**Complexity Aspects of Cooperative Games**

Daniël Paulusma

2001

Ph.D. thesis
University of Twente

Twente University **Press**

Also available in print:
www.tup.utwente.nl/uk/catalogue/technical/cooperative-games

Daniël Paulusma

# Complexity Aspects of Cooperative Games

Twente University **Press**

Publisher: Twente University Press,
P.O. Box 217, 7500 AE Enschede, the Netherlands, www.tup.utwente.nl

Cover design: Jo Molenaar, [deel 4] ontwerpers, Enschede
Print: Grafisch Centrum Twente, Enschede



ISBN 9036515866

# COMPLEXITY ASPECTS OF COOPERATIVE GAMES

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. F.A. van Vught,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen
op vrijdag 29 juni 2001 te 13.15 uur

door

Daniël Paulusma
geboren op 23 mei 1974
te Leeuwarden

Dit proefschrift is goedgekeurd door de promotor

prof.dr. U. Faigle

# Preface

This thesis is the result of the research that I started in June 1997 under supervision of prof. U. Faigle. I want to thank him for the opportunity he offered me, and I appreciated the cooperation with him.

Most of all I want to thank Walter Kern for his advice and support during the project and the useful comments he gave on my thesis. I enjoyed working with him.

Next, I want to thank prof. J.M. Bilbao, prof. C. Hoede, prof. G. van der Laan, prof. S.H. Tijs and prof. G.J. Woeginger for taking place in my graduation committee.

Over the last years I was a member of the DOS department. I appreciated the nice working environment they offered, and I want to thank some of its (former) members in particular. Theo Driessen has taken a lot of interest in my research and gave some useful comments on my thesis. Cindy Kuijpers helped me out many times, when I had a problem with Latex. I want to thank her and my other room-mates, Marcel Hunting and Petrica Pop, for the pleasant time I had with them.

Finally, I want to thank my family and friends for their support.

Daniël Paulusma Enschede, June 2001

# Contents

# Chapter 1

# Introduction

Suppose a group of persons or organizations decide to work together in order to make a profit on some market or to do an investment (building an electricity network, constructing a railway etc.). Then the question arises how to split the joint profit or cost. This allocation problem can be modeled and analyzed by (cooperative) game theory, which tries to come up with "fair" solutions (Section 1.1). This thesis studies several well-known allocation problems, where the cost or profit is computed as the optimal value of some discrete optimization problem. We are especially interested in the computational complexity of some frequently used solution concepts. Section 1.2 deals with complexity theory and in Section 1.3 we treat some polyhedral theory. For readers not familiar with graph theory some basic terminology is included in Section 1.4. We end the chapter with an outline of the thesis.

## 1.1 Game theory

Game theory is a field of mathematical research that models and analyzes situations of conflict. In such a situation, two or more individuals (the *players*) with similar or different interests are taking actions or making decisions. The foundations of game theory can be found in the paper of Von Neumann [1928] and the book "Theory of Games and Economic Behavior" by Von Neumann and Morgenstern [1944]. The situation of conflict first is described as a mathematical model (the *game*). One then uses mathematical solution methods to come to a set of proposed pay-offs for each player. For a survey on game theory we refer to the books of Owen [1995] and Shubik [1982].

This thesis deals with so-called cooperative game theory, where players are allowed to cooperate with each other in order to optimize their profits (costs).

**Definition 1.1** A *cooperative game in characteristic function form* is given by an ordered pair $(N, v)$, where $N$ is a nonempty, finite set and $v : 2^N \to \mathbb{R}_+$ is a function satisfying $v(\emptyset) = 0$. The set $N$ is called the *player set* and its $n$ elements are the *players* of the game. The function $v$ is called the *characteristic function* of the game. □

If a subset of players in $N$ decide to work together, then they form a *coalition*. The mapping $v$ assigns to each coalition $S \subseteq N$ some outcome $v(S)$, the *value* or *worth* of coalition $S$. $v(S)$ does not depend on the players outside $S$. It can be interpreted as the maximal profit or minimal cost that the players in $S$ can achieve if they decide to form a coalition. In case of a *cost function* it is common to write $c$ instead of $v$. In the rest of this thesis we speak of a *(cooperative) game* instead of a cooperative game in characteristic function form. In case of a profit function $v$ we may speak of a *profit game* and in case of a cost function $c$ we may speak of a *cost game*.

In cooperative game theory it is often assumed that the players decide to work all together. In that case the *grand coalition* $N$ is formed. The central problem is to find a "fair" distribution of the total value $v(N)$ among the individual players $i \in N$. Let $x_i$ denote the amount allocated to player $i \in N$. A vector $x \in \mathbb{R}^N$ is an *allocation* if $x$ is *efficient*, i.e., $x(N) = v(N)$. (Throughout the thesis, we use the shorthand notation $x(S) = \sum_{i \in S} x_i$.)

A *solution concept* prescribes for each game a set of allocations. In the literature a solution concept can also assign pay-off vectors that are not efficient, but here we will assume that all vectors prescribed by some solution concept are allocations.

The choice for a specific solution concept depends on the notion of "fairness" that has been specified within the decision model. Examples of solution concepts that might suggest more than one allocation are the core (Gillies [1959]) and the kernel (Davis and Maschler [1965]). A solution concept that suggests at most one allocation for each game is called a *value*. Well-known values are the Shapley value (Shapley [1953]) and the nucleolus (Schmeidler [1969]).

**Example 1.1** *Bankruptcy game* (O'Neill [1982])
Consider a situation where a company becomes a bankrupt and some creditors bring in a number of claims. The question here is how to divide the

available amount left by the company among the creditors. This situation can be modeled as a cooperative game $(N, v)$. $N$ is the set of creditors and the characteristic function $v$ is given by

$$v(S) := \max\{0, E - \sum_{i \in N \setminus S} d_i\} \quad \text{for all } S \subseteq N,$$

where $d_i$ is the claim of creditor $i \in N$ and $E < \sum_{i \in N} d_i$ denotes the available amount left by the company. A value $v(S)$ can be interpreted as a lower bound of the amount that the players in a coalition $S \subseteq N$ will receive if they do not protest against the claims of players outside $S$. A solution concept provides for one or more possible distributions of $v(N) = E$ among the creditors. □

## 1.2 Discrete optimization

In a general *discrete optimization problem* we have to optimize an *objective function* over a certain set called the *(feasible) solution set*. This set can be described by (in)equality constraints and integrality restrictions on some or all of the variables. For a survey on discrete optimization we refer to the books of Nemhauser and Wolsey [1999], Papadimitriou and Steiglitz [1982] and Schrijver [1986].

An important aspect of a solution method for an optimization problem is its computational complexity. In this section we briefly go into the main concepts of complexity theory. More extended descriptions of these concepts can be found in the books mentioned above and also in the book of Garey and Johnson [1979].

Assume that we have a solution set $S$ and objective function $f : S \to \mathbb{R}$. Then the general optimization problem is

$$\text{find a feasible solution } \bar{s} \in S \text{ such that } f(\bar{s}) = \max\{ f(s) \mid s \in S\}.$$

The associated *decision problem* is

$$\text{Given } f^* \in \mathbb{R}, \text{ is there a solution } s \in S \text{ such that } f(s) \geq f^*?$$

So a decision problem is a question that has to be answered only by “yes” or “no”. Clearly, every solution method that solves the optimization problem can be used to solve the associated decision problem. For a *discrete* optimization

problem very often also the opposite is true: If one can solve the decision problem efficiently, then one can solve the corresponding optimization problem efficiently. For this reason, the theory of complexity deals in the first place with decision problems.

An *instance* of an optimization problem is a set of data that is obtained when all the parameters that define the problem are fixed. An *algorithm* is a list of instructions that solves every instance of a problem in a finite number of steps. So the output of an algorithm is "yes" in case there is a solution and "no" otherwise.

**Example 1.2** EXACT 3-COVER (X3C)

*Instance*: A finite set $W$ with $3q$ elements and a collection $U$ containing $k \geq q$ 3-element subsets of $W$.

*Question*: Does $U$ contain an *exact cover* for $W$, i.e., is there a subcollection $U' \subseteq U$ such that every element of $W$ occurs in exactly one member of $U'$?

Determining the size of a minimum cover that contains every element of $W$ is the associated (discrete) optimization problem. □

We assume that every instance is described as a string in a binary encoding. The *size of a problem instance* is the length of the encoding, i.e., the number of bits necessary to represent the instance. We denote the size of a rational number $a \in \mathbb{Q}$ by $< a >$. For example, the size of a linear inequality $ax \leq b$, where $a$ and $b$ are rational numbers, is equal to $1 + < a > + < b >$, and the size of a vector $x \in \mathbb{Q}^n$ is equal to $n + < x_1 > + < x_2 > + \ldots + < x_n >$.

Because we want to obtain a general classification of problems, we assume that every algorithm runs on the same machine, the so-called Turing machine, and we measure the *computation time* of an algorithm by its number of performed elementary operations (additions, multiplications, comparisons etc.).

The *running time* $t(\xi)$ of an algorithm is defined as the maximum (computation) time required to solve any problem instance with size $\xi$. In this way we have an absolute guarantee on the time required to solve an instance independent of any probability distribution of the instances.

An algorithm is said to be a *polynomial time algorithm* or *efficient* if its running time $t(\xi)$ is bounded by a polynomial in $\xi$, i.e, if for all $\xi \in \mathbb{N}$, $t(\xi) = \mathcal{O}(\xi^p)$ for some fixed $p \in \mathbb{N}$. (For two functions $f : \mathbb{N} \to \mathbb{R}_+$ and $g : \mathbb{N} \to \mathbb{R}_+$

we write $f(n) = \mathcal{O}(g(n))$ if there exists positive numbers $c, n_0 \in \mathbb{N}$ such that $f(n) \leq cg(n)$ for $n \geq n_0$.)

Decision problems that are solvable in polynomial time are considered to be "easy". The class of these problems is denoted by $\mathcal{P}$. $\mathcal{P}$ includes for example the minimum spanning tree problem and the weighted matching problem. Also several game theoretic problems such as computing the nucleolus of convex games (Kuipers [1996]) or computing the nucleolus of assignment games (Solymosi and Raghavan [1994]) can be solved efficiently.

An algorithm is said to be an *exponential time algorithm* if for all $\xi \in \mathbb{N}$, $t(\xi) = \mathcal{O}(2^{\xi^p})$ for some fixed $p \in \mathbb{N}$. The class of decision problems solvable in exponential time is denoted by $EXP$. Most discrete optimization problems belong to this class.

If a problem is in $EXP \backslash \mathcal{P}$, then solving large instances of this problem will be difficult. However, for a lot of problems it is not known whether they are in $EXP \backslash \mathcal{P}$ or in $\mathcal{P}$. X3C is an example of such a problem: No polynomial time algorithm for X3C is known, and until so far X3C has not been proven to be in $EXP \backslash \mathcal{P}$ either. It is common believe that these kind of problems do not belong to $\mathcal{P}$. Moreover, some of these problems can be considered to be "harder" than others. In order to make a more specific distinction the classes $\mathcal{NP} \subseteq EXP$ and $\mathcal{NP}$-complete $\subseteq \mathcal{NP}$ are introduced.

$\mathcal{NP}$ is the class of decision problems that can be solved by a so-called *nondeterministic algorithm*. Such an algorithm consists of a guessing stage and a checking stage. In the first stage one guesses a solution and in the second stage the algorithm checks if this solution satisfies the problem conditions. If the checking stage can be done in polynomial time (with respect to the size of the instance in the guessing stage), then the problem is said to be in $\mathcal{NP}$. X3C is an example of a decision problem that is a member of $\mathcal{NP}$. A polynomial nondeterministic algorithm for X3C would be:

| | |
|---|---|
| guessing stage: | guess a subcollection $U' \subseteq U$ (of $q$ elements). |
| checking stage: | If $U'$ is an exact cover, then output "yes";<br>otherwise return. |

Obviously $\mathcal{P} \subseteq \mathcal{NP}$ holds. It is widely assumed that $\mathcal{P} = \mathcal{NP}$ is very unlikely. The class $\mathcal{NP}$ contains a subclass of problems that are considered to be the hardest problems in $\mathcal{NP}$. These problems are called $\mathcal{NP}$*-complete*

*problems*. They have the following property: If one can prove that an $\mathcal{NP}$-complete problem is a member of $\mathcal{P}$, then $\mathcal{P} = \mathcal{NP}$ holds. The technique used here is that of polynomially transforming one problem into another. A decision problem $\Pi_1$ is *polynomially transformable* to a decision problem $\Pi_2$ if there exists an algorithm that for every instance $\sigma_1$ of $\Pi_1$ produces in *polynomial time* exactly one instance $\sigma_2$ of $\Pi_2$ such that the following holds:

the answer for $\sigma_1$ is "yes" if and only if the answer of $\sigma_2$ for "yes".

This means that a polynomial time algorithm $\mathcal{A}$ for $\Pi_2$ can also be used to solve an instance of $\Pi_1$ efficiently: First polynomially transformate $\Pi_1$ into $\Pi_2$ and then use $\mathcal{A}$.

**Definition 1.2** A decision problem $\Pi$ is called *$\mathcal{NP}$-complete* if $\Pi$ is in $\mathcal{NP}$ and all other decision problems in $\mathcal{NP}$ can be polynomially transformed to $\Pi$. □

Note that polynomial transformability is a transitive relation. If $\Pi_1$ is polynomially transformable to $\Pi_2$ and $\Pi_2$ is polynomially transformable to $\Pi_3$, then $\Pi_1$ is polynomially transformable to $\Pi_3$. Clearly $\mathcal{P} = \mathcal{NP}$ must hold if one can prove for any $\mathcal{NP}$-complete problem to be in $\mathcal{P}$. If we want to prove that a decision problem $\Pi$ is $\mathcal{NP}$-complete, then we only have to show that

(i) $\Pi$ is in $\mathcal{NP}$.

(ii) Some decision problem already known to be $\mathcal{NP}$-complete can be polynomially transformed to $\Pi$.

Our example X3C is one of the six basic $\mathcal{NP}$-complete problems in Garey and Johnson [1979]. The first problem proven to be $\mathcal{NP}$-complete is the satisfiability problem (Cook [1971]). Other well-known $\mathcal{NP}$-complete problems are the traveling salesman problem and Hamiltonian cycle. Some examples of $\mathcal{NP}$-complete problems in game theory are: deciding whether the core of a minimum coloring game is empty or not (Deng, Ibaraki and Nagamochi [1999]) and testing membership in the core of minimum cost spanning tree games (Faigle, Kern, Fekete and Hochstättler [1997]).

Another technique that is used for proving that a problem can be solved in polynomial time in case another problem can is polynomial reduction: Suppose we have two problems $\Pi_1$ and $\Pi_2$ not necessarily decision problems.

A *polynomial reduction* from $\Pi_1$ to $\Pi_2$ is an algorithm $\mathcal{A}_1$ for $\Pi_1$ that uses an algorithm $\mathcal{A}_2$ for $\Pi_2$ as a subroutine and that would be a polynomial time algorithm for $\Pi_1$ if $\mathcal{A}_2$ were a polynomial time algorithm for $\Pi_2$. This is a more general technique than polynomial transformation, which can be seen as a special case of reduction in which the subroutine $\mathcal{A}_2$ is used only once.

**Definition 1.3** An optimization problem $\Pi$ is called $\mathcal{NP}$*-hard* if there exists an $NP$-complete decision problem that can be polynomially reduced to $\Pi$. □

Note that this definition includes all optimization problems for which the associated decision problem is $\mathcal{NP}$-complete. In particular, (the optimization version of) X3C is $\mathcal{NP}$-hard. Polynomially reducibility is a transitive relation. Therefore, in order to prove $\mathcal{NP}$-hardness for some problem it suffices to show that a known $\mathcal{NP}$-hard problem can be reduced to it. Some $\mathcal{NP}$-hardness results in game theory are computing the nucleolus of minimum cost spanning tree games ( Faigle, Kern and Kuipers [1998a]) and computing the Shapley value in weighted majority games (Deng and Papadimitriou [1994]).

## 1.3 Polyhedral theory

The theory we discuss in this section is derived from the the books of Schrijver [1986] and Grötschel, Lovász and Schrijver [1993].

Consider a set of points $X = \{x^1, \ldots, x^k\} \subseteq \mathbb{R}^n$ and a vector $\lambda \in \mathbb{R}^k$. The linear combination $x = \sum_{i=1}^k \lambda_i x^i$ is an *affine combination* if $\sum_{i=1}^k \lambda_i = 1$, and $x$ is called a *convex combination* if besides $\sum_{i=1}^k \lambda_i = 1$, $\lambda \geq 0$. $X$ is called *linearly (affinely) independent* if no point $x^i \in X$ can be written as a linear (affine) combination of the other points in $X$.

A subset $S \subseteq \mathbb{R}^n$ is *convex* if for every finite number of points $x^1, \ldots, x^k \in S$ any convex combination of these points is a member of $S$.

A nonempty set $C \subseteq \mathbb{R}^n$ is called a *convex cone* if $\lambda x + \mu y \in C$ for all $x, y \in C$ and for all real numbers $\lambda, \mu \geq 0$.

A convex set $P \subseteq \mathbb{R}^n$ is a *polyhedron* if there exists an $m \times n$ matrix $A$ and a vector $b \in \mathbb{R}^m$ such that

$$P = P(A, b) = \{x \in \mathbb{R}^n \mid Ax \leq b\}.$$

We call $Ax \leq b$ a *system of linear inequalities*.

A polyhedron $P \subseteq \mathbb{R}^n$ is *bounded* if there exist vectors $l, u \in \mathbb{R}^n$ such that $l \leq x \leq u$ for all $x \in P$. A bounded polyhedron is called a *polytope*.

For our purposes, only rational polyhedra are of interest. A polyhedron is *rational* if it is the solution set of a system $Ax \leq b$ of linear inequalities, where $A$ and $b$ are rational. A rational polyhedron $P \subseteq \mathbb{R}^n$ is said to have *facet complexity at most* $\varphi$, if there exists a system $Ax \leq b$ of linear inequalities with rational coefficients that has $P$ as its solution set and such that the size of each inequality of the system is at most $\varphi$. From now on we will implicitly assume a polyhedron to be rational.

A point $x \in P$ is called a *vertex* of $P$ if $x$ cannot be written as a convex combination of other points in $P$. The following theorem presents a tight upper bound on the size of a vertex.

**Theorem 1.1** *Let $P \subseteq \mathbb{R}^n$ be a polyhedron with facet complexity at most $\varphi$. Then each vertex has size polynomially bounded by $\mathcal{O}(n^2\varphi)$.* □

The *dimension* of a polyhedron $P \subseteq \mathbb{R}^n$ is equal to the maximum number of affinely independent points in $P$ minus 1. An *implicit equality* of the system $Ax \leq b$ is an inequality $\sum_{j=1}^{n} a_{ij}x_j \leq b_i$ of that system such that $\sum_{j=1}^{n} a_{ij}x_j = b_i$ for all vectors $x \in P(A, b)$. For $Ax \leq b$ we denote the (sub)system of implicit equalities by $A^{=}x \leq b^{=}$. Let rank($A$) denote the *rank of a matrix*, i.e., the maximum number of linearly independent row vectors. We have the following standard result.

**Theorem 1.2** *The dimension of a polyhedron $P(A, b) \subseteq \mathbb{R}^n$ is equal to $n - \mathrm{rank}(A^{=})$.* □

A subset $H \subseteq \mathbb{R}^n$ is called a *hyperplane* if there exists a vector $h \in \mathbb{R}^n$ and a number $\alpha \in \mathbb{R}$ such that

$$H = \{x \in \mathbb{R}^n \mid h^T x = \alpha\}.$$

A *separating hyperplane* for a convex set $S$ and vector $x \notin S$ is a hyperplane given by a vector $h \in \mathbb{R}^n$ and a number $\alpha \in \mathbb{R}$ such that $h^T x \leq \alpha$ and $h^T y > \alpha$ holds for all $y \in S$.

The *separation problem* for a polyhedron $P \subseteq \mathbb{R}^n$ is, given a vector $x \in \mathbb{R}^n$, to decide whether $x \in P$ or not, and, if $x \notin P$, to find a separating hyperplane for $P$ and $x$. A *separation algorithm* for a polyhedron $P$ is an algorithm that solves the separation problem for $P$.

*Linear programming* (LP) deals with maximizing or minimizing a linear function over a polyhedron. If $P \subseteq \mathbb{R}^n$ is a polyhedron and $d \in \mathbb{R}^n$, then we call the optimization problem

$$(LP) \quad \max\{d^T x \mid x \in P\}$$

a *linear program*. A vector $x \in P$ is called a *feasible solution* of the linear program and $x^*$ is called an *optimal solution* if $x^*$ is feasible and $d^T x^* \geq d^T x$ for all feasible solutions $x$. If for all $x \in P$ a solution $x^* \in P$ exists with $d^T x^* > d^T x$, then $(LP)$ is *unbounded*. If $(LP)$ has no optimal solution then it is either infeasible or unbounded.

