $\mathcal{L}$ in polynomial time then the nucleolus of (n, ν) can be computed in polynomial time.*

4.2.3 Reducing Linear Subspace Avoidance to Congruency-Constrainted Optimization

The goal of this subsection is to show the connection between congruency-constrained optimization and solving $(P_{\mathcal{L}})$. This connection was first drawn in the work of Pashkovich [63] for the special case of weighted voting games. Here we abstract their work to apply it to our more general framework.

By the following lemma, we can restrict our attention from linear independence over $\mathbb{R}$ to linear independence over finite fields. We present the proof for completeness.

Lemma 4.2.4 *(Pashkovich [63]) Let P be a set of prime numbers such that $|P| \geq \log_2(n!)$ with $n \geq 3$. A set of vectors $v_1, \ldots, v_k \in \{0,1\}^n$ are linearly independent over $\mathbb{R}$ if and only if there exists $p \in P$ such that $v_1, \ldots, v_k$ are linearly independent over $\mathbb{F}_p$.*

Moreover, the set P can be found in $O(n^3)$ time, and each p in P can be encoded in $O(\log(n))$ bits.

Proof: ([63]) Let A be the $n \times k$ matrix whose i^{th} column is vector v_i. If $v_1, \ldots, v_k$ are linearly independent over $\mathbb{F}_p$ then there exists a $k \times k$ submatrix B of A such that $\det(B) \neq 0$ (over $\mathbb{F}_p$). Then $\det(B) \neq 0$ over $\mathbb{R}$ and hence $v_1, \ldots, v_k$ are linearly independent over $\mathbb{R}$.

Now suppose that $v_1, \ldots, v_k$ are linearly dependent over $\mathbb{F}_p$ for all $p \in P$. If $k > n$ then clearly $v_1, \ldots, v_k$ are linearly dependent over $\mathbb{R}$ and we are done. So suppose that $k \leq n$. Let B be a $k \times k$ submatrix of A. We will show that $\det(B) = 0$.

For each $p \in P$, p divides $\det(B)$ since $v_1, \ldots, v_k$ are linearly dependent over $\mathbb{F}_p$. Note that $\det(B)$ is an integer since it is the determinant of a 0-1 matrix. Since each $p \in P$ is prime, this implies that

$$\prod_{p \in P} p \mid \det(B),$$

and hence, if $\det(B) \neq 0$ then

$$\prod_{p \in P} p \leq \det(B).$$

But in this case,

$$2^{\log_2(n!)} < \prod_{p \in P} p \leq \det(B) \leq n!,$$

with the last inequality following since B is a 0-1 matrix, yielding a contradiction. Therefore $\det(B) = 0$ as desired.

Now generate a set P of primes such that $|P| \geq \log_2(n!)$. By the Prime Number Theorem, we can simply find the first $\log_2(n!)$ primes in $O(n^3)$ time and thus we can construct P in time polynomial in n. Furthermore, the value of each prime in P will be polynomial in n (i.e. each prime in P can be encoded with $O(\log(n))$ bits).

■

This lemma enables us to reduce the problem ($P_{\mathcal{L}}$) to the problem of computing (P) subject to a congruency constraint with respect to a given prime p, $k \in \mathbb{Z}_p$, $v \in Z_p^E$:

$$\max\{f(x) : x \in \mathcal{X}, v^T x = k \mod p\}. \qquad (P_{\mathcal{L},p,v,k})$$

Lemma 4.2.5 *If one can solve ($P_{\mathcal{L},p,v,k}$) in time T then one can solve ($P_{\mathcal{L}}$) in time $O(n^6 T)$.*

Proof: For a linear subspace $\mathcal{L} \subseteq \mathbb{R}^n$, we say that $x \in \text{span}(\mathcal{L})$ over $\mathbb{F}_p$ if there exists a basis B of $\mathcal{L}$ such that $B \cup \{x\}$ is linearly dependent over the field $\mathbb{F}_p$. By Lemma 4.2.4 we can solve ($P_{\mathcal{L}}$) by solving,

$$\max\{f(x) : x \in \mathcal{X}, x \notin \text{span}(\mathcal{L}) \text{ over } \mathbb{F}_p \text{ for some } p \in P\}$$

The above problem can be solved by solving for each $p \in P$,

$$\max\{f(x) : x \in \mathcal{X}, x \notin \text{span}(\mathcal{L}) \text{ over } \mathbb{F}_p\} \qquad (P_{\mathcal{L},p})$$

and taking the solution of maximum objective value found. Let B_p be a basis of $\mathcal{L}^\perp$ over $\mathbb{F}_p$. Assuming $\mathcal{L}$ is presented to us through a fixed basis, we can compute B_p in polynomial time via Gaussian Elimination [24]. Now, x is not in the span of $\mathcal{L}$ over $\mathbb{F}_p$ if and only if there exists $v \in B_p$ such that $v^T x \neq 0 \mod p$. Hence we can solve ($P_{\mathcal{L},p}$) by solving for each $v \in B_p$,

$$\max\{f(x) : x \in \mathcal{X}, v^T x \neq 0 \mod p\} \qquad (P_{\mathcal{L},p,v})$$

and taking the solution of maximum value found. Now $v^T x \neq 0 \mod p$ if and only if there exists $k \in [p-1]$ such that $v^T x = k$ (over $\mathbb{F}_p$). Thus we can solve ($P_{\mathcal{L},p,v}$) by solving for each $k \in [p-1]$, ($P_{\mathcal{L},p,v,k}$) and taking the solution of maximum value found.

Hence by solving $O(|P|(n - \dim(\mathcal{L})) \max_{p\in P} p) = O(n^3 \max_{p \in P} p) = O(n^6)$ congruency-constrained optimization problems of the form ($P_{\mathcal{L},p,v,k}$) we can solve ($P_{\mathcal{L}}$).

■

4.3 Dynamic Programming

Our goal is to define a class of problems where tractability of (P) can be lifted to tractability of ($P_{\mathcal{L},p,v,k}$) and hence via Lemma 4.2.5 to ($P_{\mathcal{L}}$). Our candidate will be problems which have a dynamic programming formulation. The model of dynamic programming we propose is based on the model of Martin, Rardin, and Campbell [12].

The essence of a dynamic programming solution to a problem is a decomposition of a solution to the program into optimal solutions to smaller subproblems. We will use a particular type of hypergraph to describe the structure of dependencies of a problem on its subproblems.

To begin we will need to introduce some concepts. A *directed hypergraph* $H = (V, E)$ is an ordered pair, where V is a finite set referred to as the *vertices* or *nodes* of the hypergraph, and E is a finite set where each element is of the form (v, S) where $S \subseteq V$ and $v \in V \backslash S$. We refer to the elements of E as *edges* or *arcs* of H. For an arc $e = (v, S) \in E$ we call v the *tail* of e and say e is *outgoing* from v. We call S the *heads* of e, call each $u \in S$ a *head* of e, and say e is *incoming* on each $u \in S$. We call vertices with no incoming arcs *sources* and we call vertices with no outgoing arcs *sinks*. For a directed hypergraph H, the set $L(H)$ denotes the set of sinks of H.

For any non-empty strict subset of vertices $U \subset V$, we define the *cut* induced by U, denoted $\delta(U)$, as follows

$$\delta(U) := \{(v, S) \in E : v \in U \text{ and } S \cap (V \backslash U) \neq \varnothing\}.$$

We say a directed hypergraph is *connected* if it has no empty cuts.

A *directed hyperpath* is a directed hypergraph P satisfying the following:

- there is a unique vertex $s \in V(P)$ identified as the *start* of P,

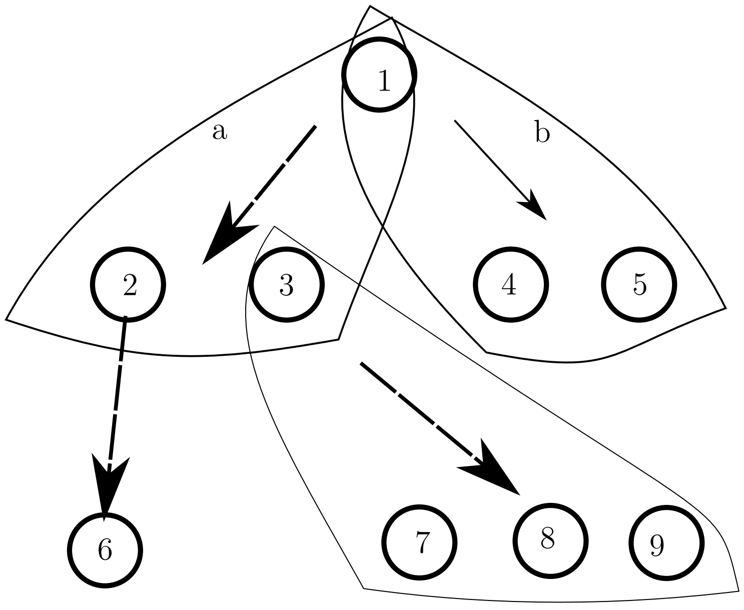

Figure 4.1: A directed acyclic hypergraph rooted at 1 with leaves $4, \ldots, 9$. Dashed arrows show a directed hyperpath.

- the start s is the tail of at most one arc of P, and the head of no arcs of H,
- every vertex in $V(P) \backslash \{s\}$ is the tail of precisely one arc of H,
- P is connected.

Observe that there is at least one, and potentially many, vertices of a path which have one incoming arc and no outgoing arcs. These vertices we call the *ends* of the path. If there is a path starting from a vertex u and ending with a vertex v then we say u is an *ancestor* to v and v is a *descendant* of u. For any vertex $v \in V(H)$, the subgraph of H *rooted at* v, denoted H_v, is the subgraph of H induced by the descendants of v (including v).

We say that a directed hypergraph $H = (V, E)$ is *acyclic* if there exists a topological ordering of the vertices of H. That is to say, there exists a bijection $t : V \to [|V|]$ such that for every $(v, S) \in E$, for each $u \in S$, $t(v) < t(u)$. Figure 4.1 depicts a directed acyclic hypergraph and one of its hyperpaths.

A common approach to dynamic programming involves a table of subproblems (containing information pertaining to their optimal solutions), and a recursive function describing how to compute an entry in the table based on the values of table entries which correspond to smaller subproblems. The values in the table are then determined in a bottom-up fashion. In our formal model, the entries in the table correspond to vertices of the hypergraph, and each hyperarc (v, S) describes a potential way of computing a feasible solution to the subproblem at v by composing the solutions to the subproblems at each node of S.

Consider a problem of the form (P). That is, we have a feasible region $\mathcal{X} \subseteq \mathbb{R}^n$ and an objective function $f : \mathcal{X} \to \mathbb{R}$ and we hope to maximize $f(x)$ subject to $x \in \mathcal{X}$. We need some language to describe how solutions to the dynamic program, i.e. paths in the directed hypergraph, will map back to solutions in the original problem space. To do this mapping back to the original space we will use an affine function. A function $g : \mathbb{R}^m \to \mathbb{R}^n$ is said to be *affine* if there exists a matrix $A \in \mathbb{R}^{n\times m}$ and a vector $b \in \mathbb{R}^n$ such that for any $x \in \mathbb{R}^m$, $g(x) = Ax + b$.

Oftentimes an affine function g will have a domain R^E indexed by a finite set E. When this happens for any $S \subseteq E$ we use $g(S)$ as a shorthand for $g(\chi(S))$ where $\chi(S)$ is the incidence vector of S. We further shorten $g(\{e\})$ to $g(e)$.

Let $H = (V, E)$ be a directed acyclic connected hypergraph with set of sources T. Let $\mathcal{P}(H)$ denote the set of paths in H which begin at a source in T and end only at sinks of H. Let $g : \mathbb{R}^E \to \mathbb{R}^n$ be an affine map which we will use to map between paths in $\mathcal{P}(H)$ and feasible solutions in $\mathcal{X}$. Let $c : \mathbb{R}^E \to \mathbb{R}$ be an affine function we will use as an objective function. We say (H, g, c) is a *dynamic programming formulation* for (P) if $g(\mathcal{P}(H)) = \mathcal{X}$, and moreover for any $x \in \mathcal{X}$,

$$f(x) = \max_{P\in g^{-1}(x)} c(P).$$

In other words, the optimal values of

$$\max\{c(P) : P \in \mathcal{P}(H)\} \tag{DP}$$

and (P) are equal, and the feasible region of (P) is the image (under g) of the feasible region of (DP). The *size* of a dynamic programming formulation is the number of arcs in $E(H)$.

In [12] the authors show that (DP) has a totally dual integral extended formulation of polynomial size. Thus they show that (DP) can be solved in polynomial time via linear programming. They further show that the extreme point optimal solution of this extended formulation lies in $\{0, 1\}^E$ under the following *reference subsets* condition: there exists a ground set I, and nonempty subsets $I_v \subseteq I$ for each vertex $v \in V(H)$ satisfying

1. $I_j \subseteq I_\ell$ for all $(\ell, J) \in H$ such that $j \in J$
2. and $I_j \cap I_{j'} = \varnothing$ for all $(\ell, J) \in E(H)$ such that $j, j' \in J$ with $j \neq j'$.

This condition is equivalent to the following *no common descendants* condition: for each $(\ell, J) \in E(H)$ for all $u \neq v \in J$, there does not exist $w \in V(H)$ such that w is a descendant of both u and v.

Lemma 4.3.1 *For any directed acyclic hypergraph $H = (V, E)$ the reference subsets condition is equivalent to the "no common descendants" condition defined above.*

Proof: First suppose that H does not satisfy the *no common descendants* condition. Then there exists $(\ell, J) \in E$ such that there exist $u, v \in J$ and $w \in V$ such w is a descendant of u and of v. Suppose for a contradiction that H has a *reference subset* system with ground set I.

We claim that $I_w \subseteq I_v$. The proof of this claim will symmetrically show that $I_w \subseteq I_u$. Then $I_w \cap I_v \supset I_w \neq \varnothing$ violating the second property of a reference subset system.

To prove the claim we will prove something stronger. In particular we will show that for any $x, y \in V$ such that y is a descendant of x, we have that $I_y \subseteq I_x$. Suppose not. Choose a counterexample x, y with path P starting at x and ending at y so that the number of edges in P is minimal. Clearly $|E(P)| \neq 0$ as otherwise $x = y$. Now, from the definition of P there exists an arc $(x, J) \in E(P)$ and there exists $z \in J$ such that there is a subgraph of P, denoted P', such that P' is a path starting at z and ending at y. By minimality, $I_y \subseteq I_z$. By the first property of reference subset systems, $I_z \subseteq I_x$. Thus $I_y \subseteq I_x$ contradicting that x, y and P form a counterexample.

Now for the other direction of the equivalence suppose that H satisfies the *no common descendants* condition. We will construct a reference subset system for H as follows. Let $I = V$ and for each $v \in V$ let I_v be the set of descendants of v. Then I satisfies the first property of a reference subset system since the descendant relation is transitive. Further, the *no common descendants* condition implies that I satisfies the second property of a reference subset system. Lastly, no I_v is empty since every vertex is their own descendant.

$\blacksquare$

We say that a dynamic programming formulation (H, g, c) of a problem (P) is integral if H satisfies the no common descendants condition. By the preceding discussion we have the following lemma

Lemma 4.3.2 *If a problem (P) has an integral dynamic programming formulation (H, g, c) then (P) can be solved in time polynomial in the encoding of (H, g, c).*

4.3.1 Congruency Constrained Dynamic Programming

In this subsection our goal is to show that when a problem of the form (P) has a dynamic programming formulation, then its congruency constrained version $(P_{\mathcal{L},p,v,k})$ has a dynamic programming formulation that is only a $O(p^3)$ factor larger than the formulation for the original problem. This will prove Theorem 4.3.7.

We begin with a handy lemma for constructing dynamic programming formulations of combinatorial optimization problems. This lemma allows us to manipulate the dynamic programming formulation for a combinatorial optimization problem to obtain a new dynamic programming formulation.

Lemma 4.3.3 *If (H, g, c) is a dynamic programming formulation for (P) and (H', g', c') is a dynamic programming formulation for (DP) with respect to hypergraph H and costs c then $(H', g \circ g', c')$ is a dynamic programming formulation for (P).*

Proof: The function $g \circ g'$ is map between $\mathcal{P}(H')$ and $\mathcal{X}$, and moreover it is affine since both g and g' are affine. Furthermore,

$$g \circ g'(\mathcal{P}(H')) = g(\mathcal{P}(H)) = \mathcal{X}.$$

Finally, for any $P \in \mathcal{P}(H')$,

$$f(g \circ g'(P)) = c(g'(P)) = c'(P).$$

■

Consider a directed hypergraph $H = (V, E)$ and an edge $(u, S) \in E$. For $v \in S$ we define the *hypergraph obtained from the subdivision of* (u, S) *with respect to* v to be the hypergraph $H' = (V', E')$ where $V' = V \dot{\cup} \{b_v\}$ for a new dummy vertex b_v and

$$E' = (E \backslash \{(u, S)\}) \cup \{(u, \{v, b_v\}), (b_v, S \backslash \{v\})\}.$$

That is, H' is obtained from H by replacing edge (u, S) with two edges: $(u, \{v, b_v\})$ and $(b_v, S \backslash \{v\})$. We call the edges $(u, \{v, b_v\})$ and $(b_v, S \backslash \{v\})$ the *subdivision* of edge (u, S).

Lemma 4.3.4 *Let $H = (V, E)$ be a directed acyclic hypergraph and let $H' = (V', E')$ be the directed acyclic hypergraph obtained via a subdivision of $(u, S) \in E$ with respect to $v \in S$. Then there is an affine function $g : \mathbb{R}^{E'} \to \mathbb{R}^E$, such that for any affine function $c : \mathbb{R}^E \to \mathbb{R}$, there exists an affine function $c' : \mathbb{R}^{E'} \to \mathbb{R}$ such that (H, g, c') is a dynamic programming formulation of the problem (DP) on H with objective c.*

Moreover if H satisfies the "no common descendants" property, this dynamic programming formulation is integral.

Proof: Take g to be the affine function defined as follows. For any $e \in E$ and $x \in \mathbb{R}^{E'}$,

$$g(x)_e := \begin{cases} x_{(u,\{v,b_v\})}, & \text{if } e = (u, S) \\ x_e, & \text{otherwise.} \end{cases}$$

It is not hard to see that g is affine, and in fact linear. In fact, g is simply the function which identifies e with $(u, \{v, b_v\})$ and every other edge in e with the corresponding edge in E'.

To see that g is a bijection between $\mathcal{P}(H)$ and $\mathcal{P}(H')$ it suffices to observe that a path P in $\mathcal{P}(H')$ uses $(u, \{v, b_v\})$ if and only P uses $(b_v, S\backslash\{v\})$. This follows immediately from the definition of hyperpaths.

Now for any $c : \mathbb{R}^E \to \mathbb{R}$ let $c' : \mathbb{R}^{E'} \to \mathbb{R}$ be the unique affine function which acts on the standard basis in the following way: for any $e \in E'$,

$$c'(e) = \begin{cases} c((u,S)), & \text{if } e = (u, \{v, b_v\}) \\ 0, & \text{if } e = (b_v, S\backslash\{v\}) \\ c(e), & \text{otherwise.} \end{cases}$$

Again since a path P in $\mathcal{P}(H')$ uses $(u, \{v, b_v\})$ if and only P uses $(b_v, S\backslash\{v\})$, it is easy to see that $c(g(P) = c'(P)$ for any $P \in \mathcal{P}(H')$. Hence (H', g, c') is the desired dynamic programming formulation.