Khachiyan [1979] showed that LP can be solved in polynomial time by means of the ellipsoid method. Grötschel, Lovász and Schrijver [1981] refined this method in such a way that the computational complexity of optimizing a linear function over a convex set $S$ depends on the complexity of the separation problem for $S$. If the convex set is a polyhedron, this can be stated as follows (see also Grötschel, Lovász and Schrijver [1993]):

**Theorem 1.3** *There exists an algorithm $ELL$ and a polynomial $p$ in two variables $n$ and $\varphi$ such that the following holds:*

*For each polyhedron $P \subseteq \mathbb{R}^n$ with facet complexity at most $\varphi$ for which there exists a separation algorithm $SEP$, $ELL$ solves the linear program*

$$\max\{d^T x \mid x \in P\}$$

*in time bounded by a polynomial in $n, \varphi, < d >$, and $T$, where $T$ is the maximum time required by $SEP$ on input vectors $x$ of size $p(n, \varphi)$.* □

Here, "solving a linear program" not only means finding an optimal solution, but it also means detecting the cases in which the linear program is infeasible or unbounded.

Now assume that for a polyhedron $P \subseteq \mathbb{R}^n$ with facet complexity at most $\varphi$ a separation algorithm $SEP$ exists that solves on inputs $x$ the separation problem for $P$ in time bounded by a polynomial in $n, \varphi$ and $< x >$. Then from Theorem 1.3 it follows immediately that $ELL$ solves the linear program $\max\{d^T x \mid x \in P\}$ in time bounded by a polynomial in $n, \varphi$, and $< d >$.

**Remark 1.1** For any polyhedron $P$, by definition, a linear system $Ax \leq b$ exists such that $P = P(A, b)$. In practice this description may be unknown

or consists of too many inequalities. Theorem 1.3 shows that this is not a problem as long as the facet complexity of $P$ is low and an efficient separation algorithm for $P$ is known. Moreover, in that case the running time of $ELL$ only depends on the size but not on the number of inequalities defining $P$. □

## 1.4 Graph theory

The games we study can be represented by graphs. We associate a cooperative game $(N, v)$ with some graph $G$ in the following way: The player set $N$ is the node set of $G$ and the value of a coalition $v(S)$ is determined as the value of a discrete optimization problem on $G$ (e.g., the weight of a maximum matching). In this section we include some terminology for readers not familiar with graph theory. For more on graph theory we refer to the book of Bondy and Murty [1976].

A *graph* $G$ is an ordered pair $(V, E)$, where $V$ is a nonempty, finite set called the *node set* and $E$ is a set of (unordered) pairs $(i, j)$ with $i, j \in V$ called the *edge set*. If another graph $G'$ has been defined we write $G'(V)$ and $G'(E)$ to make a distinction.

The elements of $V$ are called *nodes* and the elements of $E$ are called *edges*. If $e = (i, j) \in E$ we say that node $i$ and node $j$ are *adjacent*. In such a case $i$ and $j$ are called the *end points* of $e$ or *incident* with $e$. Furthermore, we say that $e$ is *incident* with $i$ and $j$, and a subset $E' \subseteq E$ of edges is said to *cover* the set of nodes incident with some edge in $E'$. A node $j$ for which there is an edge $(i, j) \in E$ is a *neighbor* of $i$. The number of neighbors of a node $i$ is called the *degree* of $i$ and denoted by $\delta(i)$ and a node with no neighbors is called an *isolated node*. Two edges are called *adjacent* if they have a common incident node.

A *multigraph* is a graph with possibly more than one edge between two nodes.

We speak of a *weighted* graph if a *weight function* $w : E \to \mathbb{R}$ is defined on the edge set $E$ of a graph $G$. The number $w(e)$ is the *weight* of an edge $e \in E$. (It can usually be interpreted as a certain profit or cost.) The *weight of a subset* $E' \subseteq E$ is equal to the sum of the weights of its edges and denoted by $w(E')$.

A *complete graph* is a graph with an edge between every pair of nodes. The complete graph on $n$ nodes is denoted by $K_n$.

A graph $G$ is a *bipartite graph* with *node classes* $V_1$ and $V_2$ if $V = V_1 \cup V_2$, $V_1 \cap V_2 = \emptyset$ and each edge joins a node of $V_1$ to a node of $V_2$.

A *subgraph* of $G$ is a graph $G' = (V', E')$ with $V' \subseteq V$ and $E' \subseteq E$. $G'$ is called the *induced* subgraph by $V'$, denoted by $G|_{V'}$, if $E' = \{(i, j) \in E \mid i, j \in V'\}$. If $V' \subseteq V$ then we let $G \backslash V'$ denote the graph obtained by removing $V'$ (and the edges incident with nodes in $V'$). If $E' \subseteq E$ then $G \backslash E'$ denotes the graph obtained by removing the edges in $E'$. We say that a graph $G$ *contains* a graph $G'$ if $G$ has $G'$ as subgraph.

A *path* from $i$ to $j$ is a graph $P = (V, E)$ with a node set that can be ordered as $v_0, \ldots, v_n$ with $v_0 = i$ and $v_n = j$ such that $E = \{(v_k, v_{k+1}) \mid k = 0, \ldots, n-1\}$. The nodes $i$ and $j$ are called the *end points* of the path and $n$ is the *length* of the path. We also write $P = v_0 v_1 \ldots v_n$.

A *cycle* is a graph for which the node set can be ordered as $v_0, \ldots, v_n$ such that $E = \{(v_k, v_{k+1}) \mid k = 0, \ldots, n-1\} \cup \{(v_n, v_0)\}$. We denote a cycle on $n$ nodes by $C_n$.

A graph $G$ is called *connected* if $G$ contains a path from $i$ to $j$ for each two nodes $i, j \in V$. A *component* $G'$ of $G$ is a maximal connected subgraph of $G$, i.e., if $\hat{G}$ is a connected subgraph of $G$ and $G'$ is a subgraph of $\hat{G}$, then $\hat{G} = G'$. The *size of a component* is its number of nodes. We denote the size of a component $G'$ by $|G'|$. A component is called *even* or *odd* if it has an even respectively odd number of nodes.

A *tree* is a connected graph $T$ that does not contain any cycle. A node $i \in V$ is called a *leaf* of a tree $T = (V, E)$, if $i$ has exactly one neighbor. A *forest* is a graph (not necessarily connected) that does not contain any cycle. A *spanning tree* of $G = (V, E)$ is a tree $(V', E')$ with $V' = V$.

A *matching* in a graph $G = (V, E)$ is a subset $M$ of $E$ such that no two edges in $M$ have a common end point. A matching $M$ *matches* a subset $V_1$ into $V_2$, if each edge in $M$ is incident with a node in $V_1$ and a node in $V_2$.

A *node cover* in a graph $G = (V, E)$ is a subset $V'$ of $V$ such that every edge in $E$ is incident with a node in $V'$.

If the pairs $(i, j)$ in the edge set of a graph are ordered, then we speak of a *directed graph* or *digraph* and we call such an ordered pair $(i, j)$ an *arc*. In this case the edge set is usually denoted by $A$. If $a = (i, j)$ is an arc, then node $i$ is called the *tail* $t(a)$ of $a$ and $j$ is called the *head* $h(a)$ of $a$. The arc $a$ is an *outcoming arc* of node $i$ and is an *incoming arc* of node $j$. The *outdegree*

of a node $i$, denoted by $\delta^+(i)$, is the number of outcoming arcs of $i$, and the *indegree* $\delta^-(i)$ is the number of incoming arcs of $i$.

A *network* $(G, l, u)$ is a directed graph $G = (V, A)$ with two distinguished nodes $s$ and $t$ and two functions $l : A \to \mathbb{R}_+$ and $u : A \to \mathbb{R}_+$ such that $l(a) \leq u(a)$ for all $a \in A$. $s$ is called the *source* and $t$ is called the *sink*. $l$ is the *lower capacity function* and $u$ is called the *upper capacity function*. $l(a)$ and $u(a)$ are respectively the *lower* and *upper capacity* of arc $a \in A$.

A *flow* from $s$ to $t$ in a network $(G, l, u)$ is a function $f : A \to \mathbb{R}$ such that for all $i \in V \backslash \{s, t\}$

$$\sum_{t(a)=i} f(a) = \sum_{h(a)=i} f(a).$$

A flow $f$ is called *feasible* if for all $a \in A \quad l(a) \leq f(a) \leq u(a)$. The following standard result can be deduced from the Hoffman-Kruskal theorem (Hoffman and Kruskal [1956]).

**Theorem 1.4** *Let $(G, l, u)$ be a network with integral capacities $l$ and $u$. Then there exists an integral feasible flow for $(G, l, u)$, if there exists a feasible flow for $(G, l, u)$.* □

## 1.5 Outline of the thesis

The usefulness of a solution concept is not only determined by its modeling adequacy but also by its computational complexity. In this thesis we study the complexity of several solution concepts with respect to various classes of cooperative games.

In Chapter 2 we discuss a number of solution concepts for cooperative games, in particular the core and the nucleolus. Furthermore, we describe some least core concepts and variants of the nucleolus like the nucleon and the per-capita nucleolus.

Chapter 3 concentrates on minimum cost spanning tree games. Various least core concepts, including the classical least core, are analyzed. By a reduction from minimum cover problems we prove that computing an element in these least cores is $\mathcal{NP}$-hard for minimum cost spanning tree games. As a consequence, computing the nucleolus, the nucleon and the per-capita nucleolus of minimum cost spanning tree games is also $\mathcal{NP}$-hard.

Chapter 4 deals with matching games. In particular, we study cardinality games and a generalization thereof (node matching games). We show that the nucleolus and hence elements in the least core of such games can be computed efficiently. The case of general (weighted) matching games remains open.

In Chapter 5 we study complexity aspects of sports competitions like national football leagues and related games. The central problem is the so-called *elimination problem*, i.e., to determine at a given intermediate state of the competition whether a particular team still has a chance of winning the competition. Our main result states that the new FIFA-rules (3 : 0 for a win) have complicated this problem considerably. We completely characterize the complexity of this problem and relate it to the core complexity of a corresponding game.

# Chapter 2

# Solution concepts for cooperative games

Recall that a solution concept $\Phi$ prescribes a set $\Phi(N, v) \subseteq \mathbb{R}^N$ of allocations for any cooperative game $(N, v)$. In Section 2.1 we treat some elementary properties that a solution concept might have. The choice for a particular solution concept depends on the "fairness" of its properties with respect to the specific game one considers. We also make some general statements on the computational complexity of solution concepts in this section. Section 2.2 deals with the core and the least core of a game. The nucleolus is discussed in Section 2.3. In Section 2.4 we generalize the concepts of the previous two sections resulting in the $f$-least core and the $f$-nucleolus (special cases: the nucleon and the per-capita nucleolus). We end the chapter with a description of the Shapley value of a game (Section 2.5).

## 2.1 Solution concepts: properties and complexity

Let $(N, v)$ be a cooperative game. An allocation $x \in \mathbb{R}^N$ is said to be an *imputation* or *individually rational* if $x_i \geq v(i)$ for all $i \in N$. An imputation allocates to each player $i \in N$ at least the amount that $i$ can receive on his own. Note that this definition is related to a profit game. In case of a cost game one has to reverse the inequalities ($x_i \leq c(i)$). The set of imputations for a game $(N, v)$ is denoted by $I(N, v)$.

$$I(N, v) = \{x \in \mathbb{R}^N \mid x(N) = v(N)\ , x_i \geq v(i) \text{ for all } i \in N\}.$$

Obviously, $I(N, v)$ is nonempty if and only if $\sum_{i \in N} v(i) \leq v(N)$. The set of allocations or *pre-imputations* for a game $(N, v)$ is denoted by $I^*(N, v)$.

$$I^*(N, v) = \{x \in \mathbb{R}^N \mid x(N) = v(N)\}.$$

In the rest of this chapter we mainly consider profit games, but the same theory can be applied to cost games. In that case some definitions have to be adjusted (more or less trivially by reversing some inequalities). In the literature sometimes the notions anti-core and anti-nucleolus are used, if one considers a cost game. Here we will still speak of the core and nucleolus of a cost game.

Below we summarize a number of elementary properties or axioms for a solution concept $\Phi$ defined on a class $\mathcal{G}$ of cooperative games (see, e.g., Driessen [1991]). We use the following notations: If $X$ is a subset of $\mathbb{R}^N$ and $\lambda \in \mathbb{R}$, then the set $\lambda X$ is equal to $\{\lambda x \mid x \in X\}$. If $Y$ is also a subset of $\mathbb{R}^N$, then the set $X + Y$ denotes the sum of $X$ and $Y$, i.e., $X + Y = \{x + y \mid x \in X, y \in Y\}$.

- *Individual rationality (Ind)*
  A solution concept $\Phi$ is called individually rational if for all games $(N, v) \in \mathcal{G}$ and all $x \in \Phi(N, v)$ $x$ is individually rational.

- *Nonemptiness (Non)*
  A solution concept $\Phi$ has this property, if for all games $(N, v) \in \mathcal{G}$

  $$\Phi(N, v) \neq \emptyset.$$

- *Dummy player property (Dum)*
  A player $i \in N$ is called a *dummy* in a game $(N, v)$ if $v(S) - v(S \backslash i) = v(i)$ for all $S \subseteq N$ with $i \in S$. A solution concept $\Phi$ has the dummy player property if for all games $(N, v) \in \mathcal{G}$, all dummy players $i \in N$, and all $x \in \Phi(N, v)$

  $$x_i = v(i).$$

  According to a solution concept that satisfies *Dum*, a dummy player receives exactly the amount that he contributes to every coalition.

- *Symmetry (Sym)*
  Let $\pi : N \to N$ be a permutation. The game $(N, v^\pi)$ is given by

$v^\pi(\pi(S)) = v(S)$ for all $S \subseteq N$. For any vector $x \in \mathbb{R}^N$ let the vector $x^\pi$ be given by $x^\pi_{\pi(i)} = x_i$ for all $i \in N$. For any set $X \subseteq \mathbb{R}^N$, $X^\pi$ defines the set

$$\{y \in \mathbb{R}^N \mid y = x^\pi \text{ for some } x \in X\}.$$

A solution concept $\Phi$ is symmetric if for all games $(N, v) \in \mathcal{G}$ and all permutations $\pi : N \to N$

$$\Phi(N, v^\pi) = \Phi(N, v)^\pi.$$

A symmetric solution concept is not influenced by renumbering of the player set.

- *Invariance (Inv)*
  Let $(N, v) \in \mathcal{G}$. Let $\lambda \in \mathbb{R}$ and $a \in \mathbb{R}^N$. The game $(N, \lambda v + a)$ is given by $(\lambda v + a)(S) = \lambda v(S) + a(S)$ for all $S \subseteq N$. A solution concept $\Phi$ is called invariant if for all games $(N, v) \in \mathcal{G}$, all $\lambda \in \mathbb{R}$, and all $a \in \mathbb{R}^N$

  $$\Phi(N, \lambda v + a) = \lambda \Phi(N, v) + \{a\}.$$

- *Additivity (Add)*
  Let $(N, v), (N, w) \in \mathcal{G}$. The game $(N, v + w)$ is given by $(v + w)(S) = v(S) + w(S)$ for all $S \subseteq N$. A solution concept $\Phi$ is additive if for all games $(N, v) \in \mathcal{G}$ and all $(N, w) \in \mathcal{G}$

  $$\Phi(N, v + w) = \Phi(N, v) + \Phi(N, w).$$

A singleton is a set that contains exactly one element. In the previous chapter we have stated that a solution concept that prescribes at most one allocation for a game $(N, v) \in \mathcal{G}$ is called a value. If a value $\Phi(N, v)$ of a game $(N, v)$ is nonempty, then we write, as a shorthand notation, $\Phi(N, v)$ not only to indicate the singleton but also to indicate the allocation itself.

This thesis in particular studies classes of games, where each game $(N, v)$ can be presented "implicitly" in terms of a *(weighted) discrete structure* from which we can derive the coalition values. More precisely, $(N, v)$ will be defined by a pair $(G, w)$, where $G$ is a graph with node set $V = N$, and $w$ is a weight function defined on the nodes and/or edges of $G$. Given a coalition $S \subseteq V$ we can compute $v(S)$ by solving a (mostly easy) discrete optimization problem corresponding with $S$ (cf. also Bilbao [2000]).

**Example 2.1** Let $G = (N, E)$ be a graph with node set $N$ and edge set $E$. Let $w : E \to \mathbb{R}_+$ be a weight function defined on $E$. For $S \subseteq N$ we denote the set of edges joining nodes of $S$ by $E(S)$. We define a cooperative game $(N, v)$ with characteristic function $v$ given by

$$v(S) = \sum_{e \in E(S)} w(e) \text{ for all } S \subseteq N.$$

□

We assume that the $v$-values we derive from the underlying discrete structure $(G, w)$ have size polynomially bounded in the size of the structure. Hence to any game $(N, v)$ in our class we may associate a *size* $< N, v >$, which is polynomially bounded in the size of the underlying structure and at the same time is an upper bound for the size of any $v$-value $v(S)$.

**Example 2.2** In Example 2.1 we may define $< N, v >= |N|^2 < w >$, where $< w >= \max\{< w(e) > \mid e \in E\}$. □

Now consider an algorithm $\mathcal{A}$ that on input $(N, v)$ (obtained from a discrete structure $(G, w)$) computes one or more allocations according to a solution concept $\Phi$. Then $\mathcal{A}$ is a polynomial time algorithm if its running time is polynomially bounded in $< N, v >$.

## 2.2 The core and least core

The *core* of a game is the most fundamental solution concept within cooperative game theory. The idea of the core essentially goes back to Von Neumann and Morgenstern [1944] and Ransmeier [1942]. The core was first introduced and named in Gillies [1959].

**Definition 2.1** The *core* of a game $(N, v)$ is the following set of allocations:

$$\text{core}(N, v) := \{x \in \mathbb{R}^N \mid x(N) = v(N), x(S) \geq v(S) \text{ for all } \emptyset \neq S \neq N\}.$$

□

A vector $x \in$ core$(N, v)$ is said to be a *core allocation*. A core allocation $x$ guarantees each coalition $S \subseteq N$ to be satisfied in the sense that it gets at least what it could gain on its own. As a solution concept the core satisfies *Ind*, *Sym* and *Inv*. Note that the core allocations form a polyhedron in $\mathbb{R}^N$.

A game $(N, v)$ is called *superadditive* if

$$v(S) + v(T) \leq v(S \cup T) \quad \text{for all } S, T \subseteq N, S \cap T = \emptyset.$$

In a superadditive game it is very likely that the grand coalition $N$ will be formed.

The following example shows that even for a superadditive three-person game the core can be empty. (Actually this is an example of a matching game on $K_3$ with unit edge weights, cf. Chapter 4.)

**Example 2.3** Consider the player set $N = \{1, 2, 3\}$ and let $v : 2^N \to \mathbb{R}_+$ be given by $v(1) = v(2) = v(3) = 0$ and $v(1, 2) = v(1, 3) = v(2, 3) = v(N) = 1$. If $x \in \mathbb{R}^3$ were in the core, then $x_1 + x_2 \geq 1$ and $x_3 \geq 0$. Together with $x(N) = 1$, this implies $x_3 = 0$. In the same way we can deduce $x_1 = x_2 = 0$, a contradiction. Hence core$(N, v)$ is empty. □

Because many interesting games, such as matching games, may have an empty core, the *additive $\epsilon$-core* of a game $(N, v)$ has been introduced (Shapley and Shubik [1966]). For a given $\epsilon \in \mathbb{R}$, the additive $\epsilon$-core of $(N, v)$ is the set of allocations

$$\{x \in \mathbb{R}^N \mid x(N) = v(N), x(S) \geq v(S) + \epsilon \text{ for all } \emptyset \neq S \neq N\}.$$

Obviously there exists an $\epsilon \in \mathbb{R}$ such that the additive $\epsilon$-core of $(N, v)$ is nonempty. If we maximize $\epsilon$ under the restriction that $\epsilon$-core of $(N, v)$ is nonempty, then we obtain the *least core* of a game $(N, v)$ (Maschler, Peleg and Shapley [1979]).

**Definition 2.2** The *least core* of a game $(N, v)$, denoted by leastcore$(N, v)$, consists of all optimal solutions $x \in \mathbb{R}^N$ of the linear program

$$\begin{array}{lllll} (LC) & \max & \epsilon & & \\ & s.t. & x(S) & \geq & v(S) + \epsilon \quad (S \neq \emptyset, N) \\ & & x(N) & = & v(N). \end{array}$$

□

The least core of a game $(N, v)$ tries to satisfy all coalitions $\emptyset \neq S \neq N$ as much as possible. Adding all inequalities $x_i \geq v(i) + \epsilon$ and using $x(N) = v(N)$ yields the upper bound

$$\epsilon \leq \frac{v(N) - \sum_{i \in N} v(i)}{|N|}.$$

Hence $(LC)$ has an optimal value $\epsilon^*$ . Obviously $\epsilon^* \geq 0$ if and only if the core of $(N, v)$ is nonempty. Furthermore, if core$(N, v)$ is nonempty then leastcore$(N, v) \subseteq$ core$(N, v)$.