Since the subdivision operation preserves the "no common descendants" property, (H', g, c) is integral if H satisfies the no common descendants condition. ■

For a directed hypergraph $H = (V, E)$ let $\Delta(H) := \max\{|S| : (u, S) \in E\}$ and let $\Gamma(H) := |\{(u, S) \in E : |S| = \Delta(H)\}|$. The following Lemma shows that we may assume the number of heads of any arc in a dynamic programming formulation is constant.

Lemma 4.3.5 *Consider a combinatorial optimization problem of the form (P). If there exists a dynamic programming formulation (H, g, c) for (P) then there exists a dynamic programming formulation (H^*, g^*, c^*) for (P) such that $\Delta(H^*) \leq 2$, and*

$$|E(H^*)| = \sum_{u \in V(H)} \sum_{(u,S) \in E(H)} (|S| - 1).$$

Moreover, if H is integral then H^* is integral.

Proof: We proceed by double induction on $\Delta(H)$ and $\Gamma(H)$. In the base case, for any $\Gamma(H)$ if $\Delta(H) \leq 2$ then (H, g, c) is the desired dynamic program. In the inductive case consider H such that $\Delta(H) > 2$ and suppose the lemma holds on all directed acyclic hypergraphs H' with $\Delta(H') < \Delta(H)$ or $\Delta(H') = \Delta(H)$ and $\Gamma(H') < \Gamma(H)$. Since $\Delta(H) \geq 3$ there exists an edge $(u, S) \in E(H)$ such that $|S| \geq 3$. Let (H', g', c') be the dynamic program for (DP) on H with objective c given by Lemma 4.3.4 where H' is obtained by a subdivision of (u, S) with respect to vertex $v \in S$. By Lemma 4.3.3 $(H', g \circ g', c')$ is a dynamic programming formulation for (P).

Now notice that either $\Delta(H') = \Delta(H) - 1$ or $\Delta(H') = \Delta(H)$ and $\Gamma(H') = \Gamma(H) - 1$. Hence by induction there is a dynamic program (H^*, g^*, c^*) for the problem (DP) on hypergraph H'

with respect to objective c' such that $\Delta(H^*) \leq 2$ and

$$\begin{aligned}
|E(H^*)| &= \sum_{u' \in V(H')} \sum_{(u',S') \in E(H')} (|S'| - 1) \\
&= |S \backslash \{v\}| - 1 + |\{v, b_v\}| - 1 + \sum_{u' \in V(H)} \sum_{(u',S') \in E(H) \backslash \{(u,S)\}} |S'| - 1 \\
&= \sum_{u' \in V(H)} \sum_{(u',S') \in E(H)} |S'| - 1,
\end{aligned}$$

where the second equality follows since H' is obtained from H via a subdivision of (u, S) with respect to v. By Lemma 4.3.3 $(H^*, g \circ g' \circ g^*, c^*)$ is a dynamic programming formulation for (P) as desired.

Since subdivision preserves the "no common descendants" property, if H is integral then H^* is integral.

■

The next lemma is our main techincal lemma. It provides the backbone of our dynamic programming formulation for $(P_{\mathcal{L},p,v,k})$ by showing that we can track the congruency of all hyperpaths rooted at a particular vertex by expanding the size of our hypergraph by a factor of $p^{\Delta(H)} + 1$.

Lemma 4.3.6 *Let $H = (V, E)$ be a directed acyclic hypergraph. Let p be a prime. Let $k \in \mathbb{Z}_p$ and let $a \in \mathbb{Z}_p^E$. There exists a directed acyclic hypergraph $H' = (V', E')$ and an affine function $g' : \mathcal{P}(H') \to \mathcal{P}(H)$, $g'(x) = Ax + b$, such that:*

1) $|E'| \leq p^{\Delta(H)+1}|E|$

2) For every $v \in V \backslash L(H)$, for every $k \in \mathbb{Z}_p$, if $\{P \in \mathcal{P}(H_v) : a(P) = k \mod p\} \neq \varnothing$ then there exists $v' \in V(H')$ such that

$$g'(\mathcal{P}(H'_{v'})) = \{P \in \mathcal{P}(H_v) : a(P) = k \mod p\}.$$

Moreover if H satisfies the "no common descendants" property then H' satisfies the "no common descendants" property.

Proof: We may assume that each source u of H has at most one arc outgoing from u. To see this, if the arcs leaving u are $(u, S_1), \ldots, (u, S_\ell)$ then we can replace u with ℓ vertices $u_1, \ldots, u_\ell$ and replace the arcs of u with (u_1, S_l),..., (u_ℓ, S_ℓ). This does not increase the number of arcs, and preserves the structure of the paths in $\mathcal{P}(H)$.

Now suppose for a contradiction that the lemma is false. Consider a counterexample $H = (V, E)$ which minimizes $|E|$. Then $E \neq \varnothing$, otherwise H is trivially not a counterexample. Let $u \in V$ be a source of H which has an outgoing arc, let $(u, S) \in E$ be the arc outgoing from u and let $d = |S|$. Arbitrarily fix an indexing on the vertices of S as in $S = \{s_1, \ldots, s_d\}$.

Since $H - u$ has fewer arcs than H, it is not a counterexample to the lemma. Hence there exists a directed acyclic hypergraph $\overline{H}$ and an affine function $\overline{g}$ satisfying properties 1) and 2) of the lemma with respect to $H - u$.

We now construct a new directed acylic hypergraph H' and affine function $g' : \mathbb{R}^{E'} \to \mathbb{R}^{E}$ via the following procedure.

1. Initialize $H' = (V', E')$ to be the hypergraph $\bar{H}$.
2. For every $k \in \mathbb{Z}_p$ such that $\{P \in \mathcal{P}(H) : a(P) = k \mod p\} \neq \varnothing$ do:
   (a) Add a new vertex u^k to V'.
   (b) For each $q \in \mathbb{Z}_p^d$ such that $a^T q = k - g((u, S)) \mod p$ do:
      - if for every $i \in [d]$ there exists $s'_i \in V'$ such that g is a bijection between $\mathcal{P}(H'_{s'_i})$ and $\{P \in \mathcal{P}(H_{s_i}) : a(P) = k - g((u, S)) \mod p\}$ then add arc $(u^k, \bigcup_{i \in [d]} s'_i)$ to E'.
3. We will define g' by its action on the standard basis. For each $e \in E'$ we define

$$g'(e) := \begin{cases} \chi((u, S)), & \text{if } e \text{ is outgoing from } u^k \text{ for some } k \in \mathbb{Z}_p \\ \overline{g}(e), & \text{otherwise.} \end{cases}$$

We will first verify that property 1) is satisfied by H'. Since $\overline{H}$ satisfies property 1), $|E(\overline{H})| \leq p^{\Delta(H-v)+1}(|E| - 1)$. By our construction of H', E' has at most $p \cdot |Z_p^d|$ more arcs than $E(\overline{H})$, and hence

$$|E'| \leq |\overline{E}| + p \cdot |Z_p^d| \leq p^{\Delta(H-v)+1}(|E| - 1) + p^{d+1} \leq p^{\Delta(H)+1}|E|.$$

Now we will show that property 2) is satisfied by H' and g'. Let $v \in V \backslash L(H)$. Let $k \in \mathbb{Z}_p$ and suppose that $\{P \in \mathcal{P}(H_v) : a(P) = k \mod p\} \neq \varnothing$. If $v \neq u$, then since $\overline{H}$ and $\overline{g}$ satisfies property 2), H' and g' satisfy property 2) by steps 1. and 3. of the construction. Thus we may assume that $v = u$. Then the vertex u^k was added to V' in step 2a) of the construction. It suffices to show that g' is a bijection between $\mathcal{P}(H'_{u^k})$ and $\{P \in \mathcal{P}(H_u) : a(P) = k \mod p\}$.

First we show surjectivity. Let $P \in \{P \in \mathcal{P}(H_u) : a(P) = k \mod p\}$. Then P is the disjoint union of (u, S) and d paths $P_1, \ldots, P_d$ where each P_i is in $\mathcal{P}(H_{s_i})$ respectively. For each $i \in [d]$, let $k_i = a(P_i) \mod p$. By property 2) applied to $\overline{H}$ and $\overline{g}$, and by our construction of H' starting from $\overline{H}$ and g' starting from $\overline{g}$, for each $i \in [d]$ there exists a vertex $s'_i \in V' \cap \overline{V}$ such that g' is a bijection between $\mathcal{P}(H'_{s'_i})$ and $\{P \in \mathcal{P}(H_{s_i}) : a(P) = k_i \mod p\}$. Since g' is such a bijection there exists $P'_i \in \mathcal{P}(H'_{s'_i})$ such that $g'(P'_i) = P_i$.

By step 2*b*) of our construction, there is an arc of the form $(u^k, \bigcup_{i\in[d]} s'_i)$ in E'. Hence there is a path $P' \in \mathcal{P}(H'_{u^k})$ which is a disjoint union of $(u^k, \bigcup_{i\in[d]} s'_i)$ with $P'_1, \dots, P'_d$. But then

$$g'(P') = g'(u^l, \bigcup_{i\in[d]} s'_i) \bigcup_{i\in[d]} g'(P'_d) = (u, S) \bigcup_{i\in[d]} P_i = P.$$

Now we show injectivity. Let $P', Q' \in \mathcal{P}(H'_{u^k})$ be paths in H' such that $g'(P') = g'(Q')$. Let $P = g'(P') = g'(Q')$. We consider the disjoint union of hyperpaths obtained by deleting u^k (and hence the single arc outgoing from u^k) from P' and Q'. Observe that

$$P - u = g'(P' - u) = g'(Q' - u).$$

By the bijectivity of g' inherited from $\overline{g}$, this implies that $P' - u = Q' - u$. But that necessarily implies that P' and Q' have the same arc outgoing from u^k. Thus $P' = Q'$. Therefore we have shown property 2).