The *excess* of a coalition $\emptyset \neq S \neq N$ in a game $(N, v)$ with respect to an allocation $x \in \mathbb{R}^N$ is defined as

$$e(S, x) := x(S) - v(S).$$

The excess $e(S, x)$ can be seen as a measure of satisfaction of $S$ with respect to the allocation $x$. If $e(S, x) < e(T, x)$ then coalition $S$ will be less satisfied with allocation $x$ than coalition $T$.

Least core allocations are just those allocations that maximize the minimal excess $e_{min}(x) := \min\{e(S, x) \mid \emptyset \neq S \neq N\}$:

$$\text{leastcore}(N, v) = \{x \in \mathbb{R}^N \mid x(N) = v(N), e_{min}(x) = \epsilon^*\}.$$

## 2.3 The nucleolus

If the least core is not yet a single point, one might try to find "the best" allocation in the least core by further pursuing the idea of maximizing minimum excess: After satisfying the coalitions with the smallest excess as much as possible, one tries to satisfy coalitions with the second smallest excess as much as possible and so on.

Given an allocation $x \in \mathbb{R}^N$, we define the *excess vector* $\theta(x) \in \mathbb{R}^{2^N-2}$ by ordering the $2^N - 2$ excess values $e(S, x)$ in a non-decreasing sequence. A vector $x \in \mathbb{R}^m$ is said to be *lexicographically smaller than or equal to* $y \in \mathbb{R}^m$, denoted by $x \preceq y$, if $x = y$ or there exists a number $1 \leq j < n$ such that $x_i = y_i$ if $i \leq j$ and $x_{j+1} < y_{j+1}$.

**Definition 2.3** The *nucleolus* of a game $(N, v)$ is the set of imputations $\{x \in I(N, v) \mid \theta(y) \preceq \theta(x)$ for all $y \in I(N, v)\}$. □

Note that the nucleolus is the set of allocations $x \in \mathbb{R}^N$ that lexicographically maximize $\theta(x)$ over $I(N, v)$. If the set of imputations is empty, then the nucleolus of $(N, v)$ is the empty set. If we lexicographically maximize over the whole set of allocations $I^*(N, v)$, we obtain the *prenucleolus* of $(N, v)$. Both nucleolus and prenucleolus are defined as set valued solution concepts. However Schmeidler [1969] proved the following:

**Theorem 2.1** *Let $S \subseteq \mathbb{R}^N$ be a nonempty convex set. Then the set $\{x \in S \mid \theta(y) \preceq \theta(x)$ for all $y \in S\}$ consists of exactly one point.* □

From this result it follows that the nucleolus prescribes a unique allocation, if $I(N, v)$ is nonempty. In that case we denote the nucleolus of $(N, v)$ by $\eta(N, v)$. By the same result also the prenucleolus, which exists for all games $(N, v)$, is a singleton.

As a solution concept the nucleolus satisfies *Ind*, *Dum*, *Sym* and *Inv*. The prenucleolus only satisfies the last three properties.

It is immediately clear that computing the nucleolus by explicit lexicographic optimization of the excess vector is not efficient: In general there are exponentially (in $|N|$) many different excess values, whereas an efficient procedure should be polynomial in $|N|$. The standard procedure for computing the nucleolus proceeds by solving up to $|N|$ linear programs (cf. Maschler, Peleg and Shapley [1979]). To present it we introduce the following notation: For a polyhedron $P \subseteq \mathbb{R}^N$ let

$$Fix\, P := \{S \subseteq N \mid x(S) = y(S) \text{ for all } x, y \in P\}$$

denote the set of coalitions *fixed* by $P$. We assume that $I(N, v)$ is nonempty and let $\mathcal{F}_0 := \{\emptyset, N\}$. First consider the linear program

$$\begin{array}{llrcll} (P_1) & \max & \epsilon & & & \\ & s.t. & x(S) & \geq & v(S) + \epsilon & (S \notin \mathcal{F}_0) \\ & & x & \in & I(N, v) & \end{array}$$

with optimum value $\epsilon_1 \in \mathbb{R}$. We let $P_1(\epsilon)$ denote the set of all $x \in \mathbb{R}^N$ such that $(x, \epsilon)$ satisfies the constraints of $(P_1)$. So core$(N, v)$= $P_1(0)$. If $\epsilon_1 \geq 0$, then least core$(N, v) = P_1(\epsilon_1)$.

Now, assume we have determined $P_1(\epsilon_1)$. We then proceed to maximize the minimal excess on those coalitions that are not already fixed, i.e., we solve

$$\begin{array}{llrcll} (P_2) & \max & \epsilon & & & \\ & s.t. & x & \in & P_1(\epsilon_1) & \\ & & x(S) & \geq & v(S) + \epsilon & (S \notin Fix\, P_1(\epsilon_1)). \end{array}$$

Clearly $(P_2)$ is bounded and feasible. Hence let $\epsilon_2 > \epsilon_1$ be the optimum value of $(P_2)$. Extending our previous notation in the obvious way, we let $P_2(\epsilon)$

denote the set of all $x \in \mathbb{R}^N$ satisfying the constraints of $(P_2)$ for $\epsilon \in \mathbb{R}$. Now proceed to

$$\begin{array}{llrcll} (P_3) & \max & \epsilon & & & \\ & s.t. & x & \in & P_2(\epsilon_2) & \\ & & x(S) & \geq & v(S)+\epsilon & (S \notin Fix\, P_2(\epsilon_2)) \end{array}$$

etc. until

$$\begin{array}{llrcll} (P_r) & \max & \epsilon & & & \\ & s.t. & x & \in & P_{r-1}(\epsilon_{r-1}) & \\ & & x(S) & \geq & v(S)+\epsilon & (S \notin Fix\, P_{r-1}(\epsilon_{r-1})) \end{array}$$

defines a unique solution $x^* \in \mathbb{R}^N$, which is equal to $\eta(N, v)$, the nucleolus of the game. We have obtained this allocation after computing

$$\epsilon_1 < \epsilon_2 < \ldots < \epsilon_r$$

and $$P_1(\epsilon_1) \subset P_2(\epsilon_2) \subset \ldots \subset P_r(\epsilon_r) = \eta(N, v).$$

The same procedure can be applied to compute the prenucleolus, for which we only have to replace the constraint $x \in I(N, v)$ by $x \in I^*(N, v)$ in the linear program $(P_1)$. If core$(N, v)$ is nonempty then $\epsilon_1 \geq 0$ and in that case the nucleolus and the prenucleolus coincide.

We see that in each iteration (implicit) equality constraints are added that are independent of previous equality constraints. This implies that the feasible regions of the above sequence of LP's decrease in dimension. Hence we conclude that $r \leq |N|$. So we compute at most $|N|$ different excess values explicitly. Note, however, that in each step we have to identify the set $Fix\, P_i(\epsilon_i)$. Furthermore, the number of constraints in each $(P_i)$ remains exponential in $|N|$.

The above "Linear Programming approach" to the nucleolus is also interesting from a structural point of view, as it implies a nice bound on the size $< \eta(N, v) >$ of the nucleolus.

**Theorem 2.2** *The nucleolus of a game $(N, v)$ has size bounded polynomially in $< N, v >$.*

**Proof**: Let $\mathcal{F}_0 \subset \ldots \subset \mathcal{F}_{r-1} \subseteq 2^N$ denote the increasing sequence of fixed sets in $(P_1), \ldots, (P_r)$, i.e., $\mathcal{F}_0 = \{\emptyset, N\}$ and

$$\mathcal{F}_i := Fix\, P_i(\epsilon_i) \qquad (i = 1, \ldots, r-1).$$

Let the polyhedron $P^* \subseteq \mathbb{R}^{r+N}$ be defined by

$$\begin{array}{lcll} x(N) & = & v(N) & \\ x_i & \geq & v(i) & (i \in N) \\ x(S) & \geq & v(S) + \tilde{\epsilon}_1 & (S \notin \mathcal{F}_0) \\ x(S) & \geq & v(S) + \tilde{\epsilon}_2 & (S \notin \mathcal{F}_1) \\ & & \vdots & \\ x(S) & \geq & v(S) + \tilde{\epsilon}_r & (S \notin \mathcal{F}_{r-1}). \end{array}$$

Then it is clear that
$\max\{\tilde{\epsilon}_1 \,|(\tilde{\epsilon}, x) \in P^*\}$ yields $\tilde{\epsilon}_1 = \epsilon_1$,
$\max\{\tilde{\epsilon}_2 \mid (\tilde{\epsilon}, x) \in P^*,\ \tilde{\epsilon}_1 = \epsilon_1\}$ yields $\tilde{\epsilon}_2 = \epsilon_2$ etc. until
$\max\{\tilde{\epsilon}_r \mid (\tilde{\epsilon}, x) \in P^*,\ \tilde{\epsilon}_1 = \epsilon_1, \ldots, \tilde{\epsilon}_{r-1} = \epsilon_{r-1}\}$ has only one solution namely

$$(\epsilon_1, \ldots, \epsilon_r, x^*),$$

where $x^*$ is the nucleolus of $(N, v)$ and $\epsilon_1, \ldots, \epsilon_r$ are the optimum values of $(P_1), \ldots, (P_r)$.

From the above it is straightforward to see that $(\epsilon_1, \ldots, \epsilon_r, x^*)$ cannot be written as a convex combination of other points in $P^*$. Hence $(\epsilon_1, \ldots, \epsilon_r, x^*)$ is a vertex of $P^*$. As such its size is polynomial in the dimension $r + |N| = \mathcal{O}(|N|)$ and the facet complexity (cf. Theorem 1.1). The latter is polynomially bounded by $< N, v >$. □

**Remark 2.1** From the proof of Theorem 2.2 it is also clear that the size of the parameters $\epsilon_i$ $(i = 1, \ldots, r)$ is polynomially bounded in $< N, v >$. □

So if we choose to compute the nucleolus of a game $(N, v)$ by using this algorithmic procedure, then the only difficulties are

(i) identifying the sets $Fix\, P_i(\epsilon_i)$ in each iteration step;

(ii) the exponential number of constraints in each $(P_i)$.

In general these difficulties turn out to be hard. No polynomial time algorithms are known for computing the nucleolus in general. For instance, computing the nucleolus of minimum cost spanning tree games is $\mathcal{NP}$-hard (Faigle, Kern and Kuipers [1998a]). Granot, Granot and Zhu [1998] study the complexity of the nucleolus in general. Several (not efficient) algorithms for

computing the nucleolus in general have been developed (see, e.g., Potters, Reijnierse and Ansing [1996]). Positive results are known for some particular classes of games. For instance, there exist efficient algorithms for computing the nucleolus of standard tree games (Megiddo [1978], Granot, Maschler, Owen and Zhu [1996]), the nucleolus of convex games (Kuipers [1996]) and the nucleolus of assignment games(Solymosi and Raghavan [1994]). Furthermore, in Chapter 4 we will show that for a subclass of matching games we can describe the polyhedra ($P_i$) by means of a polynomial number of inequalities. Also for this class of games the nucleolus can be computed efficiently.

## 2.4 The $f$-least core and $f$-nucleolus

In the literature, core relaxations different from the additive $\epsilon$-core have been studied. Faigle and Kern [1993] propose the *multiplicative $\epsilon$-core* of a game $(N, v)$. Here, the $\epsilon$-correction is directly related to the value v(S). For a given $\epsilon \in \mathbb{R}$, the multiplicative $\epsilon$-core of $(N, v)$ is the set of allocations

$$\{x \in \mathbb{R}^N \mid x(N) = v(N), x(S) \geq v(S) + \epsilon v(S) \text{ for all } \emptyset \neq S \neq N\}.$$

Also for the multiplicative $\epsilon$-core of a *profit game* $(N, v)$ there exists an $\epsilon \in \mathbb{R}$ such that the multiplicative $\epsilon$-core is nonempty. For a cost game this does not need to be true. Consider for example a two-person game $(N, c)$ with $N = \{1, 2\}$ and $c$ given by $c(1) = c(2) = 0$ and $c(N) = 1$. For all $\epsilon \in \mathbb{R}$ the multiplicative $\epsilon$-core is the set $\{x \in \mathbb{R}^2 \mid x_1 + x_2 = 1,\ x_1, x_2 \leq 0\}$, which is empty.

The *multiplicative least core* is defined in the same way as the classical least core, and we generalize as follows:

**Definition 2.4** Let $f : 2^N \to \mathbb{R}_+$. Then the *$f$-least core* of a game $(N, v)$ is the set of allocation vectors that are optimal solutions of the linear program

$$\begin{array}{lllll} (f\text{-}LC) & \max & \epsilon & & \\ & s.t. & x(S) & \geq & v(S) + \epsilon f(S) \quad (S \neq \emptyset, N) \\ & & x(N) & = & v(N). \end{array}$$

Denote this set by $f$-leastcore$(N, v)$.

If $(f\text{-}LC)$ is unbounded, then $f$-leastcore $(N, v)$ is defined to be equal to core$(N, v)$. If $(f\text{-}LC)$ is infeasible, then $f$-leastcore $(N, v)$ is the empty set. □

Obviously, the larger $f(S)$ is for some coalition $S \subset N$, the more decisive $S$ is for determining the optimum value of ($f$-$LC$). We therefore call a function $f$ as above a *priority function*. Note that $f \equiv 1$ corresponds with the classical least core and $f(S) = v(S)$ for all $S \subseteq N$ corresponds with the multiplicative least core. In case of a cost game a priority function $f$ is closely related to the concept of a *taxation function*: Shapley and Shubik [1966] define $f(S) = |S|$ for all $S \subseteq N$ and Tijs and Driessen [1986] propose the *least-tax core*, where $f(S) = v(S) - \sum_{i \in S} v(i)$ for all $S \subseteq N$. Note that for a *zero-normalized* game$(N, v)$ (a game where $v(i) = 0$ for all players $i \in N$) the least-tax core of $(N, v)$ is equivalent to the multiplicative least core of $(N, v)$.

Motivated by the examples above (and also for computational reasons) we mainly restrict our attention to priority functions $f$ that satisfy the following conditions:

(**f1**) $f$ depends only on the size and the value of a coalition, i.e., $f$ is of the type $\hat{f} : \mathbb{N} \times \mathbb{R} \to \mathbb{R}_+$. For $\emptyset \neq S \neq N$ we set $f(S) = \hat{f}(|S|, v(S))$.

(**f2**) For all $\emptyset \neq S \neq N$, $f(S)$ can be computed in time bounded by a polynomial in $< N, v >$.

Because the empty coalition and the grand coalition $N$ have already fixed payoffs, we can restrict a priority function $f : 2^N \to \mathbb{R}_+$ to the set $2^N \backslash \{\emptyset, N\}$.

The next example shows that different $f$-least cores can prescribe different allocations. We compute the additive and multiplicative least core for a simple class of games.

**Example 2.4** Suppose we have a situation where a number of persons want to make a large profit $v^*$. They can only obtain this profit if they all make a joint investment, where every person $i$ invests an amount $v(i)$. We model this situation by an *investment game* $(N, v)$, where $v$ is given by $v(S) := \sum_{i \in S} v(i)$ for all $S \subset N$ and $v(N) = v^* \geq \sum_{i \in N} v(i)$. It is straightforward to check that the additive least core yields the unique allocation $\hat{x} \in \mathbb{R}^N$ given by

$$\hat{x}_i = v(i) + \frac{v^* - \sum_{i \in N} v(i)}{|N|},$$

while the multiplicative least core prescribes the unique allocation $\tilde{x} \in \mathbb{R}^N$ given by

$$\tilde{x}_i = v^* \frac{v(i)}{\sum_{i \in N} v(i)}.$$

If players are payed according to $\hat{x}$, then each player receives his own investment plus an equal share of the amount that is left. If allocation vector $\tilde{x}$ is used, then they receive a pay-off relative to their investments. The last rule seems to be more appropriate. In that case, for example, a player $i \in N$ with investment $v(i) = 0$ is not paid anything. □

If $f(S) > 0$ for some coalition $S \subset N$, then we have an upper bound

$$\epsilon \leq \frac{v(N) - v(S) - v(N\backslash S)}{f(S) + f(N\backslash S)}.$$

Hence ($f$-*LC*) is unbounded if and only if $f \equiv 0$ on $2^N\backslash\{\emptyset, N\}$. Now assume ($f$-*LC*) to be feasible and bounded and let $\epsilon_f^*$ be the optimal value of ($f$-*LC*). Then $\epsilon_f^* \geq 0$ if and only if core$(N, v)$ is nonempty and the following proposition is obvious.

**Proposition 2.1** *If core$(N, v) \neq \emptyset$, then*

$$\emptyset \neq f\text{-leastcore}(N, v) \subseteq \text{core}(N, v) \quad \text{for all } f : 2^N \to \mathbb{R}_+.$$

□

We define the $f$-*excess* of a nonempty coalition $S \subset N$ with respect to a vector $x \in \mathbb{R}^N$ as the number

$$e^f(S, x) \quad = \quad \begin{cases} \dfrac{x(S) - v(S)}{f(S)} & \text{if } f(S) > 0 \\ \infty & \text{if } f(S) = 0 \text{ and } x(S) \geq v(S) \\ -\infty & \text{otherwise.} \end{cases}$$

Let $e_{min}^f(x) := \min\{e^f(S, x) \mid \emptyset \neq S \neq N\}$. If ($f$-*LC*) has an optimal value $\epsilon_f^*$, then $f$-leastcore$(N, v)$ can also be formulated as

$$f\text{-leastcore}(N, v) = \{x \in \mathbb{R}^N \mid x(N) = v(N), e_{min}^f(x) = \epsilon_f^*\}.$$

In general, computing an allocation in the $f$-least core of a game $(N, v)$ implies solving an exponential number of inequalities. However we can obtain the following result (cf. also Faigle, Kern and Kuipers [1998b]).

**Theorem 2.3** *Let $(N, v)$ be a cooperative game, and $f : 2^N \to \mathbb{R}_+$ be a priority function. Suppose that, for an allocation $x \in \mathbb{R}^N$, a coalition $\emptyset \neq S \neq N$*

*with $e^f(S,x) = e^f_{min}(x)$ can be computed in time bounded by a polynomial in $< N, v >$ and $< x >$. Then an allocation in $f$-leastcore$(N,v)$ can be computed efficiently.*

**Proof**: Let $P_f \subseteq \mathbb{R}^{N+1}$ denote the polyhedron of feasible solutions $(x,\epsilon)$ for $(f\text{-}LC)$. We solve the separation problem for $P_f$ as follows: Given $(\tilde{x},\tilde{\epsilon})$, we first check whether $\tilde{x}(N) = v(N)$ is satisfied. Next we compute a coalition $\emptyset \neq S \neq N$ such that $e^f(S,\tilde{x}) = e^f_{min}(\tilde{x})$. If $e^f_{min}(\tilde{x}) \geq \tilde{\epsilon}$, then $(\tilde{x},\tilde{\epsilon})$ is feasible. If this is not true, our separating hyperplane is

$$\{(x,\epsilon) \in \mathbb{R}^{N+1} \mid x(S) - f(S)\epsilon = v(S)\}.$$

By our assumption these computations can be done in time polynomially bounded in $< N, v >$ and $< \tilde{x} >$. Furthermore, $P_f$ has facet complexity at most $< N, v > + < f >$, where

$$< f >= \max\{< f(S) > \mid \emptyset \neq S \neq N\}.$$

By (f2), $< f >$ is polynomially bounded in $< N, v >$. Then the result follows directly from Theorem 1.3. □

If the condition in Theorem 2.3 is satisfied, then we can compute the optimal value $\epsilon^*_f$ of $(f\text{-}LC)$ in polynomial time. If $\epsilon^*_f$ is positive, then the core is nonempty. Otherwise the game $(N,v)$ has an empty core. So, as a direct consequence of Theorem 2.3, we can efficiently check whether core$(N,v)$ is empty or not.

**Example 2.5** (see also Faigle, Kern and Kuipers [1998b]) A *convex game* is a cost game $(N,c)$ whose characteristic function $c$ is *submodular*, i.e.,

$$c(S \cup T) + c(S \cap T) \leq c(S) + c(T) \quad \text{for all } S, T \subseteq N.$$

Let $f : 2^N \to \mathbb{R}_+$ be a priority function. In case of a cost game the $f$-least core is defined as the set of optimal solutions of the linear program

$$\begin{array}{lllll} (f\text{-}LC) & \max & \epsilon & & \\ & s.t. & x(S) & \leq & c(S) - \epsilon f(S) \quad (S \neq \emptyset, N) \\ & & x(N) & = & c(N), \end{array}$$

and the $f$-excess of a nonempty coalition $S \subset N$ for a given vector $x \in \mathbb{R}^N$ is given by

$$e^f(S,x) \;=\; \begin{cases} \dfrac{c(S) - x(S)}{f(S)} & \text{if } f(S) > 0 \\ \infty & \text{if } f(S) = 0 \text{ and } x(S) \leq c(S) \\ -\infty & \text{otherwise.} \end{cases}$$

For $f \equiv 1$ it is straightforward to check that the excess function

$$e(., x) : 2^N \backslash \{\emptyset, N\} \to \mathbb{R}$$

is also submodular. From a standard result in discrete optimization on minimizing a submodular function (cf. Grötschel, Lovász and Schrijver [1993]) it follows that computing a coalition $S$ such that $e(S, x) = e_{min}(x)$ can be done in polynomial time. Hence, by Theorem 2.3 we conclude that we can efficiently compute an element in the least core of a convex game $(N, c)$. □

Analogously to the introduction of the $f$-least core for a given priority function $f : 2^N \to \mathbb{R}_+$ we can extend the notion of the classical nucleolus to the $f$-nucleolus. Order the $2^N - 2$ $f$-excess values $e^f(S, x)$ in a non-decreasing sequence resulting in the *$f$-excess vector* $\theta^f(x)$. The *$f$-nucleolus* of $(N, v)$ is then defined to be the set of all imputations $x \in \mathbb{R}^N$ that lexicographically maximize the excess vector $\theta^f(x)$.

**Definition 2.5** The *$f$-nucleolus* of a game $(N, v)$ is the set of imputations $\{x \in I(N, v) \mid \theta^f(y) \preceq \theta^f(x) \text{ for all } y \in I(N, v)\}$. □

Note that the $f$-nucleolus for $f \equiv 1$ corresponds with the (classical) nucleolus. In the literature the following examples have already been introduced: If $f$ is given by $f(S) = v(S)$ for all $S \subset N$, then the $f$-nucleolus is called the *nucleon* (Faigle, Kern, Fekete and Hochstättler [1998]). If $f$ is given by $f(S) = |S|$ for all $S \subset N$, then the $f$-nucleolus is called the *per-capita nucleolus* (see, e.g., Young, Okada and Hashimoto [1982]). If $f$ only depends on the size of a coalition, i.e., $f(S) = f(T)$ if $|S| = |T|$ for all coalitions $S, T \subset N$, then the $f$-nucleolus coincides with the $f$-nucleolus of Wallmeier [1983].

Contrary to the nucleolus, the $f$-nucleolus of a game $(N, v)$ for some priority function $f \not\equiv 1$ does not necessarily consist of a single element. The following example illustrates this for the nucleon.

**Example 2.6** Let $(N, v)$ be a two-person game, where $N = \{1, 2\}$ and $v$ is given by $v(N) = 1$ and $v(1) = v(2) = 0$. Then the nucleon of $(N, v)$ is the set

$$\{x \in \mathbb{R}^2 \mid x_1 + x_2 = 1, x_1, x_2 \geq 0\}.$$

□

The algorithmic procedure for the nucleolus can also be applied to the general $f$-nucleolus. We assume $I(N,v)$ to be nonempty and let $\mathcal{F}_0 := \{\emptyset, N\}$. For a given priority function $f : 2^N \to \mathbb{R}_+$ we consider the linear program

$$\begin{array}{lllll} (P_1^f) & \max & \epsilon & & \\ & s.t. & x(S) & \geq & v(S)+\epsilon f(S) \quad (S \notin \mathcal{F}_0) \\ & & x & \in & I(N,v). \end{array}$$

If $(P_1^f)$ has an empty feasible solution set, then the $f$-nucleolus of $(N,v)$ is empty. Otherwise let $P_1^f(\epsilon)$ denote the set of all $x \in \mathbb{R}^N$ such that $(x,\epsilon)$ satisfies the constraints of $(P_1^f)$. If $(P_1^f)$ is unbounded, then $f(S) \equiv 0$ and the $f$-nucleolus of $(N,v)$ is equal to core$(N,v)$. Otherwise let $\epsilon_1^f \in \mathbb{R}$ denote the optimum value of $(P_1^f)$. Note that core$(N,v)$= $P_1^f(0)$. Furthermore, if core$(N,v)$ is nonempty, then $\epsilon_1^f \geq 0$, $f$-leastcore$(N,v) = P_1^f(\epsilon_1^f)$ and, as we shall see, the $f$-nucleolus of $(N,v)$ is a subset of core$(N,v)$.

In the next iteration we solve

$$\begin{array}{lllll} (P_2^f) & \max & \epsilon & & \\ & s.t. & x & \in & P_1^f(\epsilon_1^f) \\ & & x(S) & \geq & v(S)+\epsilon f(S) \quad (S \notin Fix\, P_1^f(\epsilon_1^f)). \end{array}$$

If $(P_2^f)$ is unbounded, then the $f$-nucleolus of $(N,v)$ is equal to $P_1^f(\epsilon_1^f)$. Otherwise we have an optimal value $\epsilon_2^f > \epsilon_1^f$ and we proceed to

$$\begin{array}{lllll} (P_3^f) & \max & \epsilon & & \\ & s.t. & x & \in & P_2^f(\epsilon_2^f) \\ & & x(S) & \geq & v(S)+\epsilon f(S) \quad (S \notin Fix\, P_2^f(\epsilon_2^f)) \end{array}$$

etc. until

$$\begin{array}{lllll} (P_{r+1}^f) & \max & \epsilon & & \\ & s.t. & x & \in & P_r^f(\epsilon_r^f) \\ & & x(S) & \geq & v(S)+\epsilon f(S) \quad (S \notin Fix\, P_r^f(\epsilon_r^f)) \end{array}$$

is unbounded and the $f$-nucleolus of $(N,v)$ is equal to the set

$$P_r^f(\epsilon_r^f) \quad (\subset P_{r-1}^f(\epsilon_{r-1}^f) \subset \ldots \subset P_1^f(\epsilon_1^f)\ ).$$

Note that in Example 2.4 it turns out that the nucleon of an investment game $(N,v)$ is equal to its multiplicative least core, which contains exactly one allocation. However, Example 2.6 already shows that in general the $f$-nucleolus

can consist of more than one vector. Obviously, the $f$-nucleolus of $(N, v)$ is a unique imputation if and only if $\{i\} \in Fix\, P_r^f(\epsilon_r^f)$ for all $i \in N$. The following proposition can be of importance for checking this condition.

**Proposition 2.2** *Let $f : 2^N \to \mathbb{R}_+$ be a priority function and let $(N, v)$ be a game with nonempty $f$-nucleolus. If $f(S) > 0$ for a coalition $\emptyset \neq S \neq N$ then, in some stage of the computation procedure for the $f$-nucleolus of $(N, v)$, $S$ will be fixed, i.e., $S \in Fix\, P_r^f(\epsilon_r^f)$ holds for some $r > 0$.*

**Proof**: If $(P_1^f)$ is unbounded then $f \equiv 0$ holds. Assume that the $f$-nucleolus of $(N, v)$ is the set $P_r^f(\epsilon_r^f)$ for some $r > 0$ and $S \notin Fix\, P_r^f(\epsilon_r^f)$ is a coalition with $f(S) > 0$. Let $x$ be an imputation in $P_r^f(\epsilon_r^f)$. With respect to the linear program $(P_{r+1}^f)$ we have

$$v(N) = x(N) = x(S) + x(N\backslash S) \geq v(S) + \epsilon f(S) + x(N\backslash S)$$

implying an upper bound

$$\epsilon \leq \frac{v(N) - v(S) - x(N\backslash S)}{f(S)},$$

a contradiction. □

**Corollary 2.1** *Let $f : 2^N \to \mathbb{R}_+$ be a priority function with $f(i) > 0$ for all $i \in N$. If the $f$-nucleolus of a game $(N, v)$ exists then it is a unique allocation.* □

Examples of $f$-nucleoli satisfying this condition are the (classical) nucleolus and the per-capita nucleolus.

**Corollary 2.2** *Let $f : 2^N \to \mathbb{R}_+$ be a priority function with $f(S) = 0$ if and only if $c(S) = 0$ for all $S \subset N$ and let $(N, c)$ be a cost game with $c(N) > 0$. If the $f$-nucleolus of $(N, c)$ exists then it is a unique allocation.*

**Proof**: If $(P_1^f)$ is unbounded then $f(S) = 0$ and therefore $c(S) = 0$ for all $S \subset N$. Together with $c(N) > 0$ this would imply that the $f$-nucleolus of $(N, v)$ is empty.

Now assume that the $f$-nucleolus of $(N, c)$ is a set $P_r^f(\epsilon_r^f)$ containing more than one vector. Then there exists a player $i \in N$ with $\{i\} \notin Fix\, P_r^f(\epsilon_r)$. By Proposition 2.2, $c(i) = 0$ must hold. Suppose $x \in P_r^f(\epsilon_r^f)$. If $N\backslash i$ is fixed by $P_r^f(\epsilon_r^f)$ then, because $x(N) = c(N)$, $\{i\} \in Fix\, P_r^f(\epsilon_r^f)$, a contradiction. If

$N\backslash i$ is not fixed then, again by Proposition 2.2, $c(N\backslash i) = 0$. By our assumption also $f(i) = f(N\backslash i) = 0$, and we have

$$\begin{aligned} c(N) &= x_i + x(N\backslash i) \\ &\leq c(i) - \epsilon_r^f f(i) + c(N\backslash i) - \epsilon_r^f f(N\backslash i) \\ &= 0, \end{aligned}$$

again a contradiction. □

By this result $f$-nucleoli, such as the nucleon and the $f$-nucleolus with $f(S) = \frac{c(S)}{|S|}$ for all $\emptyset \neq S \neq N$, contain at most one allocation if they are applied to a cost game.

## 2.5 The Shapley value

Recall that a value is a solution concept that prescribes at most one allocation for every game $(N, v)$. Shapley [1953] introduced the following value, which is nonempty for any game $(N, v)$.

**Definition 2.6** The *Shapley value* $\phi(N, v)$ of a game $(N, v)$ is defined as

$$\phi_i(N, v) = \sum_{S \subseteq N\backslash i} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \Big( v(S \cup i) - v(S) \Big) \quad \text{for all } i \in N.$$

□

The Shapley value for a player $i$ can be interpreted as an expected allocation. If $i$ joins a coalition $S$, then this is rewarded with its *marginal contribution* $v(S \cup i) - v(S)$. The probability that $i$ joins a coalition of size $|S|$ is set to $\frac{1}{|N|}$ for all sizes $0 \leq |S| \leq |N| - 1$ and $\binom{|N|-1}{|S|}^{-1}$ for all coalitions of size $|S|$. This results in a final probability equal to $\frac{|S|!(|N|-|S|-1)!}{|N|!}$.

The Shapley value satisfies *Non*, *Sym*, *Dum*, *Add* and *Inv*. Shapley [1953] even proved a stronger result.

**Theorem 2.4** *The Shapley value is the unique value that satisfies Sym, Dum and Add.* □

The Shapley value has been widely studied in the literature (see, e.g., Faigle and Kern [1992], Evans [1996], Driessen and Paulusma [2001], Sprumont [1990]). In general one has to compute all values $v(S)$ to obtain $\phi(N, v)$. Hence for most classes of games computing the Shapley value cannot be done in polynomial time.

The following example shows that $\phi(N, v)$ is not necessarily a core vector for a game $(N, v)$.

**Example 2.7** Consider a 3-person game $(N, v)$, where $N = \{1, 2, 3\}$ and $v$ is given by

$$\begin{array}{lllllllll} v(1) & = & 0 & v(1,2) & = & 2 & v(N) & = & 5. \\ v(2) & = & 0 & v(1,3) & = & 3 & & & \\ v(3) & = & 0 & v(2,3) & = & 4 & & & \end{array}$$

Since $x \in \mathbb{R}^3$ given by $x_1 = 0, x_2 = 2$ and $x_3 = 3$ is easily seen to be a core-vector, core$(N, v)$ is nonempty. Computing the Shapley value yields

$$\phi_1(N, v) = \frac{7}{6},\ \phi_2(N, v) = \frac{10}{6} \text{ and } \phi_3(N, v) = \frac{13}{6}.$$

$\phi(N, v)$ is not in core$(N, v)$, because coalition $\{2, 3\}$ receives $\frac{23}{6}$, which is less than $v(2, 3) = 4$. □

# Chapter 3

# Minimum cost spanning tree games

In a minimum cost spanning tree game the players are represented by nodes in a complete graph and the cost of a coalition is equal to the weight of the corresponding minimum spanning tree. In Section 3.1 we treat some basic theory on minimum spanning trees of a graph. Next we give the definition of a minimum cost spanning tree game.

In Section 3.2 we discuss the core of a minimum cost spanning tree game. Granot and Huberman [1981] prove that these games have a nonempty core by showing that certain vectors are core members. However, these allocations may not be acceptable from a modeling point of view, and Granot and Huberman [1984] present some ways to construct other core allocations from these vectors.

In Section 3.3 we study the $f$-least core of a minimum cost spanning tree game for various priority functions $f : 2^N \to \mathbb{R}_+$. This is a more general approach than the approach followed by Granot and Huberman [1984]. We prove that for a large class of priority functions $f$ computing an allocation in the $f$-least core of a general minimum cost spanning tree game is $\mathcal{NP}$-hard. As a consequence also computing $f$-nucleoli, such as the nucleolus, the nucleon and the per-capita nucleolus of minimum cost spanning tree games, is in general $\mathcal{NP}$-hard.

# 3.1 Introduction

## 3.1.1 Minimum spanning trees

Suppose an electricity network is to be built connecting a number of households to a common power station. Installing an electricity cable between any two households and between any household and the power station is possible, but will cost a certain amount. Then the first task is to construct a network that connects every household to the power station and that has minimal cost. This example belongs to a class of problems where a number of users must be connected to a common supplier, and it can be modeled as a minimum spanning tree problem.

**Definition 3.1** *Let $G = (V, E)$ be a connected graph with a positive weight function $w \geq 0$ defined on the edge set $E$. Then a minimum spanning tree (MST) is a spanning tree $T^*$ of $G$ that has minimal weight, i.e.,*

$$w(E(T^*)) = \min\{w(E(T)) \mid T \text{ is a spanning tree of } G\}.$$

□

Computing an MST of a graph $G = (V, E)$ can be done in polynomial time by, for instance, the algorithm of Kruskal [1956] or the algorithm of Prim [1957]. In the first algorithm an edge with minimal weight is chosen and afterwards edges with weight as small as possible are added as long as no cycle occurs. In the end an MST has been constructed.

---

**Algorithm of Kruskal**

**(1)** Set $E' := \emptyset$.

**(2)** IF $|E'| = |V| - 1$ THEN output $T = (V, E')$. STOP.

**(3)** Choose an edge $e' \in E \backslash E'$ such that

$$w(e') = \min\{w(e) \mid e \in E \backslash E' \text{ and } (V, E' \cup e) \text{ does not contain a cycle}\}.$$

**(4)** Set $E' := E' \cup \{e\}$. GOTO (2).

---

The second algorithm starts with an arbitrary node in $V$. From this node it constructs a tree with minimal weight that will be extended node by node until it spans the whole graph.

---

**Algorithm of Prim**:

**(1)** Set $V' := \{v\}$ for some $v \in V$ and set $E' := \emptyset$.

**(2)** IF $V' = V$ THEN output $T = (V, E')$. STOP.

**(3)** Choose an edge $(i, j) \in E$ with $i \in V'$ and $j \in V \backslash V'$ such that

$$w(i, j) = \min\{w(k, l) \mid (k, l) \in E,\ k \in V', l \in V \backslash V'\}.$$

**(4)** Set $V' := V' \cup \{j\}$ and $E' := E' \cup (i, j)$. GOTO (2).

---

**Example 3.1** Consider a complete graph $G = (V, E)$ on five nodes with weight function $w : E \to \mathbb{R}_+$ as indicated in Figure 3.1. The tree $T$ with edges

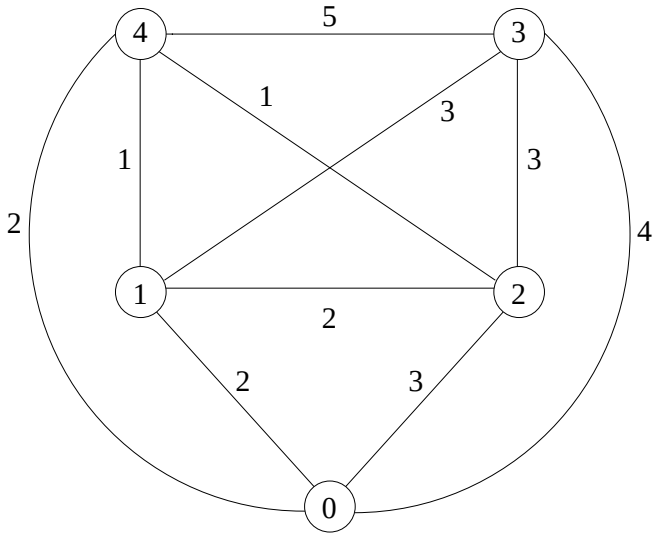


Figure 3.1

$(0, 1)$, $(1, 3)$, $(1, 4)$ and $(2, 4)$ and weight $w(T) = 7$ is easily seen to be an MST of $G$. This MST is not unique: For instance, the tree $T'$ with edges $(0, 4)$, $(1, 4)$, $(2, 3)$ and $(2, 4)$ is also an MST of $G$. □

The following proposition (see, e.g., Aarts [1994]) shows that, given an MST $T$ of a graph $G$, each induced subgraph that is connected has an MST that contains all edges of $T$ with both end nodes in the induced subgraph.

**Proposition 3.1** *Let $G = (V, E)$ be a connected graph. Then for every MST $T$ of $G$ and every set $V' \subseteq V$, for which $G|_{V'}$ is connected, an MST $T'$ of $G|_{V'}$ exists such that*

$$E(T) \cap E(G|_{V'}) \subseteq E(T').$$

**Proof**: Use the algorithm of Kruskal to construct $T$. Suppose that in a certain stage of this algorithm the edge $e' \in E(T) \cap E(G|_{V'})$ is added to $E'$. Since $w(e') = \min\{w(e) \mid e \in E \backslash E'\}$, we have that

$$w(e') = \min\{w(e) \mid e \in E(G|_{V'}) \backslash (E' \cap E(G|_{V'}))\}.$$

Furthermore, $(V, E' \cup e')$, and therefore $(V', (E' \cap E(G|_{V'})) \cup e')$, does not contain any cycle. This means that we can apply the algorithm of Kruskal for the construction of an MST $T'$ of $G|_{V'}$ in such a way that we first choose the edges in $E(T) \cap E(G|_{V'})$. □

### 3.1.2 Minimum cost spanning tree games

Consider again the example of an electricity network. After constructing a network that connects each user to the power station with minimum cost, this cost has to be divided somehow among the users. Such an allocation problem can be modeled as a minimum cost spanning tree game.

**Definition 3.2** A *minimum cost spanning tree game* (MCST-game) $(N, c)$ is determined by a set $N$ of players, a *supply* node $s \notin N$, a complete graph with node set $V = N \cup \{s\}$ and by a weight function $w \geq 0$ defined on its edge set. The cost $c(S)$ of a coalition $S \subseteq N$ is the weight of an MST in the subgraph induced by $S \cup \{s\}$. □

In the definition above we see that $c(N)$ is the weight of an MST in the original graph, which is exactly the minimum total cost of constructing the network. Because we only consider positive weight functions $w$, for any MCST-game $(N, c)$ we have

$$c(S) + c(T) \geq c(S \cup T) \quad \text{for all } S, T \subseteq N, S \cap T = \emptyset.$$

So an MCST-game is *subadditive*, which makes the assumption that all players decide to work together in order to divide the cost $c(N)$ more likely.

The underlying discrete structure of an MCST-game $(N, c)$ is a complete graph $G$ and an edge weighting $w$. Let $< w >$ denote the maximum size of the edge weights, i.e., $< w >:= \max\{< w(i, j) >\}$. We define $< N, c >= |N| < w >$.

A basic observation is now the following. If a node $i \in N$ occurs as a leaf in some MST $T$ for $G$ and if $e$ is the unique edge in $T$ incident with $i$, then $T\backslash e$ is an MST for the subgraph induced by $V\backslash i$. So in this case we immediately deduce that

$$c(N\backslash i) = c(N) - w(e). \tag{3.1}$$

**Example 3.2** Consider again the graph $G$ in Example 3.1. If we assume node 0 to be the supply node then we obtain an MCST-game $(N, c)$, where $N = \{1, 2, 3, 4\}$ and $c : 2^N \to \mathbb{R}_+$ is given by

$$\begin{array}{llllllll}
c(1) = 2 & c(1,2) = 4 & c(1,2,3) = 7 & c(N) = 7. \\
c(2) = 3 & c(1,3) = 5 & c(1,2,4) = 4 & \\
c(3) = 4 & c(1,4) = 3 & c(1,3,4) = 6 & \\
c(4) = 2 & c(2,3) = 6 & c(2,3,4) = 6 & \\
 & c(2,4) = 4 & & \\
 & c(3,4) = 6 &
\end{array}$$

□

## 3.2 The core of a minimum cost spanning tree game

Minimum cost spanning tree problems have been widely studied in the literature. After their introduction by Bird [1976], various results about the core and nucleolus were established (see, e.g., Aarts [1994], Faigle, Kern and Kuipers [1998a], Granot and Huberman [1981], [1984]).

Granot and Huberman [1981] proved the following theorem, which shows that any MCST-game has a nonempty core and that a core allocation can be found in polynomial time. A core allocation as defined below is called a *standard core allocation*. We denote the standard core allocation corresponding to an MST $T^*$ by $x^*$.