It is easy to see that the construction preserves the no common descendants property.

■

We are now ready to show our main theorem, which says that for any combinatorial optimization problem which has a dynamic program, its congruency-constrained version also has a dynamic program of proportional size.

Theorem 4.3.7 *Consider an instance of a combinatorial optimization problem (P). Let p be a prime, let $v \in \mathbb{Z}_p^n$, and let $k \in \mathbb{Z}_p$. Consider the corresponding congruency-constrained optimization problem ($P_{\mathcal{L},p,v,k}$). If (P) has a dynamic programming formulation (H, g, c) then ($P_{\mathcal{L},p,v,k}$) has a dynamic programming formulation (H', g', c') such that $|E(H')| \le p^3 \cdot |V(H)| \cdot |E(H)|$.*

Moreover if (H, g, c) is integral then (H', g', c') is integral.

Proof: We first apply Lemma 4.3.5 to problem (*P*) and dynamic programming formulation (H, g, c) to obtain a dynamic programming formulation (H^*, g^*, c^*) for (*P*) such that

$$\Delta(H^*) \le 2 \quad \text{and} \quad |E(H^*)| \le |V(H)| \cdot |E(H)|.$$

Since g^* is an affine function, there exists a matrix A and a vector b such that $g^*(x) = Ax+b$ for any x. Now apply Lemma 4.3.6 to hypergraph H^* with prime p, integer $k - v^T b \in \mathbb{Z}_p$, and vector $A^T v \in \mathbb{Z}_p^E$. We obtain a directed acyclic hypergraph H' and affine function $g' : \mathcal{P}(H') \to \mathcal{P}(H^*)$ such that

1) $|E'| \leq p^{\Delta(H^*)+1}|E(H^*)| \leq p^3|V(H)||E(H)|$ and

2) For every $v \in V(H^*)\backslash L(H^*)$, if

$$\{P \in \mathcal{P}(H^*_v) : v^T A\chi(P) = k - v^T b \mod p\} \neq \varnothing$$

then there exists $v' \in V(H')$ such that g' is a bijection between $\mathcal{P}(H'_{v'})$ and $\{P \in \mathcal{P}(H^*_v) : v^T A\chi(P) = k - v^T b \mod p\}$.

Let $c' : \mathbb{R}^{E(H')} \to \mathbb{R}$ be the affine function $c^* \circ g'$. Let T be the set of sources in H' that satisfy the hypothesis in property 2). We claim that

$$D := (\bigcup_{u \in T} H'_u, g^* \circ g', c')$$

is a dynamic programming formulation for (P). Indeed by property 2),

$$g'(\mathcal{P}(\bigcup_{u \in T} H'_u)) = \{P \in: \mathcal{P}(H^*) : v^T A(P) = k - v^T b \mod p\}.$$

Now observe that

$$\begin{aligned} &\{P \in: \mathcal{P}(H^*) : v^T A(P) = k - v^T b \mod p\} \\ &= \{P \in: \mathcal{P}(H^*) : v^T (A(P) + b) = k \mod p\} \\ &= \{P \in: \mathcal{P}(H^*) : v^T g^*(P) = k \mod p\}. \end{aligned}$$

Since $g^*(\mathcal{P}(H^*)) = \mathcal{X}$, we have that

$$g^*(\{P \in: \mathcal{P}(H^*) : v^T g^*(P) = k \mod p\}) = \{x \in \mathcal{X} : v^T x = k \mod p\}.$$

Therefore

$$g^* \circ g'(\mathcal{P}(\bigcup_{u \in T} H'_u)) = \{x \in \mathcal{X} : v^T x = k \mod p\},$$

the feasible region of ($P_{\mathcal{L},p,v,k}$).

Lastly for any $P \in \mathcal{P}(\bigcup_{u \in T} H'_u)$, we have

$$f(g^* \circ g'(P)) = c(g'(P)) = c'(P).$$

Thus D is a dynamic programming formulation for ($P_{\mathcal{L},p,v,k}$). Observe that the lemmas applied in the construction of D all preserve integrality. Hence if (H, g, c) is integral then D is integral. ■

Due to Theorem 4.3.7 and Lemma 4.3.2 we have the following Corollary.

Corollary 4.3.8 *If (P) has an integral dynamic programming formulation (H, g, c) then for any v, k, p problem $(P_{\mathcal{L},p,v,k})$ can be solved in time polynomial in size of H, the prime p, and the encoding of g, c, v, k,*

Via this Corollary, Lemma 4.2.3, and Lemma 4.2.5 we obtain our first main result: Theorem 4.0.1.

4.4 Applications

In this section we show some applications of Theorem 4.0.1 to computing the nucleolus of cooperative games. We will study weighted voting games, b-matching games on bipartite graphs where one side has all b-values 1, and b-matching games on graphs of bounded treewidth.

4.4.1 Weighted Voting Games

The first application is to Weighted Voting Games. In [63] a pseudopolynomial time algorithm for computing the nucleolus of Weighted Voting Games was given. We show how the same result can be obtained as a special case of Theorem 4.0.1. Recall that a weighted voting game (n, ν) has value function $\nu : 2^{[n]} \to \{0, 1\}$ determined by a vector $w \in \mathbb{Z}^n$ and $T \in Z$, such that for any $S \subseteq [n]$, $\nu(S) = 1$ if and only if $w(S) \geq T$.

We partition $2^{[n]}$ into two classes: $N_0 := \{S \subseteq [n] : w(S) < T\}$ and $N_1 := \{S \subseteq [n] : w(S) \geq T\}$. If we can design a dynamic programming formulation for the minimum excess coalition problem restricted to N_0: $\max\{-x(S) : w(S) \leq T - 1, S \subseteq [n]\}$ and a dynamic programming formulation for the minimum excess coalition problem restricted to N_1: $\max\{-x(S)+1 : w(S) \geq T, S \subseteq [n]\}$, then the dynamic programming formulation which takes the maximum of these two formulations will provide a dynamic programming formulation for the minimum excess coalition problem of the weighted voting game.

If we let $W[k, D]$ denote $\max\{-x(S) : w(S) \leq D, S \subseteq [k]\}$ then we can solve the minimum excess coalition problem restricted to N_0 by computing $W[n, T - 1]$ via the following recursive expression, which is essentially a dynamic program for Knapsack Cover,

$$W[k, D] = \begin{cases} \max\{W[k-1, D-w_k], W[k-1, D]\}, & \text{if } k > 1 \\ -x_1, & \text{if } k = 1 \text{ and } w_1 \leq D \\ -\infty, & \text{if } k = 1 \text{ and } w_1 > D. \end{cases}$$

It is not hard to construct a dynamic programming formulation (H_0, g_0, c_0) for the minimum excess coalition problem restricted to N_0 by following this recursive expression. The hypergraph

H_0 will in fact be a rooted tree (i.e. all heads will have size one), and H_0 will have $O(nT)$ vertices and arcs. Via a similar technique, a dynamic programming formulation (H_1, g_1, c_1) with $O(nT)$ arcs can be constructed for the minimum excess problem restricted to N_1. Then by taking the union these dynamic programming formulations, we obtain an integral dynamic programming formulation of size $O(nT)$. Therefore by Theorem 4.0.1 we obtain a short proof that

Theorem 4.4.1 *([63] [25]) The nucleolus of a weighted voting game can be computed in pseudopolynomial time.*

In the following subsections we will see how the added power of hyperarcs lets us solve the more complex problem of computing the nucleolus of b-matching games on graphs of bounded treewidth.

4.4.2 Star Matching

In [4] Bateni, Hajiaghayi, Immorlica, and Mahini show that the nucleolus of a simple weighted b-matching game on a bipartite graph can be computed in polynomial time in the case where the b-value of every vertex on one side of the bipartition has b-value 1. Their proof is quite technical and specialized to the problem. With access to the dynamic programming framework we can provide a dramatically simpler proof.

Theorem 4.4.2 *([4]) Let $G = (V, E)$ be a b-valued, w-weighted graph and (n, ν) the associated b-matching game. Suppose that G is bipartite with bipartition $A \dot\cup B = V$ and suppose that for every node $a \in A$ we have $b_a = 1$.*

Then the nucleolus of (n, ν) can be computed in polynomial time.

Note that since one side of the bipartition has b-values all 1 we need not make the distinction between simple and non-simple b-matching games.

Proof: By Theorem 4.0.1 it will suffice to provide a dynamic programming formulation for the minimum excess problem with respect to a particular characterization set. In the previous and subsequent application we use the trivial characterization $[n]$ but this problem will actually take advantage of a specific characterization set.

First we want to show that $\mathcal{C}(n, \nu)$ is non-empty. Consider dual to the fractional b-matching linear program:

$$\begin{aligned} \min \ & b^T y \\ \text{s.t. } & y_u + y_v \geq w(uv) && \forall uv \in E \\ & y \geq \mathbb{0}. \end{aligned}$$

Since G is bipartite, and $b_a = 1$ for all $a \in A$, the optimal value of this linear program is equal to $\nu(V)$ [73]. Let y^* be an optimal solution to the linear program above. Let $x^* \in \mathbb{R}^n$ be chosen such that $x^*_u = b_u y^*_u$. We claim that $x^* \in \mathcal{C}(n, \nu)$. Clearly x^* is efficient and individually rational. Let $S \subseteq V$ and let M be a maximum weight b-matching in $G[S]$. Then

$$x^*(S) \geq \sum_{uv \in M} x^*(uv) = \sum_{uv \in M} b_u y^*_u + b_v y^*_v \geq w(uv)$$

with the last inequality following since the restriction of y^* to S is dual feasible for the b-matching linear program on $G[S]$. Therefore the core of (n, ν) is non-empty.