**Theorem 3.1** *Let $T^*$ be a minimum spanning tree belonging to an MCST-game $(N, c)$. Then the vector $x^* \in \mathbb{R}^N$ that allocates to player $i \in N$ the weight of the first edge $i$ encounters on the (unique) path from $i$ to $s$ in $T^*$ is a vertex of core$(N, c)$.* □

However, Granot and Huberman [1981] also point out that standard core allocations may not be acceptable from a modeling point of view. The following example illustrates this.

**Example 3.3** Consider an MCST-game $(N, c)$ obtained from a complete graph $G$ on four nodes with edge weighting $w$ as indicated in Figure 3.2. Clearly,

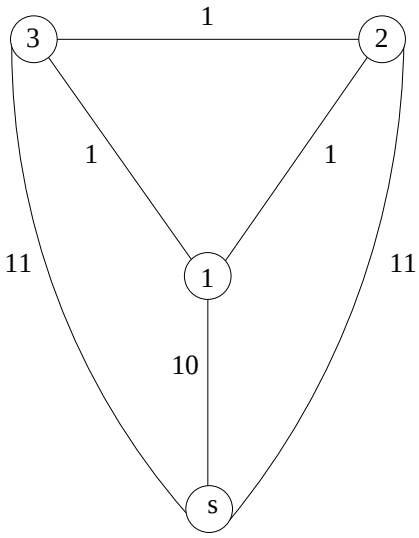


Figure 3.2

the tree $T^*$ with edges $(s, 1)$, $(1, 2)$ and $(1, 3)$ and weight $w(T^*) = 12$ is an MST of $G$. The corresponding standard core allocation $x^*$ is given by $x_1^* = 10$ and $x_2^* = x_3^* = 1$. So most of the cost is charged to player 1, although player 2 and 3 highly depend on this player for their connection to $s$. This motivates the search for other core allocations such as the vector $x$ given by $x_1 = 2$ and $x_2 = x_3 = 5$. □

The question arises how to find possibly more appropriate core allocations and what is the computational complexity of computing these allocations.

Granot and Huberman [1984] propose the following procedure to obtain core allocations not equal to a standard core allocation: Let $T^*$ be a minimum

spanning tree belonging to an MCST-game $(N, c)$ determined by a complete graph $G$ with node set $V = N \cup \{s\}$ and edge weighting $w$. Suppose $i$ is not a leaf of $T^*$. Let $F_i(T^*)$ denote the set of *immediate followers* of node $i$ in $T^*$, i.e.,

$$F_i(T^*) = \{ j \in N \mid i \text{ is the 1st node } j \text{ encounters on the path from } j \text{ to } s \text{ in } T^* \}.$$

For all core allocations $x$ we have $x(N) = c(N)$ and $x(N\backslash i) \leq c(N\backslash i)$. These constraints imply that according to $x^* \in \text{core}(N, c)$ at least an amount of

$$c(N) - c(N\backslash i)$$

is charged to node $i$. If $x_i^* > c(N) - c(N\backslash i)$ then one could try to modify the standard core allocation $x^*$ such that node $i$ pays less, while the immediate followers of $i$ are charged a higher amount for their connection to $s$ via $i$. For this purpose, Granot and Huberman [1984] introduced the so-called *weak demand operation*. This operation can be applied on an arbitrary allocation $x \in \mathbb{R}^N$, but here we show only its effect on standard core allocations. For those allocations the method transfers an amount from $i$ to $F_i(T^*)$ in such a way that the resulting vector is still a core vector and player $i$ is charged exactly $c(N) - c(N\backslash i)$.

In order to explain the method we have to use Proposition 3.1. By this proposition, there exists an MST $T'$ of the subgraph of $G$ induced by $N\backslash i$ such that

$$E(T^*) \cap E(G|_{N\backslash i}) \subseteq E(T'). \tag{3.2}$$

For each $j \in F_i(T^*)$ there exists an edge $e_j$ that is the first edge $j$ encounters on the unique path from $j$ to $s$ in $T'$ that is not an edge in $T^*$. Then the weak demand operation applied in $T^*$ by $i$ on $x^*$ with respect to $T'$ yields the vector $\tilde{x} \in \mathbb{R}^N$ given by

$$\tilde{x}_j \;=\; \begin{cases} c(N) - c(N\backslash i) & \text{if } j = i \\ w(e_j) & \text{if } j \in F_i(T^*) \\ x_j^* & \text{otherwise.} \end{cases}$$

The example below shows that in case $T'$ is not the unique tree satisfying (3.2) the weak demand operation can yield a different vector.

**Example 3.4** Let $(N, c)$ be an MCST-game $(N, c)$ obtained from a complete graph $G$ on five nodes with edge weighting $w$ as indicated in Figure 3.3. The

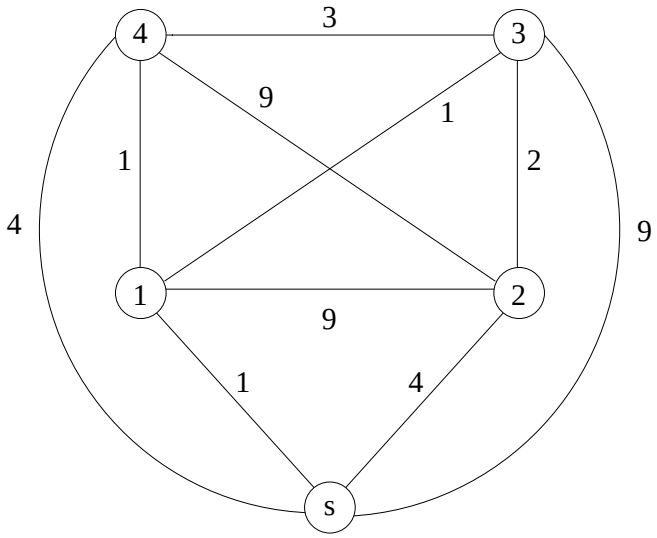


Figure 3.3

tree $T^*$ with edges $(s,1)$, $(1,3)$, $(1,4)$ and $(2,3)$ is an MST of $G$ with weight $w(T^*) = 5$. Then $c(N) = 5$ and $x^* \in \mathbb{R}^4$ is the vector $(1,2,1,1)$.

Node 1 is not a leaf of $T^*$ and its set of immediate followers $F_1(T^*)$ is equal to $\{3,4\}$. The tree $T'$ with edges $(s,2)$, $(2,3)$ and $(3,4)$ is an MST of $G|_{N\setminus 1}$ with weight $w(T') = 9$. So $c(N\setminus 1) = 9$. The weak demand operation by 1 on $x^*$ with respect to $T'$ yields the allocation $\tilde{x}^1 = (-4,2,4,3)$. Also the tree $T''$ with edges $(s,4)$, $(2,3)$ and $(3,4)$ is an MST of $G|_{N\setminus 1}$ satisfying (3.2). Applying the weak demand operation with respect to $T''$ yields the allocation $\tilde{x}^2 = (-4,2,3,4)$. □

Note that in Example 3.4 both $\tilde{x}^1$ and $\tilde{x}^2$ are core allocations. The following result by Granot and Huberman [1984] states that this holds for all vectors obtained after applying a weak demand operation on a standard core allocation.

**Theorem 3.2** *Let $T^*$ be an MST belonging to an MCST-game $(N,c)$ determined by a complete graph $G$ with edge weighting $w$. Let $i \in N$, and let $T'$ be an MST of $G_{N\setminus i}$ such that*

$$E(T^*) \cap E(G|_{N\setminus i}) \subseteq E(T').$$

*Then the vector $\tilde{x} \in \mathbb{R}^N$ obtained by the weak demand operation applied in $T^*$ by $i$ on $x^*$ with respect to $T'$ is an element in core$(N,c)$.* □

**Remark 3.1** From the proof of Proposition 3.1 it is immediately clear that any weak demand operation can be performed in polynomial time. □

The example below shows that applying the weak demand operation on a standard core allocation $x^*$ does not necessarily have to yield a "new" allocation, which is not again a standard core allocation.

**Example 3.5** Let $(N, c)$ be an MCST-game $(N, c)$ obtained from a complete graph $G$ on three nodes with edge weighting $w$ as indicated in Figure 3.4. Both $T_1$ with edge set $E(T_1) = \{(s, 1), (1, 2)\}$ and $T_2$ with edge set $E(T_2) =$

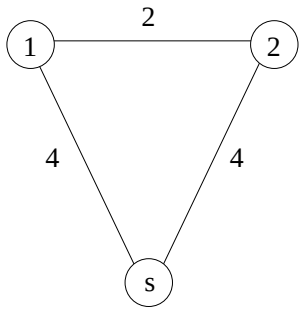


Figure 3.4

$\{(s, 2), (1, 2)\}$ are minimum spanning trees of $G$. The corresponding standard core allocations are $x^1 = (4, 2)$ and $x^2 = (2, 4)$. The weak demand operation by 1 on $x^1$ yields $x^2$ and the weak demand operation by 2 on $x^2$ yields $x^1$. □

Besides a weak demand operation, Granot and Huberman [1984] also introduce the *strong demand operation*: Let $T^*$ be a minimum spanning tree belonging to an MCST-game $(N, c)$. Fix a certain player $i \in N$ that is not a leaf in $T^*$. Then the strong demand operation is a function that assigns to every allocation $x \in \text{core}(N, c)$ the set $S_i(x) \subseteq \text{core}(N, c)$. This set contains all allocations $y \in \text{core}(N, c)$ that can be obtained by transferring the maximal amount from $x_i$ to $F_i(T^*)$ such that the resulting vector is still a core allocation. Clearly, an allocation $\tilde{x}$ obtained after applying a weak demand operation by $i$ on $x^*$ is an element in $S_i(x^*)$. An explicit description of the set $S_i(x)$ is not known for a general MCST-game. However, in case the MST $T^*$ is a path with the supply $s$ as one of its two end points, $S_i(x)$ contains exactly one point, and Granot and Huberman [1984] were able to give an expression for this singleton.

## 3.3 The $f$-least core of a minimum cost spanning tree game

### 3.3.1 Introduction

In the previous section only new core allocations were obtained from standard core allocations by transferring a certain amount from one fixed player to its set of immediate followers. Here we follow a more general approach by considering the $f$-least core of an MCST-game $(N, c)$ for a number of priority functions $f \not\equiv 0$. Recall that the $f$-least core of a cost game $(N, c)$ is the set of allocation vectors that are optimal solutions of the linear program

$$\begin{array}{lllll} (f\text{-}LC) & \max & \epsilon & & \\ & s.t. & x(S) & \leq & c(S) - \epsilon f(S) \quad (S \neq \emptyset, N) \\ & & x(N) & = & c(N). \end{array}$$

Since core$(N, c)$ is nonempty, by Proposition 2.1 the $f$-least core of an arbitrary priority function $f$ is a nonempty subset of core$(N, c)$.

We are interested in the computational complexity of computing an element in the $f$-least core for a priority function $f$. First we show that this approach can be seen as a generalization of the methods described in the previous section.

Let $(N, c)$ be an MCST-game. Assume that a player $i \in N$ is more important than the other players, for instance, by its position in the network. We can express this by a priority function $f^i : 2^N \to \mathbb{R}_+$ given by

$$f^i(S) \;=\; \begin{cases} 1 & \text{if } S = \{i\} \\ 0 & \text{otherwise.} \end{cases}$$

The following proposition gives a characterization of the $f^i$-least core of an MCST-game.

**Proposition 3.2** *Let $(N, c)$ be an MCST-game and $i \in N$. Then*

$$f^i\text{-}leastcore(N, c) = core(N, c) \cap \{x \in \mathbb{R}^N \mid x_i = c(N) - c(N\backslash i)\}.$$

**Proof**: Suppose $x \in f^i$-leastcore$(N, c)$. The feasibility constraints $x(N) = c(N)$, $x(N\backslash i) \leq c(N\backslash i)$ together with $x_i \leq c(i) - \epsilon$ yield the upper bound

$$\epsilon \leq c(i) + c(N\backslash i) - c(N).$$

Let $\epsilon^* = c(i) + c(N\backslash i) - c(N)$. Then $\epsilon^*$ is the optimal value of ( $f^i$-$LC$), if we can show that a feasible solution $(x, \epsilon^*)$ of ( $f^i$-$LC$) exists.

Now let $T^*$ be an MST belonging to $(N, c)$. If $x_i^* = c(N) - c(N\backslash i)$ we have $x_i^* = c(i) - \epsilon^*$ and, since $x^* \in \text{core}(N, c)$, $x^*(S) \leq c(S)$ for all $S \subset N$. Otherwise the vector $\tilde{x}$ obtained after applying a weak demand operation by player $i$ in $T^*$ on $x^*$ satisfies $\tilde{x}_i = c(N) - c(N\backslash i) = c(i) - \epsilon^*$, and by Theorem 3.2 $\tilde{x}$ is a member of core$(N, c)$, and thus $\tilde{x}(S) \leq c(S)$ for all $S \subset N$. Hence $\epsilon^*$ is the optimal value of ( $f^i$-$LC$).

Note that $\epsilon^* \geq 0$. By the feasibility constraints of ( $f^i$-$LC$) an allocation $x$ is an element in $f^i$-leastcore$(N, c)$ if and only if

$$x_i = c(N) - c(N\backslash i) \text{ and } x(S) \leq c(S) \quad \text{for all } \emptyset \neq S \neq N,$$

which proves the proposition. □

From the proof of Proposition 3.2 it is clear that $S_i(x^*) \subseteq f^i$-leastcore$(N, c)$ for any MST $T^*$ belonging to $(N, c)$. In particular an allocation $\tilde{x} \in \mathbb{R}^N$ obtained after applying a weak demand operation by $i$ on $x^*$ is an element of $f^i$-leastcore$(N, c)$. Because such a vector can be computed efficiently (cf. Remark 3.1), the following corollary holds.

**Corollary 3.1** *Let $(N, c)$ be an MCST-game and $i \in N$. Then computing an element in $f^i$-leastcore$(N, c)$ can be done in polynomial time.* □

In the following we will introduce a class of priority functions for which computing the $f$-nucleolus is $\mathcal{NP}$-hard. This class contains both the nucleolus and the nucleon. Below we give an example in which the nucleolus and the nucleon are computed.

**Example 3.6** Consider an MCST-game $(N, c)$ obtained from a complete graph $G = (V, E)$ on three nodes and edge weights as defined in Figure 3.5. The tree $T^*$ with edges $(s, 1)$ and $(1, 2)$ is an MST of $G$ and the standard core allocation $x^*$ is given by $x_1^* = 10$ and $x_2^* = 1$, which may be considered to be unfair with respect to player 1. Solving the linear program

$$\begin{array}{lrcl} \max & \epsilon & & \\ s.t. & x_1 + x_2 & = & 11 \\ & x_1 & \leq & 10 - \epsilon \\ & x_2 & \leq & 11 - \epsilon \end{array}$$

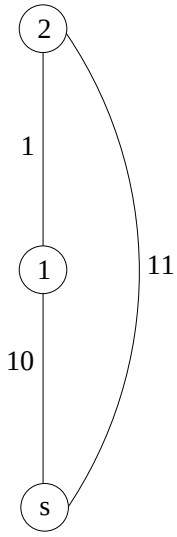


Figure 3.5

yields the unique (additive) least core allocation $x$ given by $x_1 = 5$ and $x_2 = 6$. Solving the linear program

$$\begin{array}{lrcl} \max & \epsilon & & \\ s.t. & x_1 + x_2 & = & 11 \\ & x_1 & \leq & 10 - 10\epsilon \\ & x_2 & \leq & 11 - 11\epsilon \end{array}$$

yields the unique multiplicative least core allocation $y$ given by $y_1 = 5\frac{5}{21}$ and $y_2 = 5\frac{16}{21}$. □

In Example 3.3 both the additive and multiplicative least core are equal to the singleton $\{(0, 6, 6)\}$.

The rest of this chapter is based on Faigle, Kern and Paulusma [2000]. We will define a class of priority functions, for which computing an element in the $f$-least core of an MCST-game $(N, c)$ turns out to be $\mathcal{NP}$-hard. This class contains priority functions $f$ such as

- $f(S) = 1$ for all $\emptyset \neq S \neq N$
- $f(S) = c(S)$ for all $\emptyset \neq S \neq N$
- $f(S) = |S|$ for all $\emptyset \neq S \neq N$.

The proof uses a reduction from minimum cover problems. We show that computing a least core allocation for a special class of graphs introduced in

Faigle, Kern, Fekete and Hochstättler [1997] is already $\mathcal{NP}$-hard. These graphs will be treated in Section 3.3.2. Section 3.3.3 contains the proof of the theorem. In Section 3.3.4 the functions mentioned above are treated. By giving sufficient conditions for a priority function $f$ to satisfy a number of properties defined in Section 3.3.3, we prove that computing an element in the $f$-least core of MCST-games is $\mathcal{NP}$-hard for these functions. Megiddo [1978] and Granot, Maschler, Owen and Zhu [1996] give a polynomial algorithm for computing the nucleolus in case the underlying graph is not a complete graph but a tree. Faigle, Kern and Kuipers [1998a] show that computing the nucleolus of a general MCST-game is $\mathcal{NP}$-hard. Here we obtain this result as an immediate corollary from our main theorem. Also, as a consequence of this result, computing the nucleon and the per-capita nucleolus of MCST-games is in general $\mathcal{NP}$-hard.

### 3.3.2 Minimum cover graphs

In this section we define a minimum cover graph and we show how we can construct an MCST-game from such a graph.

Let $q \in \mathbb{N}$, and let $U$ be a set of $k \geq q$ elements and $W$ be a set of $3q$ elements.

Consider a bipartite graph with node set $U \cup W$ (partitioned into $U$ and $W$) such that each node $u \in U$ is adjacent to exactly three nodes in $W$. We say that node $u \in U$ *covers* its three neighbors in $W$.

A set $D \subseteq U$ is called a *cover* if each $w \in W$ is incident with some $u \in D$. A *minimum cover* is a cover that minimizes $|D|$. Finding a minimum cover is a well-known $\mathcal{NP}$-hard problem. It includes the $\mathcal{NP}$-complete problem X3C (cf. Example 1.2). Below we show that we may restrict ourselves to a subclass of minimum cover problems.

**Proposition 3.3** *Finding a minimum cover under the following assumptions is $\mathcal{NP}$-hard.*

*(***C1***)* *Each node in $W$ has degree 2 or more.*

*(***C2***)* *The size of a minimum cover is at most $q+2$.*

**Proof**: Suppose $w \in W$ is a node with degree 1, $w$ is connected to $u$ and $u$ is also connected to $w_1$ and $w_2$. Add a node $\hat{u}$ to $U$ and connect it to $w$, $w_1$ and $w_2$. The size of a minimum cover will not change. Hence computing the size

of a minimum cover, in case (C1) holds, is at least as hard as computing the size of a minimum cover in the general case.

To show the validity of (C2), add nodes $u_1, u_2, \ldots, u_q$ to $U$ that cover $W$. Each $u_i$ $(i = 1, \ldots, q)$ covers exactly 3 nodes in $W$. Next delete $u_q$. The size of a minimum cover will be less than or equal to $q+2$. If the size is greater than $q$, the original problem has no exact cover. If the size of a minimum cover is equal to $q$, then also delete $u_{q-1}$. Again the size of a minimum cover will be at most $q+2$. If the size is greater than $q$, the original problem has no exact cover. If the size is equal to $q$, also delete $u_{q-2}$ and so on. In each step of the procedure only problems that have a minimum cover with size at most $q+2$ are considered. If $u_1$ would be deleted, one arrives at the original problem. Hence computing the size of a minimum cover, in case (C2) holds, is at least as hard as computing the size of a minimum cover in the general case. □

From now on we assume that conditions (C1) and (C2) hold. We construct an MCST-game from a minimum cover problem as follows (cf. Faigle, Kern, Fekete and Hochstättler [1997]). First we define a *minimum cover graph* $G = (V, E)$. The node set of $G$ consists of $U \cup W$ and three additional nodes: The *Steiner node* $St$, the *guardian* $g$ and the supply $s$. The edge set $E$ of $G$ comprises the following (cf. Figure 3.6).

- all edges $e$ from the bipartite graph on $U \cup W$, each of them having weight $w(e) = q+1$;
- for each $u \in U$, an edge $(u, St)$ between $u$ and $St$ with weight $w(u, St) = q$ and an edge $(u, g)$ between $u$ and $g$ with weight $w(u, g) = q+1$;
- an edge $(St, g)$ between $St$ and $g$ with weight $w(St, g) = q+1$;
- an edge $(g, s)$ between $g$ and $s$ with weight $w(g, s) = 2q-1$.

We extend $G$ to the complete graph $\overline{G}$ on $V$ with weights induced from $G$, i.e., if $e = (i, j)$ is an edge in $\overline{G}$, then $w(i, j)$ is the weight of a shortest path from $i$ to $j$ in $G$.