Observe that every b-matching on G is a disjoint union of stars with their centers in B. Since the core of (n, ν) is non-empty, we can apply Corollary 3.3.3 to see that

$$\mathcal{S} = \{S \subseteq V : G[S] \text{ is a star with its center in } B\}$$

is a characterization set for (n, ν).

Given $x \in \mathbb{R}^n$ we now present a dynamic programming formulation for the minimum excess problem

$$\min\{x(S) - \nu(S) : S \in \mathcal{S}\}.$$

Suppose $S \in \mathcal{S}$. Then for some integer $k \leq n$, $G[S]$ is a k-star centered at some vertex $v \in B$ with edges to vertices $u_1, \ldots, u_k \in A$. The excess $\mathrm{ex}(x, S)$ of such a star decomposes into

$$x(S) - \nu(S) = x_v + \sum_{i=1}^{k} (x_{u_i} - w(vu_i)).$$

Finding the minimum excess of a star decomposes into the following decisions: choose the center of the star, choose how many edges the star will have, and then choose which edges will be in the star.

For a k-star with a fixed center at a vertex v, the problem of finding edges which minimize its excess can be seen as a max knapsack problem with items with vu for each edge $vu \in \delta(v)$ with cost $w(uv) - x_u$, and knapsack capacity k. In Section 4.4.1 we already designed a dynamic programming formulation for the maximum knapsack problem. Let $(H_{v,k}, g_{v,k}, c_{v,k})$ be the associated dynamic programming formulation.

Then we can construct a dynamic programming formulation for the minimum excess problem in our setting by constructing a hypergraph H rooted at r with arcs from r to the root of $H_{v,k}$ for each $v \in B$, and each $k \in [n] \cup \{0\}$. The cost of each such arc will be x_v, and taking such an arc will be mapped to adding v to our coalition (i.e. choosing v as the center of our star). Each dynamic program with hypergraph $H_{v,k}$ tells us which vertices in A are joining our coalition and the associated charge to the total excess.

Since our dynamic programmin formulation will be the disjoint union of $O(n^2)$ dynamic programs each with $O(n^2)$ arcs, by Theorem 4.0.1 this formulation implies that the nucleolus of (n, ν) can be computed in polynomial time. ■

4.4.3 Treewidth

Consider a graph $G = (V, E)$. We call a pair $(T, \mathcal{B})$ a *tree decomposition* [41] [69] of G if $T = (V_T, E_T)$ is a tree and $\mathcal{B} = \{B_i \subseteq V : i \in V_T\}$ is a collection of subsets of V, called *bags*, such that

1. $\bigcup_{i \in V_T} B_i = V$, i.e. every vertex is in some bag,
2. for each $v \in V$, the subgraph of T induced by $\{i \in V_T : v \in B_i\}$ is a tree, and
3. for each $uv \in E$, there exists $i \in V_T$ such that $u, v \in B_i$.

The *width* of a tree decomposition is the size of the largest bag minus one, i.e. $\max_{i \in V_T}\{|B_i| - 1\}$. The *treewidth* of graph G, denoted $\text{tw}(G)$, is minimum width of a tree decomposition of G. Figure 4.3 shows an example of the tree decomposition of a graph.

We may assume that tree decompositions of a graph have a special structure. We say a tree decompostion $(T, \mathcal{B})$ of G is *nice* if there exists a vertex $r \in V_T$ such that if we view T as a tree rooted at r then every vertex $i \in V_T$ is one of the following types:

- **Leaf:** i has no children and $|B_i| = 1$.
- **Introduce:** i has one child j and $B_i = B_j \dot\cup \{v\}$ for some vertex $v \in V$.
- **Forget:** i has one child j and $B_i \dot\cup \{v\} = B_j$ for some vertex $v \in V$.
- **Join:** i has two children j_1, j_2 with $B_i = B_{j_1} = B_{j_2}$.

It turns out that if a graph has a tree decomposition of width w then a nice tree decomposition of width w times the number of vertices of the graph can be computed in time polynomial in w and the number of vertices of the graph.

Theorem 4.4.3 *([48] Lemma 13.1.3) If $G = (V, E)$ has a tree decompostion of width w with n tree vertices then there exists a nice tree decomposition of G of width w and $O(|V|)$ tree vertices which can be computed in $O(|V|)$ time.*

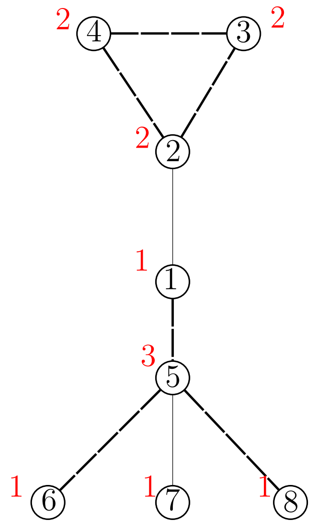

Figure 4.2: A b-matching game on a graph of bounded treewidth. Red numbers indicate b-values and dashed edges indicate a b-matching. Figure 4.3 shows the tree decomposition of this graph.

4.4.4 Dynamic Program for b-Matching Games

We want to show that on graphs of bounded treewidth, the nucleolus of b-matching games can be computed efficiently. Fix a graph $G = (V, E)$, a vector of b-values $b \in \mathbb{Z}^V$, and tree decomposition $(T, \mathcal{B})$ of treewidth w, where T is rooted at r, to be used throughout this section. For $i \in V(T)$ let T_i denote the subtree of T rooted at i, and also let $G_i := G[\bigcup_{j \in V(T_i)} B_j]$. For any $v \in V(G_i)$, let $\delta_i(v) := \{uv \in E(G_i)\}$. An example of such a graph is depicted in Figure 4.2.

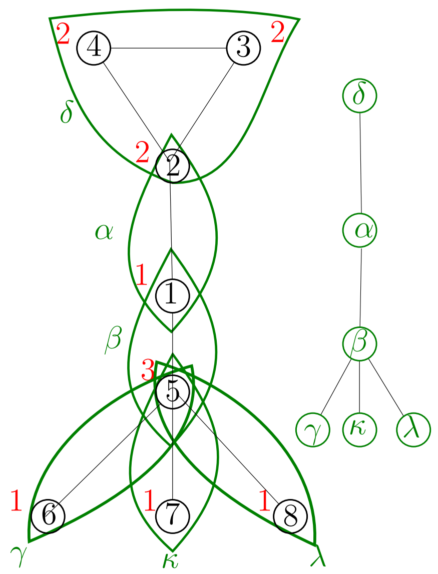

Figure 4.3: Tree decomposition of the graph in Figure 4.2. On the right is the tree from the decomposition, with each associated bag labelled on the left.

For any $i \in V(T)$, $X \subseteq B_i$, $d \in \{d \in \mathbb{Z}^{B_i} : 0 \leq d \leq \Delta(G)\}$, and $F \subseteq E(B_i)$, we define the combinatorial optimization problem C[i,X,d,F] to be the problem of finding a b-matching M and a set of vertices S such that M uses only edges of G_i, S uses only vertices of G_i, the intersection of M and $E(B_i)$ is F, the number of edges in M adjacent to u is d_u for each u in B_i, and the vertices in S not intersecting an edge in F is X. Formally C[i,X,d,F] is defined as follows

$$
\begin{aligned}
\max\ & w(M) - x(S) && && \text{(C[i,X,d,F])}\\
\text{s.t. } & |M \cap \delta_i(v)| \leq b_v && \forall v \in V\\
& |M \cap \delta_i(u)| \leq d_u && \forall u \in B_i\\
& d_u = 0 && \forall u \in X\\
& M \cap E(B_i) = F\\
& S = V(M) \dot{\cup} X\\
& X \subseteq B_i\\
& M \subseteq E(G_i).
\end{aligned}
$$

We define C[i] to be the union over all $C[i,X,d,F]$.

$$\begin{aligned} \max\ & w(M) - x(S) \\ \text{s.t. } & (M,S) \text{ is feasible for } C[i,X,d,F] \\ & \quad \text{for some } (X,d,F) \in B_i \times \mathbb{Z}^{B_i} \times E(G_i). \end{aligned} \tag{C[i]}$$

We will show a dynamic programming formulation (H,g,c) for $C[i]$. Since the feasible region of the minimum excess coalition problem for b-matching games is the image of the feasible region of $C[i]$ under the linear map which projects out M, and $\nu(S) - x(S) = \max_{(M,S)\text{feasible for } C[i]} w(M) - x(S)$, the existence of (H,g,c) will imply the existence a dynamic programming formulation of the minimum excess coalition problem for b-matching games of the same encoding length.

Lemma 4.4.4 *Let $i \in V(T)$. There exists an integral dynamic programming formulation (H,g,c) for C[i] such that*

1) *$|E(H)| \leq |V(T_i)| \cdot w \cdot \Delta(G)^w \cdot w^2$ and*

2) *For every $j \in V(T_i)$, $X \subseteq B_i$, $d \in \mathbb{Z}^{B_i}$, and $F \subseteq E(B_i)$, if C[i,X,d,F]) has a feasible solution then there exists $a \in V(H)$ such that (H_a,g,c) is an integral dynamic programming formulation for C[i,X,d,F]*

Proof: We proceed by induction on the distance from i to a leaf of T. If i is a leaf of T the Lemma is trivial. So we suppose that i is not a leaf of T and the Lemma holds for all vertices j of T which are closer to a leaf than i.

Now we proceed by case distinction on which class of node i is: an Introduce node, a Forget node, or a Join node.

Case: Introduce. If i is an Introduce node then i has one child j, and $B_i = B_j \dot\cup \{v\}$ for some vertex $v \in V$. Let (H',g',c') be the dynamic programming formulation for $C[j]$ guaranteed by the inductive hypothesis. We construct a new dynamic programming formulation (H,g,c) via the following procedure:

1. Initialize H to H'.

2. For each $e' \in E(H')$, define the action of g on e' to be $g(e') = g'(e')$.

3. For each $e' \in E(H')$, define the action of c on e' to be $c(e') = c'(e')$.

4. For every $X \subseteq B_i$, $d \in Z^{B_i}$, $F \subseteq E(B_i)$ such that $C[i,X,d,F]$ has a feasible solution do:

(a) Add a new vertex $a^{i,X,d,F}$ to $V(H)$

(b) Let $a^{j,X\cap B_j,d',F\cap E(B_j)}$ be the vertex of $V(H')$ guaranteed by property 2), such that

$$(H_{a^{j,X\cap B_j,d',F\cap E(B_j)}}, g, c)$$

is a dynamic programming formulation for $C[j, X\cap B_j, d', F\cap E(B_j)]$, where

$$d'_u := d_u - \begin{cases} 1, & \text{if } uv \in F \\ 0, & \text{otherwise.} \end{cases}$$

for all $u \in B_j$.

(c) Add arc $e = (a^{i,X,d,F}, \{a^{j,X\cap B_j,d',F\cap E(B_j)}\})$ to $E(H)$.

(d) Define action of g on e to be

$$g(e) = \begin{pmatrix} F\backslash E(B_j) \\ (X\backslash B_j) \cup (\{v\} \cap V(F)) \end{pmatrix} \in \{0,1\}^{E(G_i),V(G_i)}$$

(e) Define action of c on e to be

$$c(e) = w(F\backslash E(B_j)) - x(X\backslash B_j) - x(\{v\} \cap V(F)).$$

Property 1) is immediate from induction and the number of arcs added by the construction.

To verify property 2) consider $C[i', X, d, F]$ satisfying the hypothesis of property 2). If $i' \neq i$ property 2) holds immediately by induction. So suppose $i' = i$. We claim that $(H_{a^{i,X,d,F}}, g, c)$ is a dynamic programming formulation for $C[i, X, d, F]$.

Let

$$\begin{pmatrix} M & S \end{pmatrix}^T$$

be a feasible solution to $C[i, X, d, F]$. For any

$$\begin{pmatrix} M & S \end{pmatrix}^T$$

feasible for $C[i, X, d, F]$ there exists

$$\begin{pmatrix} M' & S' \end{pmatrix}^T$$

feasible for $C[j, X\cap B_j, d', F\cap E(B_j)]$ (where d' is as defined in Step 2b) of the construction) such that

$$M = M' \dot{\cup} (F \cap E(B_j)) \quad \text{and} \quad S = S' \dot{\cup} (X\backslash B_j) \cup (\{v\}) \cap V(F))) .$$

Moreover, for any

$$\begin{pmatrix} M' & S' \end{pmatrix}^T$$

feasible for $C[j, X \cap B_j, d', F \cap E(B_j)]$, we have that

$$\begin{pmatrix} M' \dot{\cup} (F \cap E(B_j)) \\ S' \dot{\cup} (X \backslash B_j) \cup (\{v\} \cap V(F)) \end{pmatrix}$$

is feasible for $C[i, X, d, F]$. Hence by Step 2d) and Step 2e) of the construction, and the inductive hypothesis, g and c behave as desired for the dynamic program $(H_{a^{i,X,d,F}}, g, c)$ to be a dynamic programming formulation of $C[i, X, d, F]$.

Case: Forget. If i is a Forget node then i has one child j, and $B_i \dot{\cup} \{v\} = B_j$ for some vertex $v \in V$. Let (H', g', c') be the dynamic programming formulation for $C[j]$ guaranteed by the inductive hypothesis. We construct a new dynamic programming formulation (H, g, c) via the following procedure:

1. Initialize H to H'.
2. For each $e' \in E(H')$, define the action of g on e' to be $g(e') = g'(e')$.
3. For each $e' \in E(H')$, define the action of c on e' to be $c(e') = c'(e')$.
4. For every $X \subseteq B_i, d \in Z^{B_i}, F \subseteq E(B_i)$ such that $C[i, X, d, F]$ has a feasible solution do:
   (a) Add a new vertex $a^{i,X,d,F}$ to $V(H)$.
   (b) For each $J \subseteq \delta(v) \cap E(B_j)$, $d' \in \mathbb{Z}^{B_j}$ such that $d'_u = d_u$ for all $u \in B_i$, and for each $Y \in \{X, X \cup \{v\}\}$, if $C[j, Y, d', F \cup J]$ has a feasible solution do:
      i. Let $a^{j,Y,d',F\cup J}$ be the vertex of $V(H')$ guaranteed by property 2), such that $(H_{a^{j,Y,d',F\cup J}}, g, c)$ is a dynamic programming formulation for $C[j, Y, d', F \cup J]$.
      ii. Add arc $e = (a^{i,X,d,F}, \{a^{j,Y,d',F\cup J}\})$ to $E(H)$.
      iii. Define action of g on e to be $g(e) = 0$.
      iv. Define action of c on e to be $c(e) = -w(J) + x(Y \cap \{v\})$.

Property 1) is immediate from induction and the number of arcs added by the construction.

To verify property 2) consider $C[i', X, d, F]$ satisfying the hypothesis of property 2). If $i' \neq i$ property 2) holds immediately by induction. So suppose $i' = i$. We claim that $(H_{a^{i,X,d,F}}, g, c)$ is a dynamic programming formulation for $C[i, X, d, F]$.

Let

$$\begin{pmatrix} M & S \end{pmatrix}^T$$

be feasible for $C[i, X, d, F]$. Then there exists

$$\begin{pmatrix} M' \\ S' \end{pmatrix} \in \{0,1\}^{E(G_j), V(G_j)}$$

such that $M' = M \dot\cup J$ for some $J \in \delta(v) \cap E(B_j)$ and $S' = S \cup Y$ for some $Y \in \{X, X \cup \{v\}\}$. Further there exists $d' \in \mathbb{Z}^{B_j}$ such that

$$|M' \cap \delta(u)| = d'_u \quad \text{for all } u \in B_j.$$

Hence (M', S') is feasible for $C[j, Y, d', F \cup J]$. Moreover for any

$$\begin{pmatrix} M' & S' \end{pmatrix}^T$$

feasible for $C[j, Y, d', F \cup J]$, we have that

$$\begin{pmatrix} M' \backslash J \\ (S \cup X) \backslash Y \end{pmatrix}$$

is feasible for $C[i, X, d, F]$. Thus by induction and Step 2*b*) of the construction, g and c behave as desired for $(H_{a^{i,X,d,F}}, g, c)$ to be a dynamic programming formulation of $C[i, X, d, F]$.

Case: Join. If i is a Join node then i has two children: j_1 and j_2, and $B_{j_1} = B_{j_2} = B_i$. Let (H^1, g^1, c^1) be the dynamic programming formulation for $C[j_1]$ guaranteed by the induction hypothesis, and let (H^2, g^2, c^2) be the similarly guaranteed dynamic programming formulation for $C[j_2]$ We construct a new dynamic programming formulation (H, g, c) via the following procedure:

1. Initialize $H = H^1 \dot\cup H^2$.
2. For each $e' \in E(H^1)$, define the action of g on e' to be $g(e') = g^1(e')$. Similarly for each $e' \in E(H^2)$, define the action of g on e' to be $g(e') = g^2(e')$.
3. For each $e' \in E(H^1)$, define the action of c on e' to be $c(e') = c^1(e')$. Similarly for each $e' \in E(H^2)$, define the action of c on e' to be $c(e') = c^2(e')$.
4. For every $X \subseteq B_i$, $d \in \mathbb{Z}^{B_i}$, $F \subseteq E(B_i)$ such that $C[i, X, d, F]$ has a feasible do:
   (a) Add a new vertex $a^{i,X,d,F}$ to $V(H)$.
   (b) For every $d^1 \in \mathbb{Z}^{B_{j_1}}$ such that $C[j_1, X, d^1, F]$ has a feasible solution, and for every $d^2 \in \mathbb{Z}^{B_{j_2}}$ such that $C[j_2, X, d^2, F]$ has a feasible solution, if $d^1 + d^2 - |F \cap \delta(u)| = d_u$ for all $u \in B_i$ then do:
      i. Let $a^{j_1,X,d^1,F}$ be the vertex of $V(H^1)$ guaranteed by property 2), such that $(H_{a^{j_1,X,d^1,F}}, g, c)$ is a dynamic programming formulation for $C[j_1, X, d^1, F]$.
      ii. Let $a^{j_2,X,d^2,F}$ be the vertex of $V(H^2)$ guaranteed by property 2), such that $(H_{a^{j_2,X,d^2,F}}, g, c)$ is a dynamic programming formulation for $C[j_2, X, d^2, F]$.
      iii. Add arc $e = (a^{i,X,d,F}, \{a^{j_1,X,d^1,F}, a^{j_2,X,d^2,F}\})$ to $E(H)$.

iv. Define action of g on e to be $g(e) = 0$.

v. Define action of c on e to be

$$c(e) = -w(F) + x(\{u \in B_i : d^1_u > 0 \text{ and } d^2_u > 0\}).$$

Property 1) is immediate from induction and the number of arcs added by the construction.

To verify property 2) consider $C[i', X, d, F]$ satisfying the hypothesis of property 2). If $i' \neq i$ property 2) holds immediately by induction. So suppose $i' = i$. We claim that $(H_{a^{i,X,d,F}}, g, c)$ is a dynamic programming formulation for $C[i, X, d, F]$.