An MST in $\overline{G}$ is obtained by connecting each $w \in W$ to some $u \in U$ by which it is covered. Such a $u \in U$ exists because each node $w \in W$ has a neighbor in $U$ (indeed, it has at least 2 neighbors in $U$). Then one connects each $u \in U$

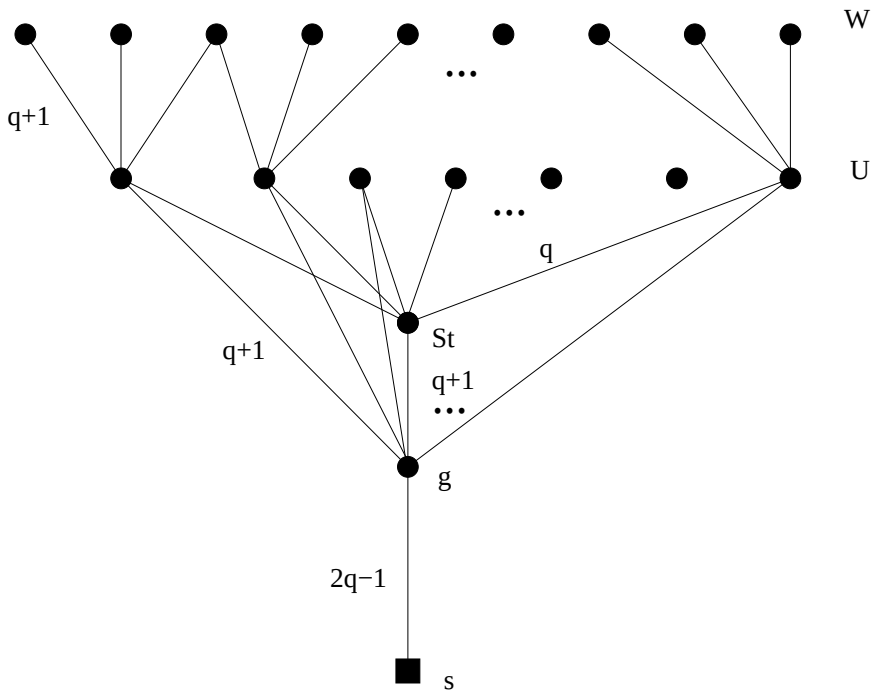


Figure 3.6. a minimum cover graph

to *St*, and finally connects *St* to *g* and *g* to *s*. The resulting MST has a total weight of

$$c(N) \;=\; 3q(q+1) + kq + 3q.$$

Furthermore, by (C1), each $w \in W$ is covered by at least *two* nodes in $U$. Hence it is straightforward to see that the following property holds for $\overline{G}$:

(**L**) For each $v \in U \cup W$, there exists an MST $T$ in the graph $\overline{G}$ such that $v$ is a leaf of $T$.

### 3.3.3 The $f$-least core of minimum cover graphs

Consider a graph $G = (V, E)$ and its completion $\overline{G}$ as described in the previous section. The $f$-leastcore$(N, c)$, relative to a priority function $f : 2^N \to \mathbb{R}_+$, of the corresponding MCST-game consists of all allocation vectors that

are optimal solutions of the linear program

$$\begin{array}{lllll} (f\text{-}LC) & \max & \epsilon & & \\ & s.t. & x(S) & \leq & c(S)-\epsilon f(S) \quad (S\neq\emptyset, N) \\ & & x(N) & = & c(N). \end{array}$$

where $N = V\backslash s$ and $c(S)$ is the weight of an MST in $\overline{G}$ connecting $S$ to the supply $s$.

Suppose $(x,\epsilon)$ is a feasible solution of $(f\text{-}LC)$. Then the feasibility constraints $x(N) = c(N)$ and $x(N\backslash i) \leq c(N\backslash i) - \epsilon f(N\backslash i)$ imply $x(i) \geq c(N) - c(N\backslash i) + \epsilon f(N\backslash i)$ for $i \in N$. Hence, by property (L) of the previous section and (3.1), we have the following inequalities

$$\begin{array}{rcll} x(w) & \geq & q+1+\epsilon f(N\backslash w) & (w\in W) \\ x(u) & \geq & q+\epsilon f(N\backslash u) & (u\in U). \end{array}$$

Furthermore, the coalition $S = N\backslash g$ can be connected to the supply node $s$ at a total cost of $c(N)$. Hence, the feasibility constraints of $(f\text{-}LC)$ also imply

$$x(g) \;\geq\; \epsilon f(N\backslash g)\;.$$

This motivates the following definition.

For $\epsilon > 0$, let $x^\epsilon \in \mathbb{R}^N$ be the allocation defined by

$$\begin{array}{lcll} x^\epsilon(w) & = & q+1+\epsilon f(N\backslash w) & \text{for all } w\in W \\ x^\epsilon(u) & = & q+\epsilon f(N\backslash u) & \text{for all } u\in U \\ x^\epsilon(g) & = & \epsilon f(N\backslash g) & \\ x^\epsilon(St) & = & c(N)-x^\epsilon(U\cup W\cup g). & \end{array}$$

By condition (f1) from Section 2.3, it is straightforward to check that the following parameters do not depend on the particular representative $w \in W$ or $u \in U$:

$$\begin{array}{lcll} f^w & := & f(N\backslash w) & (w\in W) \\ f^u & := & f(N\backslash u) & (u\in U). \end{array}$$

Define for a cover $D \subseteq U$

$$\epsilon^f(D) \;=\; \frac{|D|+2q-1}{|D|\,f^u+3qf^w+f(N\backslash g)+f(D\cup W\cup g)},$$

and let

$$\epsilon^f = \min\{\epsilon^f(D) \mid D \subseteq U \text{ covers } W\}.$$

To make sure that $\epsilon^f$ is finite we assume that $f(S) > 0$ whenever $|S| > 0$ and $c(S) > 0$.

**Remark 3.2** Suppose $D \subseteq U$ is a cover and consider the coalition $S = D \cup W \cup \{g\}$. The cost $c(S)$ is easily seen to be $c(S) = 3q(q+1) + |D|(q+1) + 2q - 1$, i.e., $c(S)$ only depends on $|D|$. Then, by (f1), also $f(S)$ and, therefore $\epsilon^f(D)$ only depends on $|D|$, i.e., satisfies $\epsilon^f(D_1) = \epsilon^f(D_2)$ if $|D_1| = |D_2|$ for all covers $D_1, D_2 \subseteq U$. As a consequence, we can a priori compute all possible values of $\epsilon^f(D)$ for $|D|$ ranging from $q$ to $k$. □

**Lemma 3.1** *If $\epsilon^*$ is the optimal value of ($f$-LC) then $\epsilon^* \leq \epsilon^f$.*

**Proof**: Let $(x, \epsilon^*)$ be an optimal solution of ($f$-LC). As we have seen, the feasibility constraints imply

$$\begin{array}{lcll} x(w) & \geq & q + 1 + \epsilon^* f^w & (w \in W) \\ x(u) & \geq & q + \epsilon^* f^u & (u \in U) \\ x(g) & \geq & \epsilon^* f(N \backslash g). & \end{array}$$

Suppose $D \subseteq U$ is a cover for which $\epsilon^f(D) = \epsilon^f$. Consider the coalition $S = \{g\} \cup D \cup W$. Then

$$x(S) \;\geq\; \epsilon^* f(N \backslash g) + |D|q + \epsilon^* |D| f^u + 3q(q+1) + \epsilon^* 3q f^w$$

whereas,

$$c(S) \;=\; 3q(q+1) + |D|(q+1) + 2q - 1.$$

Since $x(S) \leq c(S) - \epsilon^* f(S)$, we get

$$\epsilon^* \;\leq\; \frac{|D| + 2q - 1}{|D| f^u + 3q f^w + f(N \backslash g) + f(D \cup W \cup g)} \;=\; \epsilon^f \,.$$

□

We call a priority function $f : 2^N \to \mathbb{R}_+$ *feasible* if $f$ satisfies the following properties (with respect to MCST-games on minimum cover graphs):

**(P1)** $\epsilon^f$ is the optimal value of ( $f$-$LC$).

**(P2)** For a cover $D \subseteq U$ of size $q \leq |D| \leq q+2$, we have

$$\epsilon^f = \epsilon^f(D) \text{ if and only if } D \subseteq U \text{ is a minimum cover.}$$

Our main result can be formulated as follows:

**Theorem 3.3** *For the class of feasible priority functions, the problem of computing an allocation vector $x \in$ $f$-leastcore$(N, c)$ of MCST-games is $\mathcal{NP}$-hard.*

**Proof**: Suppose $(x, \epsilon^f)$ is an optimal solution of ( $f$-$LC$). First we will show that for all $w \in W$

$$x(w) \;=\; q+1+\epsilon^f f^w.$$

The feasibility constraints of ( $f$-$LC$) imply

$$\begin{array}{rrll} & x(w) & \geq & q+1+\epsilon^f f^w \quad (w \in W) \\ & x(u) & \geq & q+\epsilon^f f^u \quad\;\; (u \in U) \\ \text{and} & x(g) & \geq & \epsilon^f f(N\backslash g). \end{array}$$

Now let $D \subseteq U$ be a cover for which $\epsilon^f = \epsilon^f(D)$. Consider the coalition $S = \{g\} \cup D \cup W$. Then

$$\begin{aligned} & 3q(q+1+\epsilon^f f^w) + |D|(q+\epsilon^f f^u) + \epsilon^f f(N\backslash g) \\ \leq\;& x(S) \\ \leq\;& c(S) - \epsilon^f f(S) \\ =\;& c(S) - (|D|+2q-1) + (|D|+2q-1) - \epsilon^f f(S) \\ =\;& c(S) - (|D|+2q-1) + \epsilon^f(|D| f^u + 3q f^w + f(N\backslash g) + f(S)) - \epsilon^f f(S) \\ =\;& 3q(q+1+\epsilon^f f^w) + |D|(q+\epsilon^f f^u) + \epsilon^f f(N\backslash g). \end{aligned}$$

Hence $x(S) \leq c(S) - \epsilon^f f(S)$ implies that for all $w \in W$

$$x(w) \;=\; q+1+\epsilon^f f^w.$$

Hence $x \in f$-leastcore$(N, c)$ provides us with the value of the parameter $\epsilon^f$. We can efficiently compute the size $|D|$ of a minimum cover $D \subseteq U$ as follows: Compute $\epsilon^f(D)$ for $|D| = q$, $|D| = q+1$ and $|D| = q+2$ (cf. Remark 3.2). By (C2), it suffices to compute $\epsilon^f(D)$ only for these sizes. By (P2), $\epsilon^f = \epsilon^f(D)$ for at least one of these sizes. Note that a cover $D$ of size $|D| \leq k-2$ implies the existence of covers with size $|D|+1$ and $|D|+2$. Hence, by (P2), the size of a minimum cover $|D|$ will be the maximum of the sizes for which equality holds.

Given an allocation vector $x \in f$-leastcore$(N, c)$, we can thus compute the size of a minimum cover $D$ in polynomial time. Hence the computation of such a vector is at least as hard as the computation of the size of a minimum cover. □

Since the core of an MCST-game $(N, c)$ is nonempty, we have that $\epsilon^*$, the optimal value of $(f\text{-}LC)$, is positive. Then the $f$-nucleolus of $(N, c)$ is a subset of $f$-leastcore$(N, c)$ and the corollary below immediately follows.

**Corollary 3.2** *For the class of feasible priority functions, the problem of computing an allocation vector $x$ in the $f$-nucleolus of an MCST-game $(N, c)$ is $\mathcal{NP}$-hard.* □

The next theorem gives a characterization of the set of feasible priority functions $f : 2^N \to \mathbb{R}_+$.

**Theorem 3.4** *The set of feasible priority functions $f : 2^N \to \mathbb{R}_+$ forms a convex cone (minus $f \equiv 0$).*

**Proof**: It is obvious that $\alpha f$ is feasible for $\alpha > 0$ if $f$ is feasible. Now suppose $f_1,\ f_2 : 2^N \to \mathbb{R}_+$ are feasible. We will show that $f := f_1 + f_2$ is feasible. It is straightforward to verify that for all covers $D \subseteq U$,

$$\epsilon^f(D) = \frac{\epsilon^{f_1}(D)\epsilon^{f_2}(D)}{\epsilon^{f_1}(D) + \epsilon^{f_2}(D)}.$$

First we prove that $f$ satisfies (P2). For a cover $D \subseteq U$ of size $q \leq |D| \leq q+2$, we have

$$\epsilon^f = \epsilon^f(D) \text{ if and only if } D \subseteq U \text{ is a minimum cover.}$$

"$\Rightarrow$" Suppose $|D|$ is not minimum and $\tilde{D} \subseteq U$ is a minimum cover of $W$. Since $f_1$ and $f_2$ satisfy (P2), $\epsilon^{f_1}(\tilde{D}) < \epsilon^{f_1}(D)$ and $\epsilon^{f_2}(\tilde{D}) < \epsilon^{f_2}(D)$. Hence $\epsilon^f(\tilde{D}) < \epsilon^f(D)$, which implies that $\epsilon^f(D)$ is not minimal.

"$\Leftarrow$" Suppose $\tilde{D} \subseteq U$ is a cover and $\epsilon^f(\tilde{D}) < \epsilon^f(D)$. Then at least one of the two inequalities $\epsilon^{f_1}(\tilde{D}) < \epsilon^{f_1}(D)$ and $\epsilon^{f_2}(\tilde{D}) < \epsilon^{f_2}(D)$ must be valid. Hence, by (P2), $D$ is not a minimum cover.

We now show that (P1) holds for $f$. Above we have proved that

$$\epsilon^f = \frac{\epsilon^{f_1}\epsilon^{f_2}}{\epsilon^{f_1} + \epsilon^{f_2}}.$$

By Lemma 3.1 it suffices to show that $\epsilon^f$ is a feasible value for ($f$-$LC$).

Assume that $(x^1, \epsilon^{f_1})$ is an optimal solution of ($f_1$-$LC$), and that $(x^2, \epsilon^{f_2})$ is an optimal solution of ($f_2$-$LC$). We show that $(x, \epsilon^f)$, where $x \in \mathbb{R}^N$ is given by

$$x(i) = \frac{\epsilon^{f_2}x^1(i) + \epsilon^{f_1}x^2(i)}{\epsilon^{f_1} + \epsilon^{f_2}} \quad \text{for all } \ i \in N,$$

is a (feasible) solution of ($f$-$LC$). First note that $x(N) = c(N)$ because $x^1$ and $x^2$ are allocations. Now suppose $\emptyset \neq S \neq N$. Then

$$\begin{aligned} c(S) - \epsilon^f f(S) &= c(S) - \frac{\epsilon^{f_1}\epsilon^{f_2}}{\epsilon^{f_1} + \epsilon^{f_2}}(f_1(S) + f_2(S)) \\ &= \frac{\epsilon^{f_2}(c(S) - \epsilon^{f_1} f_1(S)) + \epsilon^{f_1}(c(S) - \epsilon^{f_2} f_2(S))}{\epsilon^{f_1} + \epsilon^{f_2}} \\ &\geq \frac{\epsilon^{f_2}x^1(S) + \epsilon^{f_1}x^2(S)}{\epsilon^{f_1} + \epsilon^{f_2}} \\ &= x(S). \end{aligned}$$

□

### 3.3.4 Sufficient conditions for $\mathcal{NP}$-hardness

The purpose of this section is to present some conditions for priority functions that are easy to check and imply feasibility. For example, they can be used to prove feasibility of the three specific priority functions mentioned in the introduction of this section. To state our conditions below, we introduce the following notation: A coalition $S \subset N$ is *connected*, if the induced subgraph $G|_S$ is connected.

**Conditions:**

**(S1)** $f^w \leq f^u \leq (1+\frac{1}{q})f^w$.

**(S2)** There exists a number $M \in \mathbb{R}_+$, independent of $q$ and $k$, for which

$$f(S) \leq Mf^w \quad \text{for all } \emptyset \neq S \neq N.$$

**(S3)** For all connected coalitions $S, S' \subset N$ with $|S| > \frac{1}{3}q$ and $0 \leq |S'| - |S| \leq 2$

$$|f(S') - f(S)| \leq \frac{1}{4}f^w.$$

**(S4)** $f(S) > 0$ whenever $|S| > 0$ and $c(S) > 0$.

**Theorem 3.5** *Let the priority function $f : 2^N \to \mathbb{R}_+$ satisfy conditions (S1), (S2), (S3) and (S4). Then $f$ satisfies (P1) and (P2), provided $q$ is sufficiently large.*

**Proof**: Let $D \subseteq U$ be a cover with minimum size. We will prove that $x := x^{\epsilon^f(D)}$ and $\epsilon := \epsilon^f(D)$ are feasible for ($f$-$LC$). By Lemma 3.1 and the definition of $\epsilon^f$, $\epsilon^f = \epsilon^f(D)$ is then the optimal value of ($f$-$LC$). Because $\epsilon^f(D)$ only depends on $|D|$ (cf. Remark 3.2), $D$ can be any minimum cover. In the end we will show that $\epsilon^f < \epsilon^f(D)$ for all covers $D \subseteq U$ that are not minimal.

Let $\emptyset \neq S \neq N$ maximize $\delta(S) := x(S) - c(S) + \epsilon f(S)$. We have to show that $\delta(S) \leq 0$. Suppose $\delta(S) > 0$.

Recall that

$$\begin{array}{lcll} x(w) & = & q+1+\epsilon f^w & \text{for all } w \in W \\ x(u) & = & q+\epsilon f^u & \text{for all } u \in U \\ x(g) & = & \epsilon f(N\backslash g). & \end{array}$$

For the rest of the proof, we need the following relations.

$$\frac{2}{3}\frac{1}{f^w} \leq \epsilon \leq \frac{3}{4}\frac{1}{f^w}. \tag{3.3}$$

$$\text{If } \delta(S) > 0 \text{ , then } |S| > \frac{1}{3}q. \tag{3.4}$$

Proof of (3.3): We have

$$\begin{aligned}
\epsilon &= \frac{|D|+2q-1}{|D|\,f^u+3qf^w+\,f(N\backslash g)+\,f(D\cup W\cup g)} \\
&\leq \frac{|D|+2q-1}{|D|\,f^u+3qf^w} \\
&\leq^{(S1)} \frac{|D|+2q-1}{|D|\,f^w+3qf^w} \\
&\leq^{(C2)} \frac{3q+1}{4q+2}\frac{1}{f^w} \\
&\leq \frac{3}{4}\frac{1}{f^w},
\end{aligned}$$

and

$$\begin{aligned}
\epsilon &= \frac{|D|+2q-1}{|D|\,f^u+3qf^w+\,f(N\backslash g)+\,f(D\cup W\cup g)} \\
&\geq^{(S1),(S2)} \frac{|D|+2q-1}{|D|(1+\frac{1}{q})\,f^w+3qf^w+2Mf^w} \\
&\geq^{(C2)} \frac{|D|+2q-1}{|D|+3q+2M+2}\frac{1}{f^w} \\
&\geq \frac{3q-1}{4q+2M+2}\frac{1}{f^w} \qquad \text{(since } |D|\geq q\text{)} \\
&\geq \frac{2}{3}\frac{1}{f^w} \qquad \text{(for } q \text{ sufficiently large).}
\end{aligned}$$

Proof of (3.4): First we show that $x(St) \leq \frac{1}{3}q$. We have

$$\begin{aligned}
x(St) &= c(N) - x(g) - kx(u) - 3qx(w) \\
&\leq 3q + 3q(q+1) + kq - kq - \epsilon k f^u - 3q(q+1) - \epsilon 3q f^w \\
&= 3q - \epsilon k f^u - \epsilon 3q f^w \\
&\leq 3q - \epsilon q f^u - \epsilon 3q f^w \qquad \text{(since } k \geq q\text{)} \\
&\leq^{(S1)} 3q - \epsilon 4q f^w \\
&\leq^{(3.3)} \tfrac{1}{3}q.
\end{aligned}$$

Hence, in particular, $x(St) \leq q + 1 + \epsilon f^u$. Thus, for $S \subset N$ we have

$$\begin{aligned}
x(S) &\leq^{(S1)} (q + 1 + \epsilon f^u)|S| + \epsilon f(N \backslash g) \\
&\leq^{(S1),(S2)} (q + 1 + \epsilon(1 + \tfrac{1}{q})\, f^w)|S| + \epsilon M f^w \\
&\leq^{(3.3)} (q+2)|S| + M, \\
c(S) &\geq q|S| + q - 1, \quad \text{and} \\
\epsilon f(S) &\leq^{(S2)} \epsilon M f^w \\
&\leq^{(3.3)} M.
\end{aligned}$$

Hence

$$\begin{aligned}
0 &< x(S) - c(S) + \epsilon f(S) \\
&\leq (q+2)|S| + M - q|S| - q + 1 + M \\
&= 2|S| - q + 1 + 2M.
\end{aligned}$$

Then $|S| > \frac{1}{2}q - \frac{1}{2} - M > \frac{1}{3}q$ (for $q$ sufficiently large).

This completes the proof of (3.4). We now continue the proof of the theorem by establishing a sequence of claims.

**Claim (1)**: If $St \in S$ then $|S| < |N| - 1$.

Suppose $S = N \backslash i$ for some $i \in \{g\} \cup U \cup W$. Then, by definition of $x$, $\delta(S) = 0$, a contradiction, since we assume $\delta(S) > 0$.

**Claim (2)**: $St \in S$ or $g \in S$.

Suppose $S \subseteq U \cup W$, $S = S_1 \cup S_2 \cup \ldots \cup S_r$ with $S_i$ $(i = 1, \ldots, r)$ connected. By connecting one subset $S_i$ to $s$ with cost at least $3q + (|S_i| - 1)(q + 1)$ and by connecting the other subsets $S_j$ $(j \neq i)$ to $S_i$ with cost at least $2q + (|S_j| - 1)(q + 1)$, it is clear that

$$\begin{aligned} c(S) &\geq 3q + 2q(r-1) + \sum_{i=1}^{r} (|S_i| - 1)(q+1) \\ &= r(q-1) + q + |S|(q+1). \end{aligned}$$

Furthermore we have

$$\begin{aligned} x(S) &= |S \cap U|\epsilon f^u + |S \cap W|\epsilon f^w + |S \cap U|q + |S \cap W|(q+1) \\ &\leq^{(S1)} |S \cap U|\epsilon(1 + \tfrac{1}{q})\, f^w + |S \cap W|\epsilon f^w + |S \cap U|q + |S \cap W|(q+1) \\ &\leq^{(3.3)} \tfrac{3}{4}|S \cap U|(1 + \tfrac{1}{q}) + \tfrac{3}{4}|S \cap W| + |S \cap U|q + |S \cap W|(q+1) \\ &= \tfrac{4q^2+3q+3}{4q}|S \cap U| + (q + \tfrac{7}{4})|S \cap W|, \end{aligned}$$

and

$$\begin{aligned} \epsilon f(S) &\leq^{(S2)} M\epsilon f^w \\ &\leq^{(3.3)} M. \end{aligned}$$

Since each node in $U$ is adjacent to exactly three nodes in $W$, for $i = 1, \ldots, r$

$$|S_i \cap W| \leq 2|S_i \cap U| + 1.$$

Hence

$$|S \cap U| = \sum_{i=1}^{r} |S_i \cap U| \geq \sum_{i=1}^{r} \frac{|S_i \cap W| - 1}{2} = \frac{1}{2}|S \cap W| - \frac{1}{2}r.$$

Then

$$\begin{aligned}
\delta(S) &= x(S) - c(S) + \epsilon f(S) \\
&\leq \tfrac{4q^2+3q+3}{4q}|S\cap U| + (q+\tfrac{7}{4})|S\cap W| - r(q-1) - q - |S|(q+1) + M \\
&= \tfrac{3}{4}|S\cap W| - \tfrac{q-3}{4q}|S\cap U| - r(q-1) - q + M \\
&\leq \tfrac{3}{4}|S\cap W| + \tfrac{q-3}{4q}(\tfrac{1}{2}r - \tfrac{1}{2}|S\cap W|) - r(q-1) - q + M \\
&\leq M + 2 - \tfrac{1}{8}q \qquad \text{(since } |S\cap W| \leq 3q \text{ and } r \geq 1\text{)} \\
&\leq 0 \qquad \text{(for } q \text{ sufficiently large),}
\end{aligned}$$

a contradiction, since we assume $\delta(S) > 0$. Hence $g \in S$ or $St \in S$.

**Claim (3)**: $S \cap U$ covers $S \cap W$.

Up to now, we have proved that an MST for $S$ looks as follows. Each $u \in S \cap U$ is connected to $g$ (with cost $q+1$) or to $St$ (with cost $q$). Each covered $w \in S \cap W$ is connected to a node $u \in S \cap U$ (with cost $q+1$). Each uncovered $w \in S \cap W$ is without loss of generality joined to $g$ (with cost $2q+2$) or to $St$ (with cost $2q+1$). Now suppose $w \in S \cap W$ is not covered by $S \cap U$. Suppose $w$ is covered by $u(\notin S)$. Then

$$c(S\backslash w \cup u) \leq c(S) - (q+1),$$

and

$$\begin{aligned}
& \delta(S\backslash w \cup u) - \delta(S) \\
= \quad & x(u) - x(w) + c(S) - c(S\backslash w \cup u) + \epsilon(f(S\backslash w \cup u) - f(S)) \\
\geq^{(S2)} \quad & \epsilon(f^u - f^w) + q - \epsilon M f^w \\
\geq^{(S1),(3.3)} \quad & q - \tfrac{3}{4}M \\
> \quad & 0 \qquad \text{(for } q \text{ sufficiently large),}
\end{aligned}$$

contradicting the maximality of $\delta(S)$. Hence $S \cap U$ covers $S \cap W$. In particular, $S$ is connected and, by (3.4), $|S| > \frac{1}{3}q$.

**Claim (4)**: $S$ contains all $w$ covered by $S \cap U$.

Suppose $w \in W$ is covered by $S \cap U$ and $w \notin S$. By claim (1), $|N \backslash S| \geq 2$. Then $S \cup \{w\} \neq N$ and

$$\begin{aligned} \delta(S \cup w) - \delta(S) &= x(w) + c(S) - c(S \cup w) + \epsilon(f(S \cup w) - f(S)) \\ &= \epsilon(f^w + f(S \cup w) - f(S)) \\ &>^{(S3)} 0, \end{aligned}$$

contradicting the maximality of $\delta(S)$.

**Claim (5)**: $St \notin S$.

Suppose $St \in S$. If $S \cap U = U$ then, by claim (4), $S \cap W = W$. Hence $S = N \backslash g$ in contradiction to claim (1). Suppose $u \notin S$. By claim (1), $|N \backslash S| > 1$. Then $S \cup \{u\} \neq N$ and, by claim (3), $c(S \cup u) = c(S) + q$. We have

$$\begin{aligned} \delta(S \cup u) - \delta(S) &= x(u) + c(S) - c(S \cup u) + \epsilon(f(S \cup u) - f(S)) \\ &= \epsilon(f^u + f(S \cup u) - f(S)) \\ &\geq^{(S1)} \epsilon(f^w + f(S \cup u) - f(S)) \\ &>^{(S3)} 0, \end{aligned}$$

contradicting the maximality of $\delta(S)$.

**Claim (6)**: $S \cap W = W$.

Suppose $w \in W \backslash S$. By claim (4), $w$ is not covered by $S \cap U$. Because each node in $W$ has at least two neighbors in $U$, we have $|S \cap U| \leq |U| - 2$. Suppose $w$ is covered by $u (\notin S)$. By claim (2) and claim (5), $g \in S$. Then $c(S \cup u \cup w) = c(S) + 2q + 2$ and

$$\begin{aligned} & \delta(S \cup u \cup w) - \delta(S) \\ = \; & x(u) + x(w) + c(S) - c(S \cup u \cup w) + \epsilon(f(S \cup u \cup w) - f(S)) \\ = \; & -1 + \epsilon(f^u + f^w + f(S \cup u \cup w) - f(S)) \\ >^{(S1),(S3)} \; & -1 + \epsilon \tfrac{3}{2} f^w \\ \geq^{(3.3)} \; & 0, \end{aligned}$$

contradicting the maximality of $\delta(S)$.

**Claim (7)**: $S = \{g\} \cup D \cup W$ for some minimum cover $D \subseteq U$.

Up to now, we have proved that $S = \{g\} \cup D' \cup W$ for some cover $D' \subseteq U$. Suppose $\tilde{D} \subseteq U$ is a cover with $|\tilde{D}| < |D'|$. Without loss of generality we may assume that $|\tilde{D}| = |D'| - 1$ (otherwise add a sufficient number of nodes $u \in U$ to $\tilde{D}$). Let $\tilde{S} = \{g\} \cup \tilde{D} \cup W$. It is obvious that $x(\tilde{S}) = x(S) - q - \epsilon f^u$ and $c(\tilde{S}) = c(S) - q - 1$. Then

$$\begin{aligned} & \delta(\tilde{S}) - \delta(S) \\ = \quad & 1 + \epsilon(f(\tilde{S}) - f(S) - f^u) \\ >^{(S1),(S3)} \quad & 1 - \epsilon\tfrac{4}{3} f^w \\ \geq^{(3.3)} \quad & 0, \end{aligned}$$

contradicting the maximality of $\delta(S)$.

We have proved that

$$S = \{g\} \cup D \cup W \text{ for some minimum cover } D \subseteq U.$$

Then (by definition of $\epsilon$) $\delta(S) = 0$. Hence $(x, \epsilon)$ is a feasible solution of $(f\text{-}LC)$. Because $\epsilon^f(D)$ only depends on $|D|$, $D$ can be any minimum cover. This proves (P1) and the "$\Leftarrow$"-part of (P2).

We complete the proof by showing that $\epsilon^f < \epsilon^f(D)$ for all covers $D \subseteq U$ that are not minimal. We have already proved that for all coalitions $\emptyset \neq T \neq N$

$$\delta(T) \leq \delta(S) = 0,$$

where $S = \{g\} \cup D \cup W$ and $D \subseteq U$ can be any minimum cover. Let $D' \subseteq U$ be a cover that is not minimal and $S' = \{g\} \cup D' \cup W$. In the proof of claim (7) we have shown that there exists a cover $\tilde{D} \subseteq U$ with $|\tilde{D}| = |D'| - 1$ and $\delta(\tilde{S}) > \delta(S')$, where $\tilde{S} = \{g\} \cup \tilde{D} \cup W$. Hence

$$\delta(S') < \delta(\tilde{S}) \leq \delta(S) = 0,$$

which is equivalent to $\epsilon^f < \epsilon^f(D')$. □

Consider again the priority functions mentioned in the introduction:

- $f(S) = 1$ for all $\emptyset \neq S \neq N$
- $f(S) = |S|$ for all $\emptyset \neq S \neq N$
- $f(S) = c(S)$ for all $\emptyset \neq S \neq N$.

It is straightforward to check that these functions satisfy (S1), (S2) and (S4). Furthermore, the first two functions obviously satisfy (S3). In order to show that condition (S3) is also valid for $f$ given by $f(S) = c(S)$ for $\emptyset \neq S \neq N$, we deduce that for a connected coalition $S$ of size $|S| \geq \frac{1}{3}q$

$$c(S) \leq |S|(q+1) + 2q - 1 \leq |S|q + 7|S| \tag{3.5}$$

and

$$c(S) \geq |S|q + q - 1 \geq |S|q. \tag{3.6}$$

By using these two inequalities condition (S3) is easily seen to be satisfied. Hence, by Theorem 3.3 and Theorem 3.5, the problem of computing an allocation vector $x \in f$-leastcore$(N, c)$ of MCST-games is $\mathcal{NP}$-hard for these functions. By Corollary 3.2 computing the corresponding $f$-nucleoli is $\mathcal{NP}$-hard (see also Faigle, Kern and Kuipers [1998a] for $f \equiv 1$).

**Theorem 3.6** *Computing the nucleolus, the nucleon and the per-capita nucleolus of MCST-games is $\mathcal{NP}$-hard.* □

Furthermore, one can verify that functions, such as

- $f(S) = c(S)|S|$ for all $\emptyset \neq S \neq N$
- $f(S) = \dfrac{c(S)}{|S|}$ for all $\emptyset \neq S \neq N$,

satisfy (S1), (S2), (S3) (use (3.5) and (3.6)) and (S4). Hence these functions also belong to the class of feasible priority functions.

We end our discussion by mentioning some priority functions for which our approach does not yield any $\mathcal{NP}$-hardness result. In particular, functions that give high priority to small conditions, such as

- $f(S) = e^{-|S|}$ for all $\emptyset \neq S \neq N$
- $f(S) = \dfrac{1}{|S|}$ for all $\emptyset \neq S \neq N$,

violate conditions (S2) and (S3).

As a generalization of the class of priority functions $f^i$, suppose there is a set $T \subseteq N$ of important individuals with size $|T| \geq 2$. One may then consider the priority function

$$f(S) = \begin{cases} 1 & \text{if } S = \{i\},\ i \in T \\ 0 & \text{otherwise,} \end{cases}$$

which does not satisfy (S4).

Whether the $f$-least core or $f$-nucleolus for any of these functions can be computed efficiently is still an open problem.

# Chapter 4

# Matching games

In a matching game the players are represented by nodes in a certain graph and the profit of a coalition is equal to the weight of the corresponding maximum matching. Section 4.1 contains an introduction into matching theory. We treat the Gallai-Edmonds decomposition, which plays an important role in the rest of this chapter.

In Section 4.2 we consider general matching games. We show that an element in the least core can be computed efficiently. Furthermore, in case the core is nonempty, the nucleolus can be computed efficiently.

In Section 4.3 we restrict ourselves to matching games on a graph, where the weight on an edge is defined as the sum of certain weights defined on the incident nodes. Cardinality matching games belong to this class. We show that the nucleolus can be computed efficiently. This result is based on an alternative characterization of the least core, which may be of independent interest.

## 4.1 Matching theory

In this chapter we shall need some fundamental results and concepts from matching theory, which will be treated in this section. For more on matching theory we refer to the book of Lovász and Plummer [1986]. We start with the following example as given in Shapley and Shubik [1972].

Consider a real estate market. In this market we have a group $N_1$ of home-owners and a group $N_2$ of prospective purchasers. The first group tries to sell

a house, while the second group is interested to buy one. Now suppose $j \in N_2$ is a purchaser, who is interested in the house of person $i \in N_1$. Let $w(i, j)$ denote the difference between the maximum offer of $j$ and the minimum selling price of $i$. If $w(i, j) \geq 0$, person $i$ and $j$ are willing to do business with each other. Then $w(i, j)$ can be interpreted as a "common profit".

We can model this situation by a bipartite graph $G$ with node classes $N_1$ and $N_2$. Between two nodes $i \in N_1$ and $j \in N_2$ an edge $(i, j)$ exists if and only if $w(i, j) \geq 0$. Then the problem of maximizing the total profit comes down to an *assignment problem*, i.e., to find a maximum weight matching in a *bipartite* graph.

We can generalize these kind of problems to nonbipartite graphs. Each $i \in V$ has exactly one good to offer. He can do business with at most one other person. A pair $i, j \in V$, each with their own good to offer, obtains a common profit $w(i, j)$, if they do business with each other. They are only willing to do that, if $w(i, j) \geq 0$. So an edge between $i$ and $j$ exists if and only if $w(i, j) \geq 0$. Take for example a second hand car market, where each person has one car and some money. The problem of maximizing the total profit is called a *maximum weight matching problem.*

**Definition 4.1** Let $G = (V, E)$ be a graph with a weight function $w$ defined on the edge set $E$. Then a *maximum weight matching* is a matching $M^*$ in $G$ that has maximum weight, i.e.,

$$w(M^*) = \max\{w(M) \mid M \subseteq E \text{ is a matching } \}.$$

□

**Example 4.1** Consider the graph $G = (V, E)$ with edge weighting $w$ as indicated in Figure 4.1. The matching $M^* = \{(1, 6), (3, 4), (5, 8)\}$ is a maximum weight matching in $G$ with weight $w(M^*) = 15$. Node 2 and 7 are not covered by $M^*$. The matching $M$ is not the unique maximum weight matching, since the matching $M' = \{1, 5), (3, 4), (6, 8)\}$ also has weight $w(M') = 15$. Note that every *perfect* matching, such as $\tilde{M} = \{(1, 2), (3, 4), (5, 6), (7, 8)\}$, has weight at most $w(M) = 14$. □

As a special case of maximum weight matching problems we introduce the class of *node matching problems*. Let $G = (V, E)$ model the following market situation: Each $i \in V$ has a "weight" $w_i \geq 0$ indicating his importance or

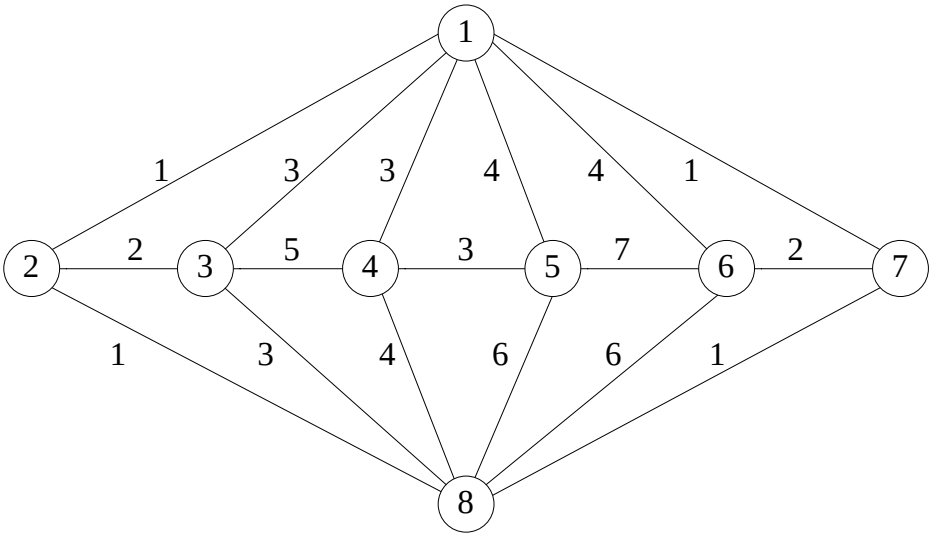


Figure 4.1

power. The edges in $E$ correspond with pairs of potential business partners. Now assume that, if $i$ and $j$ do business with each other, their common profit equals $w(i, j) = w_i + w_j$. If we want to maximize the total profit, then again we have to find a maximum weight matching. In this case we call the problem a node matching problem.

**Definition 4.2** The *node matching problem* is the problem of finding a maximum weight matching in a graph $G = (V, E)$ with edge weighting $w : E \to \mathbb{R}_+$ that can be expressed as the sum of certain positive weights on the incident nodes: There exists a weight function $\bar{w} : V \to \mathbb{R}_+$ on the nodes of $G$ in such a way that

$$w(i, j) = \bar{w}_i + \bar{w}_j \quad \text{for all } (i, j) \in E.$$

□

If no misunderstanding is possible, then we also indicate the node weight function by $w$.

Edmonds [1965a] proved that the maximum weight matching problem can be solved in polynomial time. Before we go into that result, we first restrict ourselves to *cardinality matching problems*.

A *maximum matching* in a graph $G = (V, E)$ is a matching $M^*$ of maximum cardinality, i.e.,

$$|M^*| = \max\{|M| \mid M \subseteq E \text{ is a matching } \}.$$

The problem of finding a maximum matching in a graph is called the *cardinality matching problem*. Note that this problem can been seen as the simplest case of a node matching problem by defining a node weighting $w \equiv \frac{1}{2}$ on $V$.

A *minimum node cover* is a node cover $V^* \subseteq V$ of minimum cardinality, i.e.,

$$|V^*| = \min\{|V'| \mid V' \subseteq V \text{ is a node cover } \}.$$

The following well-known min-max relation, valid for bipartite graphs, is due to König [1931].

**Theorem 4.1** *The size of a maximum matching in a bipartite graph $G = (V, E)$ is equal to the size of a minimum node cover in $G$.* □

Edmonds [1965b] constructed a polynomial time algorithm for solving the cardinality matching problem. The main idea is to find an *augmenting path*.

Let $M$ be a matching in a graph $G = (V, E)$. A path $P = v_0 v_1 \dots v_k$ in $G$ is an *alternating path* (relative to $M$) if for each $i = 1, \dots, k-1$ either $(v_{i-1}, v_i)$ or $(v_i, v_{i+1})$ belongs to $M$. A path $P = v_0 v_1 \dots v_k$ in $G$ is called *augmenting* (relative to $M$), if $P$ is alternating and both nodes $v_0$ and $v_k$ are not covered by $M$. So the edges $(v_0, v_1), (v_2, v_3), \dots, (v_{k-1}, v_k)$ do *not* belong to $M$, while the edges $(v_1, v_2), (v_3, v_4), \dots, (v_{k-2}, v_{k-1})$ belong to $M$. Note that in this case the length of $P$ is odd.

For two subsets $E_1, E_2 \subseteq E$ we let $E_1 \triangle E_2$ denote the set $E_1 \backslash E_2 \cup E_2 \backslash E_1$. It is straightforward to see that, for a matching $M$ and alternating path $P$, $M \triangle P$ is again a matching in $G$. We say that $M \triangle P$ is the matching obtained after *reversing $M$ along $P$*. If $P$ is augmenting, then $|M \triangle P| = |M| + 1$. Furthermore, it is not difficult to prove the following lemma.

**Lemma 4.1** *Let $G = (V, E)$ be a graph and let $M_1, M_2$ be matchings in $G$. Then each component of the graph $G' = (V, M_1 \triangle M_2)$ is either an alternating path (possibly of length 0) or an even cycle.* □

To check whether a matching $M$ is a maximum matching or not, it is sufficient to know whether there exists an augmenting path.

**Theorem 4.2** *Let $M$ be a matching in a graph $G = (V, E)$. Then either $M$ is a maximum matching, or there exists an augmenting path relative to $M$.*

**Proof**: If $M$ is a maximum matching and $P$ is an augmenting path, then $M \triangle P$ would be a larger matching, a contradiction.

If $M$ is not a maximum matching, then there exists a matching $|M'| > |M|$. At least one of the components in $G' = (V, M_1 \triangle M_2)$ contains more edges of $M'$ than of $M$. By Lemma 4.1, such a component must be an augmenting path. □

The running time of the cardinality matching algorithm of Edmonds [1965b] is bounded by $\mathcal{O}(|V|^4)$. Over the years this method has been sharpened (for a survey, see Lovász and Plummer [1986]). Eventually Micali and Vazirani [1980] showed that the running time can be reduced to $\mathcal{O}(|V|^{\frac{1}{2}}|E|)$.