Let

$$\begin{pmatrix} M & S \end{pmatrix}^T$$

be feasible for the problem $C[i, X, d, F]$. Then there exists

$$\begin{pmatrix} M_1 & S_1 \end{pmatrix}^T$$

feasible for the problem $C[j_1, X, d^1, F]$ and

$$\begin{pmatrix} M_2 & S_2 \end{pmatrix}^T$$

feasible for the probelm $C[j_2, X, d^2, F]$ such that

$$d^1 + d^2 - |F \cap \delta(u)| = d_u$$

for all $u \in B_i$, satisfying that

$$M = M_1 \cup M_2 \quad \text{and} \quad S = S_1 \cup S_2.$$

Moreover for any

$$\begin{pmatrix} M_1 & S_1 \end{pmatrix}^T$$

feasible for $C[j_1, X, d^1, F]$ and for any

$$\begin{pmatrix} M_2 & S_2 \end{pmatrix}^T$$

feasible for $C[j_2, X, d^2, F]$ such that $d^1 + d^2 - |F \cap \delta(u)| = d_u$ for all $u \in B_i$, we have that

$$\begin{pmatrix} M_1 \cup M_2 \\ S_1 \cup S_2 \end{pmatrix}$$

is feasible for $C[i, X, d, F]$. Hence $g(\mathcal{P}\left(H_{a^{i,X,d,F}}\right))$ is equal to the feasible region $C[i, X, d, F]$ by Steps $2b)ii - 2b)iv$ of the construction. Furthermore

$$\begin{aligned} w(M) - x(S) &= w(M_1) + w(M_2) - w(M_1 \cap M_2) - x(S_1) - x(S_2) + x(S_1 \cap S_2) \\ &= w(M_1) - x(S_1) + w(M_2) - x(S_2) - w(F) \\ &+ x(\{u \in B_i : d^1_u > 0 \text{ and } d^2_u > 0\}) \end{aligned}$$

with the second equality following since there are no edges from a vertex in $V(G_{j_1})\backslash B_i$ to a vertex in $V(G_{j_2})\backslash B_i$ by the properties of tree decompositions. Hence c behaves as desired for $(H_{a^{i,X,d,F]}}, g, c)$ to be a dynamic programming formulation of $C[i, X, d, F]$.

It is not hard to see that the constructions in each case preserve integrality. ■

By the preceding discussion, Lemma 4.4.4, and Theorem 4.0.1 we have shown Theorem 4.0.2.

Chapter 5

Conclusion

In Chapter 2 we gave an algorithm for computing the nucleolus of any weighted cooperative matching game in polynomial time (Theorem 2.0.1). To achieve this we gave a compact formulation of each linear program in the MPS Scheme (Theorem 2.3.1) when the core of the underlying game is empty. This compact formulation required us to first obtain a universal allocation in the relative interior of the first linear program of the MPS Scheme (P_1). Finding such an allocation required the use of the ellipsoid method or similar method of solving linear programs using a separation oracle. While the ellipsoid method runs in polynomial time in this case, the hidden constants can make the technique prohibitive in practical settings.

When the core is non-empty allocations in (P_1) are in bijection with solutions to the dual of the fractional matching problem [20]. Using this fact allows for the design of algorithms for the nucleolus of matching games with non-empty core which are more combinatorial in nature and with faster runtimes, see for instance [7]. It would be interesting to see if there is a combinatorial algorithm for computing the nucleolus of weighted matching games with empty core. The thesis of John Hardwick [42] provides a graphical algorithm for computing the nucleolus of unweighted matching games. Such work may provide an interesting starting point.

In a similar vein, it is an interesting open question to find a characterization of the the allocations in the first linear program of the MPS Scheme for matching games with empty core in terms of dual solutions to an associated weighted matching problem. If that is not possible, maybe characterizing an interesting subset of such allocations, such as in the intersection with the kernel?

Open Question 5.0.1 *Is there a subset of allocations in the leastcore of a matching game with empty core which admit a characterization in terms of the dual of some weighted matching problem?*

Open Question 5.0.2 *Is there a combinatorial algorithm for computing the nucleolus of a weighted matching game with empty core in polynomial time?*

Open Question 5.0.3 *What is the optimal runtime of an algorithm which computes the nucleolus of a weighted matching game with empty core?*

In Chapter 3 we turned our attention from matching games to b-matching games. In this setting we generalized work on the hardness of computing the nucleolus. In Theorem 3.0.1 we show that deciding whether allocation is equal to the nucleolus of a simple b-matching game is NP-hard on bipartite graphs. This holds even when the underlying graph is bipartite, the graph is unweighted, the maximum degree is 7, and every vertex has b-value 3.

When $b_v \leq 2$ for each $v \in V$, we addressed some basic cases with polynomial nucleolus computation algorithms. We also showed that even when the core is empty, the leastcore constraints can be separated in polynomial time (Theorem 3.2.3). The final gap is computing the nucleolus of simple b-matching games in general. This is open even when $b_v \leq 2$ for all $v \in V$, the core is non-empty, and the graph is unweighted.

Open Question 5.0.4 *Is there an algorithm which computes the nucleolus of a b-matching game where $b_v \leq 2$ for all $v \in V$ in polynomial time? How about in the special case where the graph is bipartite and the edges are unweighted? Alternatively, is this problem* NP*-hard?*

In Chapter 4 we provided a general framework for computing the nucleolus of a cooperative matching game by proving that when you can characterize the min excess problem of the game as a dynamic programming problem you can compute the nucleolus with only a polynomial factor expansion of the time cost of solving the dynamic program (Theorem 4.0.1). We applied this result to b-matching games on graphs of bounded treewidth (Theorem 4.0.2). It would be interesting to explore what other potential applications this technique has and if it can be generalized further.

Chapter 4 also contained a discussion of weighted voting games, the work on which was a major inspiration for our work on dynamic programming techniques for computing the nucleolus. An open problem which arose from this discussion relates to analyzing Kopelowitz Scheme on weighted voting games. The question is: how many rounds does Kopelowitz Scheme need to find the nucleolus of a weighted voting game? Can we prove it only needs a polynomial number of rounds, or on the other hand can we exhibit an example that has a superpolynomial lower bound on the number of rounds?

Open Question 5.0.5 *Can the number of rounds needed by Kopelowitz Scheme to compute the nucleolus of a weighted voting game be bounded by a polynomial?*

Much of the theoretical discussion around computing the nucleolus centres around techniques based on hierarchies of linear programs, such as Kopelowitz Scheme, the MPS Scheme, or even the relaxed MPS Scheme. This is not necessarily the only way forward though. For related solution concepts there are other polynomial-time techniques. For instance Faigle, Kern, and Kuipers [30] show how a sequence of local transfers can obtain an allocation in the leastcore intersect prekernel of a cooperative game. The nucleolus is contained in this set, and hence when this set is a singleton their algorithm computes the nucleolus.

In particular, the core intersect prekernel is unique for convex games [60]. Characterizing the convexity of generalizations of matching games including hypergraph matching games [56] and b-matching games [55] has been studied. What is interesting though is that convexity is only a sufficient condition for core intersect prekernel being unique. It would be interesting to explore other direct characterizations of matching games and their relatives which have a unique core intersect prekernel.

Open Question 5.0.6 *Is there a characterization of b-matching games which have a unique leastcore intersect prekernel which can be verified in polynomial time?*

An other open direction based on this line discussion is designing local transfer schemes for the nucleolus directly. Is there a polynomial time, or even finite, algorithm in the vein of the Faigle, Kern, Kuipers (FKK) algorithm which computes the nucleolus from some starting allocations by a sequence of transfer of value between coalitions?

To conclude, allow us to discuss what we think is the major open question in this area: what is the connection between separating the leastcore constraints and computing the nucleolus? To the best of our knowledge, for every major class of cooperative combinatorial optimization games where the leastcore constraints are efficiently separable, the nucleolus is also efficiently computable. Further for every major class of cooperative combinatorial optimization games where computing the nucleolus is NP-hard, separating the leastcore constraints also is NP-hard.

Open Question 5.0.7 *Can one characterize the combinatorial optimization games for which being able to efficiently compute the minimum excess coalition with respect to any given allocation implies being able to compute the nucleolus efficiently?*

For the leastcore intersect prekernel the FKK algorithm explicitly shows that efficient separation of the leastcore implies efficient computation of a point in the leastcore intersect prekernel. Is this same relationship true for the nucleolus of a cooperative game? Our work in Chapter 4 shows it is does in the special case where leastcore separation takes the form of a dynamic program, and the FKK algorithm shows it does in the special case where the leastcore intersect prekernel is unique.

Our work in the other chapters of this thesis can be thought of as unified by this belief in a relationship between leastcore separation and nucleolus computation. We generalized the hardness result in [6] to bipartite graphs because of intuition derived from the core separation problem being proven to be hard in that setting. Even deeper, the compact formulation for the linear programs in the MPS Scheme we present in Chapter 2 centres around the excess assigned to a universal allocation in first the linear program. Moreover the constraints in the compact formulation are derived by using the way in which the dual to the leastcore separation problem decomposes the underlying graph.

References

[1] Neda Akbari, Mohammad Hossein Niksokhan, and Mojtaba Ardestani. Optimization of water allocation using cooperative game theory (Case study: Zayandehrud basin). *Journal of Environmental Studies*, 2015.

[2] Robert J Aumann and Michael Maschler. Game theoretic analysis of a bankruptcy problem from the talmud. *Journal of economic theory*, 36(2):195–213, 1985.

[3] Haris Aziz and Bart de Keijzer. Shapley meets shapley. In *31st International Symposium on Theoretical Aspects of Computer Science*, page 99, 2014.

[4] 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.

[5] J. M. Bilbao, J. R. Fernández, N. Jiménez, and J. J. López. Voting power in the European Union enlargement. *European Journal of Operational Research*, 2002.

[6] Péter Biró, Walter Kern, Dömötör Pálvölgyi, and Daniel Paulusma. Generalized matching games for international kidney exchange. 2019.

[7] Péter Biró, Walter Kern, and Daniël Paulusma. Computing solutions for matching games. *Int J Game Theory*, 41:75–90, 2012.

[8] Péter Biró, Walter Kern, Daniël Paulusma, and Péter Wojuteczky. The stable fixtures problem with payments. *Games and Economic Behavior*, 11(9):241–24, feb 2017.

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

[10] Rodica Brânzei, Elena Iñarra, Stef Tijs, and José M. Zarzuelo. A simple algorithm for the nucleolus of airport profit games. *International Journal of Game Theory*, 2006.

[11] Rodica Brânzei, Tamás Solymosi, and Stef Tijs. Strongly essential coalitions and the nucleolus of peer group games. *International Journal of Game Theory*, 33(3):447–460, 2005.

[12] Brian A Campbell, R Kipp Martin, Ronald L Rardin, and Brian A Campbell. Polyhedral characterization of discrete dynamic programming. *Operations Research*, 38(1):127–138, 1990.

[13] 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.

[14] Karthekeyan Chandrasekaran. Graph stabilization: A survey. In *Combinatorial Optimization and Graph Algorithms*, pages 21–41. Springer, 2017.

[15] Ning Chen, Pinyan Lu, and Hongyang Zhang. Computing the nucleolus of matching, cover and clique games. In *AAAI*, 2012.

[16] Karen S Cook and Toshio Yamagishi. Power in exchange networks: A power-dependence formulation. *Social networks*, 14(3-4):245–265, 1992.

[17] Morton Davis and Michael Maschler. The kernel of a cooperative game. *Naval Research Logistics Quarterly*, 12(3):223–259, September 1965.

[18] Xiaotie Deng and Qizhi Fang. Algorithmic cooperative game theory. In *Pareto Optimality, Game Theory And Equilibria*, pages 159–185. Springer, 2008.

[19] Xiaotie Deng, Qizhi Fang, and Xiaoxun Sun. Finding nucleolus of flow game. *Journal of Combinatorial Optimization*, 2009.

[20] Xiaotie Deng, Toshihide Ibaraki, and Hiroshi Nagamochi. Algorithmic aspects of the core of combinatorial optimization games. *Mathematics of Operations Research*, 24(3):751–766, 1999.

[21] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005.

[22] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of Research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

[23] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17(3):449–467, 1965.

[24] Jack Edmonds. Systems of distinct representatives and linear algebra. *Journal of Research of the National Bureau of Standards Section B Mathematics and Mathematical Physics*, 1967.

[25] Edith Elkind, Leslie Ann Goldberg, Paul W Goldberg, and Michael Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

[26] Edith Elkind and Dmitrii Pasechnik. Computing the nucleolus of weighted voting games. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 327–335. SIAM, 2009.

[27] Kimmo Eriksson and Johan Karlander. Stable outcomes of the roommate game with transferable utility. *International Journal of Game Theory*, 29(4):555–569, 2001.

[28] Ulrich Faigle, Walter Kern, Sándor P Fekete, and Winfried Hochstättler. The nucleon of cooperative games and an algorithm for matching games. *Mathematical Programming*, 83(1-3):195–211, 1998.

[29] Ulrich Faigle, Walter Kern, and Jeroen Kuipers. Note computing the nucleolus of min-cost spanning tree games is np-hard. *International Journal of Game Theory*, 27(3):443–450, 1998.

[30] Ulrich Faigle, Walter Kern, and Jeroen Kuipers. On the computation of the nucleolus of a cooperative game. *International Journal of Game Theory*, 30(1):79–98, 2001.

[31] Linda Farczadi. *Matchings and games on networks*. University of Waterloo, 2015.

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

[33] Steven Fortune, John Hopcroft, and James Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, 10(2):111–121, 1980.

[34] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[35] Michael R Garey and David S Johnson. Computers and intractability.

[36] Donald B Gillies. Solutions to general non-zero-sum games. *Contributions to the Theory of Games*, 4(40):47–85, 1959.

[37] Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical computer science*, 38:293–306, 1985.

[38] Daniel Granot, Frieda Granot, and Weiping R Zhu. Characterization sets for the nucleolus. *International Journal of Game Theory*, 27(3):359–374, 1998.

[39] G. Greco, E. Malizia, L. Palopoli, and F. Scarcello. The complexity of the nucleolus in compact games. *ACM Transactions on Computation Theory*, 7(1), 2014.

[40] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2 of *Algorithms and Combinatorics: Study and Research Texts*. Springer-Verlag, Berlin, 1988.

[41] Rudolf Halin. S-functions for graphs. *Journal of geometry*, 8(1-2):171–186, 1976.

[42] John Hardwick. *Graphical Algorithms for Finding the Nucleolus of Binary-Valued Matching Games*. PhD thesis, University of Illinois at Chicago, 2017.

[43] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

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

[45] Walter Kern and Daniël Paulusma. Matching Games: The Least Core and the Nucleolus. *Mathematics of Operations Research*, 28(2):294–308, 2003.

[46] L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 1980.

[47] Jon Kleinberg and Éva Tardos. Balanced outcomes in social exchange networks. In *Proceedings of the fourtieth annual ACM symposium on Theory of computing - STOC 08*, page 295, New York, New York, USA, 2008.

[48] Ton Kloks. *Treewidth: computations and approximations*, volume 842. Springer Science & Business Media, 1994.

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

[50] Jochen Könemann, Kanstantsin Pashkovich, and Justin Toth. Computing the nucleolus of weighted cooperative matching games in polynomial time. *Mathematical Programming*, 2020.

[51] Jochen Könemann and Justin Toth. A general framework for computing the nucleolus via dynamic programming. In *International Symposium on Algorithmic Game Theory*, pages 307–321. Springer, 2020.

[52] Tjalling C Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.

[53] Alexander Kopelowitz. Computation of the kernels of simple games and the nucleolus of n-person games. Technical report, Hebrew Univ Jerusalem (Israel) Dept of Mathematics, 1967.

[54] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[55] Soh Kumabe and Takanori Maehara. Convexity of b-matching games.

[56] Soh Kumabe and Takanori Maehara. Convexity of hypergraph matching game. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 663–671, 2020.

[57] Lap Chi Lau, Ramamoorthi Ravi, and Mohit Singh. *Iterative methods in combinatorial optimization*, volume 46. Cambridge University Press, 2011.

[58] Jean Lemaire. An application of game theory: cost allocation. *ASTIN Bulletin: The Journal of the IAA*, 14(1):61–81, 1984.

[59] Janny Leung, Martin Grotschel, Laszlo Lovasz, and Alexander Schrijver. Geometric Algorithms and Combinatorial Optimization. *The Journal of the Operational Research Society*, 1989.

[60] Michael Maschler, Bezalel Peleg, and Lloyd S Shapley. The kernel and bargaining set for convex games. *International Journal of Game Theory*, 1(1):73–93, 1971.

[61] Michael Maschler, Bezalel Peleg, and Lloyd S Shapley. Geometric Properties of the Kernel, Nucleolus, and Related Solution Concepts. *Mathematics of Operations Research*, 4(4):303–338, 1979.

[62] Leonardo Militano, Antonio Iera, and Francesco Scarcello. A fair cooperative content-sharing service. *Computer Networks*, 2013.

[63] Kanstantsin Pashkovich. Computing the Nucleolus of Weighted Voting Games in Pseudo-polynomial Time. pages 1–12, 2018.

[64] Daniël Paulusma. *Complexity aspects of cooperative games*. Twente University Press, 2001.

[65] Bezalel Peleg and Peter Sudhölter. *Introduction to the theory of cooperative games*, volume 34. Springer Science & Business Media, 2007.

[66] Ján Plesník. A note on the complexity of finding regular subgraphs. *Discrete mathematics*, 49(2):161–167, 1984.

[67] Jos Potters, Hans Reijnierse, and Amit Biswas. The nucleolus of balanced simple flow networks. *Games and Economic Behavior*, 54(1):205–225, 2006.

[68] Hans Reijnierse and Jos Potters. The B-nucleolus of TU-games. *Games and Economic Behavior*, 1998.

[69] Neil Robertson and Paul D Seymour. Graph minors. III. Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36(1):49–64, 1984.

[70] Thomas Rothvoß. The matching polytope has exponential extension complexity. *Journal of the ACM (JACM)*, 64(6):41, 2017.

[71] David Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on applied mathematics*, 17(6):1163–1170, 1969.

[72] David Schmeidler. The Nucleolus of a Characteristic Function Game. *SIAM Journal on Applied Mathematics*, 17(6):1163–1170, 1969.

[73] Alexander Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2002.

[74] Lloyd S Shapley. A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317, 1953.

[75] Lloyd S Shapley and Martin Shubik. Quasi-cores in a monetary economy with nonconvex preferences. *Econometrica: Journal of the Econometric Society*, pages 805–827, 1966.

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

[77] Tamás Solymosi, T. E.S. Raghavan, and Stef Tijs. Computing the nucleolus of cyclic permutation games. In *European Journal of Operational Research*, 2005.

[78] Tamás Solymosi and Tirukkannamangai ES Raghavan. An algorithm for finding the nucleolus of assignment games. *International Journal of Game Theory*, 23(2):119–143, 1994.

[79] Tamás Solymosi and Balázs Sziklai. Characterization sets for the nucleolus in balanced games. *Operations Research Letters*, 2016.

[80] Balázs Sziklai, Tamás Fleiner, and Tamás Solymosi. On the core and nucleolus of directed acyclic graph games. *Mathematical Programming*, 163(1-2):243–271, 2017.

Index