**Theorem 4.3** *Given a graph $G = (V, E)$, a maximum matching can be found in time bounded by $\mathcal{O}(|V|^{\frac{1}{2}}|E|)$.* □

For our main results, as stated in Section 4.3, we make use of the *Gallai-Edmonds decomposition* of a graph. In order to describe this decomposition we need the following basic concepts.

A (*near-*)*perfect matching* is a matching that covers all nodes (except one). A graph is *factor-critical* if removing any node results in a perfectly matchable graph.

Let $V' \subseteq V$. We let $\mathcal{C} = \mathcal{C}(V')$ denote the set of even components of $G \backslash V'$ and $\mathcal{D} = \mathcal{D}(V')$ the set of odd components of $G \backslash V'$. The subset $V' \subseteq V$ is called a *Tutte set*, if each maximum matching $M$ of $G$ decomposes as

$$M = M_{\mathcal{C}} \cup M_{V',\mathcal{D}} \cup M_{\mathcal{D}},$$

where $M_{\mathcal{C}}$ is a perfect matching in $\bigcup \mathcal{C}$, the union of all even components. $M_{\mathcal{D}}$ induces a near-perfect matching in all odd components $D \in \mathcal{D}$ and $M_{V',\mathcal{D}}$ is a matching that matches $V'$ (completely) into $\bigcup \mathcal{D}$, the union of odd components. Note that $M_{V',\mathcal{D}}$ has to match each $i \in V'$ in a different component $D \in \mathcal{D}$.

Equivalently, $V'$ is a Tutte-set if and only if the size $m^*$ of a maximum matching in $G$ equals

$$m^* = \sum_{C \in \mathcal{C}} \frac{|C|}{2} + |V'| + \sum_{D \in \mathcal{D}} \frac{(|D| - 1)}{2}.$$

Tutte sets can be found efficiently. More precisely, the following result, which has been proven by both Gallai [1963], Gallai [1964] and Edmonds [1965b], is true.

**Theorem 4.4** (Gallai-Edmonds Decomposition). *Given $G = (V, E)$, one can efficiently construct a unique Tutte set $T \subseteq V$ such that*

*(i) all odd components $D \in \mathcal{D}$ are factor-critical*

*(ii) for each $D \in \mathcal{D}$ there is some maximum matching that does not completely cover $D$.*

□

Let $G = (V, E)$ be a graph with node set $V$ and edge set $E$. Suppose $T \subseteq V$ is the Tutte set satisfying condition (i) and (ii) of the theorem above. Clearly, a node $v \in V$ is a node in $\bigcup \mathcal{D}$ if and only if there exists a maximum matching in $G$ not covering $v$.

**Remark 4.1** The cardinality matching algorithm of Edmonds [1965b] can be used to construct the Tutte set $T$ as described in Theorem 4.4. Given a graph $G = (V, E)$, we compute the size $m^*$ of a maximum matching in $G$. We next compute for all $i \in V$ the size of a maximum matching in $G \backslash i$. If this size is equal to $m^*$, then $i$ is a node in $\bigcup \mathcal{D}$. Otherwise $i$ is not a node in $\bigcup \mathcal{D}$. Then the subset of $V \backslash V(\bigcup \mathcal{D})$ that consists of all nodes $v \in V$ adjacent to at least one node in $\bigcup \mathcal{D}$ forms the required Tutte set $T$. □

**Example 4.2** Let $G = (V, E)$ be the graph as shown in Figure 4.2. It is straightforward to see that the set $T = \{v_1, v_2\}$ is the unique Tutte set satisfying conditions (i) and (ii) of Theorem 4.4. We have $\mathcal{D} = D_1 \cup D_2 \cup D_3 \cup D_4$ and $\mathcal{C} = C_1 \cup C_2$. □

We now return to the general maximum weight matching problem. Let $G = (V, E)$ be a complete graph with even node set $V$ and edge weighting $w$. A *minimum weight perfect matching* is a perfect matching $M^*$ in $G$ that has minimum weight, i.e.,

$$w(M^*) = \min\{w(M) \mid M \subseteq E \text{ is a perfect matching } \}.$$

Edmonds [1965a] obtained the following result.

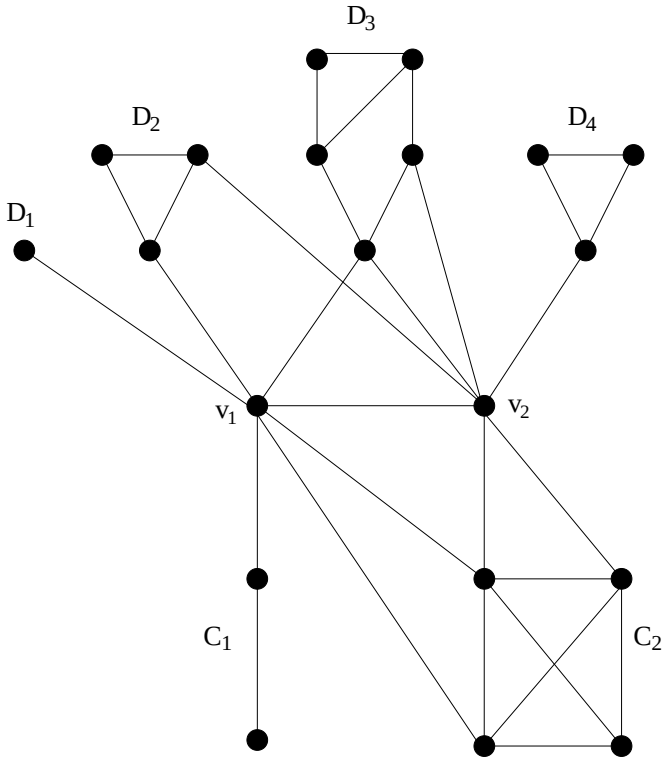


Figure 4.2. Gallai-Edmonds decomposition of a graph

**Theorem 4.5** *Let $G = (V, E)$ be a complete graph with even node set $V$ and edge weighting $w : E \to \mathbb{R}$. Then a minimum weight perfect matching in $G$ can be found in polynomial time.* □

This theorem implies that also the maximum weight matching problem can be solved efficiently. This can be seen as follows. Let $G = (V, E)$ be a graph with edge weighting $w : E \to \mathbb{R}$. Since edges with negative weights will not occur in any maximum weight matching, we may without loss of generality assume that $w \geq 0$. If $G$ is not already a complete graph, we add edges with zero weight. If $|V|$ is odd, then we add a node $u$ to $V$ and edges from $u$ to the nodes in $V$ with zero weight. This way we have extended $G$ to a complete graph $\bar{G}$. Finding a maximum weight matching in $G$ is now equivalent to finding a minimum weight perfect matching in $\bar{G}$ with edge weighting $\bar{w}$ given by $\bar{w}(e) = -w(e)$ for all $e \in \bar{G}(E)$.

**Corollary 4.1** *Let $G = (V, E)$ be a graph with edge weighting $w : E \to \mathbb{R}$. Then a maximum weight matching in $G$ can be found in polynomial time.* □

A *minimum weight maximum matching* in a graph $G = (V, E)$ with edge weighting $w : E \to \mathbb{R}$ is a maximum matching $M^*$ that has minimum weight, i.e.,

$$w(M^*) = \min\{w(M) \mid M \subseteq E \text{ is a maximum matching }\}.$$

Also the next result is a direct consequence of Theorem 4.5.

**Corollary 4.2** *Let $G = (V, E)$ be a graph with edge weighting $w : E \to \mathbb{R}$. Then a minimum weight maximum matching in $G$ can be found in polynomial time.*

**Proof**: If $G$ is not already a complete graph, then we add edges with a sufficient large weight, say $w(E) + 1$. If $|V|$ is odd, then we add a node $u$ to $V$ and edges from $u$ to the nodes in $V$ with weight $w(E) + 1$. This way we have extended $G$ to a complete graph $\bar{G}$. Then computing a minimum weight maximum matching in $G$ is equivalent to computing a minimum weight perfect matching in $\bar{G}$. □

## 4.2 Matching games in general

### 4.2.1 Introduction

Consider again the example of an exchange market consisting of a group of persons $N$ as described in Section 4.1: Each $i \in N$ has exactly one good to offer and can do business with at most one other person. A pair $i, j \in N$ is only willing to do business with each other, if they obtain a common profit $w(i, j) \geq 0$. So an edge between $i$ and $j$ only exists in case $w(i, j) \geq 0$. Obviously the total profit will be maximal, if all persons in $N$ cooperate and a maximum weight matching can be constructed. The problem of dividing the total profit among the persons in $N$ can be modeled as a matching game.

**Definition 4.3** A *matching game* $(N, v)$ is determined by a graph $G = (N, E)$ with node set $N$ and by a rational weight function $w \geq 0$ defined on the edge set $E$. The value $v(S)$ of a coalition $S \subseteq N$ is the value of a maximum weight matching in the subgraph induced by $S$. □

In this definition we assume that $w$ is a positive function, since edges with a negative weight will not occur in any maximum weight matching. It is

straightforward to see that

$$v(S) + v(T) \leq v(S \cup T) \quad \text{for all } S, T \subseteq N, S \cap T = \emptyset.$$

So a matching game is superadditive, and $v(N)$ is the maximum total profit that players in $N$ can obtain. Furthermore, $v(i) = 0$ for all individual players $i \in N$.

The underlying discrete structure of a matching game $(N, v)$ is a graph $G$ and an edge weighting $w$. Let $< w >$ again denote the maximum size of the edge weights, i.e., $< w >= \max\{< w(i, j) > \mid (i, j) \in E\}$. Then we may define $< N, v >= |N| < w >$.

**Example 4.3** Consider the graph $G = (N, E)$ with edge weighting $w$ as shown in Figure 4.3. The matching game $(N, v)$ obtained from this graph is determined by $N = \{1, 2, 3, 4\}$ and $v : 2^N \to \mathbb{R}_+$ given by

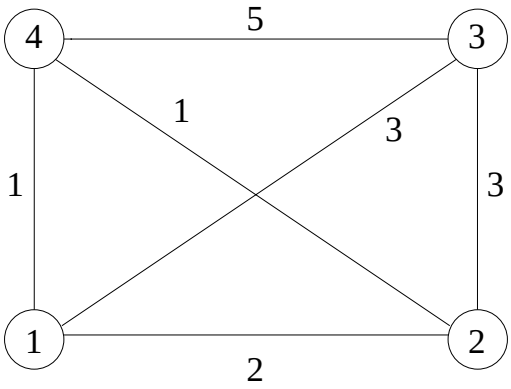

Figure 4.3

$$\begin{array}{llllllllll}
v(1) = 0 & v(1,2) = 2 & v(1,2,3) = 3 & v(N) = 7. \\
v(2) = 0 & v(1,3) = 3 & v(1,2,4) = 2 & \\
v(3) = 0 & v(1,4) = 1 & v(1,3,4) = 5 & \\
v(4) = 0 & v(2,3) = 3 & v(2,3,4) = 5 & \\
 & v(2,4) = 1 & & \\
 & v(3,4) = 5 &
\end{array}$$

□

Given a graph $G = (N, E)$ and edge weighting $w$, a *node matching game* arises if a positive weight function $\bar{w} : N \to \mathbb{R}_+$ on the nodes of $G$ exists in such a way that

$$w(i, j) = \bar{w}_i + \bar{w}_j \quad \text{for all } (i, j) \in E.$$

If $w \equiv 1$ (or equivalently $\bar{w} \equiv \frac{1}{2}$), then we call the corresponding node matching game a *cardinality matching game.*

A matching game associated with a bipartite graph is called an *assignment game*. Shapley and Shubik [1972] introduced this game and showed that the core of an assignment game is always nonempty. Assignment games have been widely studied in the literature (see, e.g., Quint [1991], Granot and Granot [1992]). Solymosi and Raghavan [1994] present an efficient algorithm for computing the nucleolus of an assignment game.

With respect to a general matching game, Faigle, Kern, Fekete and Hochstättler [1998] present an efficient algorithm for computing the nucleon and point out that the problem of computing the nucleolus remains unsolved. In the next section we summarize the results known for general matching games.

### 4.2.2 Solution concepts for matching games

The simplest matching game is a matching game $(N, v)$ determined by $G = K_2$, the complete graph on two nodes. The core of this game is equal to

$$\text{core}(N, v) = \{x \in \mathbb{R}^2 \mid x_1 + x_2 = w(1, 2) \text{ and } x \geq 0\},$$

and the nucleolus of $(N, v)$ is the allocation $(\frac{1}{2}w(1, 2), \frac{1}{2}w(1, 2))$.

During the rest of this chapter we will assume that $G \neq K_2$. In contrast to an assignment game, a general matching game can have an empty core. This can already be the case for a matching game on three nodes (cf. Example 2.3, which can be interpreted as a matching game). Below we give some more examples of (cardinality) matching games with empty core. We have also computed the nucleolus of these games.

**Example 4.4** Let $G = (N, E)$ be the graph with edge weighting $w \equiv 1$ as shown in Figure 4.4. $N$ is split into $\{t\} \cup N(D_1) \cup N(D_2)$. Consider the standard procedure for computing the nucleolus (see Section 2.3). The linear

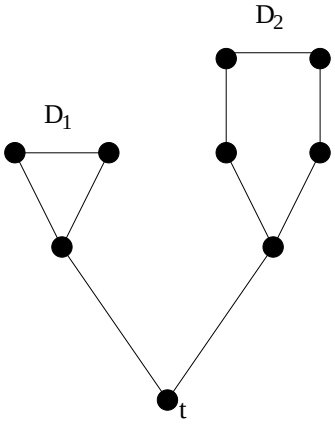

Figure 4.4

program $(P_1)$ has already a unique optimal solution: the nucleolus $\eta(N, v)$ given by

$$\eta_i(N, v) = \begin{cases} \frac{4}{7} & \text{if } i = t \\ \frac{3}{7} & \text{if } i \in N(D_1) \cup N(D_2) \end{cases}$$

and $\epsilon_1 = -\frac{3}{7}$. □

**Example 4.5** Let $G = (N, E)$ be the graph with edge weighting $w \equiv 1$ as shown in Figure 4.5. We have $N = N(D_1) \cup N(D_2) \cup N(D_3)$. Then $\epsilon_1 = -1$ and $P_1(-1)$ contains all allocations $x \in \mathbb{R}^N$ for which

$$\begin{array}{lcll} x_i & = & x_j & (i, j \in N(D_p),\ p = 1, \ldots, 3) \\ x_i + x_j & = & \frac{1}{2} & (i \in N(D_1),\ j \in N(D_2)) \\ x_i & = & \frac{1}{2} & (i \in N(D_3)) \\ x & \geq & 0. & \end{array}$$

The nucleolus $\eta(N, v)$ is given by

$$\begin{array}{lcll} \eta(N, v) & \equiv & \frac{1}{4} & \text{on } N(D_1) \cup N(D_2) \\ \eta(N, v) & \equiv & \frac{1}{2} & \text{on } N(D_3). \end{array}$$

□

For $E' \subseteq E$ we let $N(E')$ denote the set of nodes covered by $E$. If no misunderstanding is possible, we also use the following shorthand notation: If

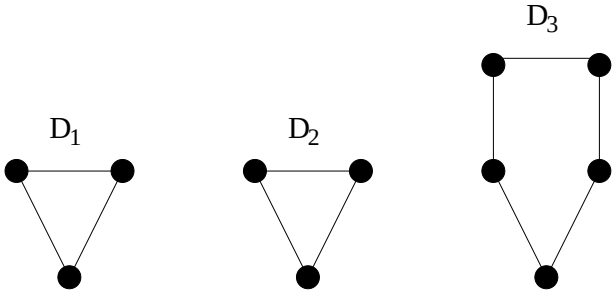


Figure 4.5

$e = (i, j) \in E$, we write $x(e) = x(\{i, j\})$. More generally, if $M \subseteq E$ is a matching, we let $x(M) := x(N(M))$. The following theorem gives an alternative characterization of the core of a matching game. It turns out that only the core constraints for the one-and two-person coalitions are of importance.

**Theorem 4.6** *Let $(N, v)$ be a matching game obtained from a graph $G = (N, E)$ with edge weighting $w$. Then core$(N, v)$ equals the set of allocations $x \in \mathbb{R}^N$ that are in the polyhedron $P_c$ defined by the following constraints:*

$$\begin{array}{lrcll} P_c & x(N) & = & v(N) & \\ & x(e) & \geq & w(e) & \text{for all } e \in E \\ & x_i & \geq & 0 & \text{for all } i \in N. \end{array}$$

*If $x \in$ core$(N, v)$, then*

*(i) $x_i = 0$ for all $i \in N$ not covered by a maximum weight matching.*

*(ii) $x(e) = w(e)$ for all $e \in E$ contained in a maximum weight matching.*

**Proof**: The proof of the first part is straightforward, using the fact that the above constraints imply $x(S) \geq v(S)$ for all $S \subset N$.

Suppose core$(N, v)$ is nonempty, and let $x \in$ core$(N, v)$. Let $M$ be a maximum weight matching in $G$. The core constraints imply $x(N) = v(N) = w(M)$, $x \geq 0$ and $x(e) \geq w(e)$ for all $e \in M$. Then

$$w(M) = x(N) = x(N \backslash N(M)) + x(M) \geq x(M) \geq w(M)$$

implies that $x_i = 0$ for all $i \notin N(M)$ and $x(e) = w(e)$ for all $e \in M$. □

So, for matching games the number of inequalities defining the core have been reduced to a polynomial number.

**Corollary 4.3** *Checking whether the core of a matching game $(N, v)$ is empty or not can be done in polynomial time.* □

By definition, the least core of a matching game $(N, v)$ contains all optimal solutions of the linear program

$$\begin{array}{lllll} (LC) & \max & \epsilon & & \\ & s.t. & x(S) & \geq & v(S)+\epsilon \quad (S \neq \emptyset, N) \\ & & x(N) & = & v(N). \end{array}$$

Let $\epsilon^*$ denote the optimal value of $(LC)$. Also computing an element in the least core of a matching game $(N, v)$ can be done in polynomial time. In order to prove this we need the following lemmas. Recall that we assume that $G \neq K_2$, in which case $\epsilon^* = \frac{1}{2}w(i, j)$.

**Lemma 4.2** *Let $(N, v)$ be a matching game. Then $\epsilon^* = 0$ if core$(N, v)$ is nonempty, and $\epsilon^* < 0$ if core$(N, v)$ is empty.*

**Proof**: Obviously $\epsilon^* \geq 0$ if and only if core$(N, v)$ is nonempty. Suppose $\epsilon^* > 0$ and $x \in \mathbb{R}^N$ is an allocation in the least core of $(N, v)$. Let $M$ be a maximum weight matching. Since $\epsilon^* > 0$, $x$ is also a core allocation. By Theorem 4.6 $x(e) = w(e)$ for an edge $e \in M$, which contradicts the least core constraint $x(e) \geq w(e) + \epsilon > w(e)$. □

**Lemma 4.3** *Let $(N, v)$ be a matching game. Then least core$(N, v) \subseteq \mathbb{R}^N_+$.*

**Proof:** If core$(N, v)$ is nonempty, the lemma follows trivially from Theorem 4.6. Suppose core$(N, v)$ is empty, and assume to the contrary that $(x, \epsilon_1)$ is an optimal solution of $(LC)$, and $x_i < 0$ for some $i \in N$. By Lemma 4.2 $\epsilon_1 < 0$.

Claim: If $S \subseteq N$ satisfies $x(S) \geq v(S) + \epsilon_1$ with equality, then $i \in S$.

Proof: Assume to the contrary that $i \notin S$.

case (i) $S \subset S \cup i \subset N$.
Then $x(S \cup i) < x(S) = v(S) + \epsilon_1 \leq v(S \cup i) + \epsilon_1$ contradicts the feasibility of $x$.

case (ii) $S \subset S \cup i = N$.
Then $x(N) = x(S) + x_i = v(S) + \epsilon_1 + x_i < v(S) \leq v(N)$ again contradicts the feasibility of $x$.

Hence the claim is true. But then we may slightly increase $x_i$ and decrease $x$ on $N \backslash i$ uniformly by the same total amount, thereby obtaining a better solution. This proves the lemma. □

Due to Lemma 4.3, the linear program $(LC)$ is equal to $(P_1)$, the first linear program that has to be solved in order to compute the nucleolus (see Section 2.3). Lemma 4.3 also implies that the nucleolus is nonempty for all matching games. Let $\mathcal{M}$ denote the set of matchings $M \subseteq E$. Then both linear programs can now be stated as

$$\begin{array}{llrcll} (P_1) & \max & \epsilon & & & \\ & s.t. & x(M) & \geq & w(M)+\epsilon & (M \in \mathcal{M}) \\ & & x(N) & = & v(N) & \\ & & x & \geq & 0. & \end{array}$$

**Theorem 4.7** *Computing an allocation in the least core of a matching game $(N, v)$ can be done in polynomial time.*

**Proof**: By Theorem 2.3 it suffices to show that for given $x \in \mathbb{R}^N$ and $\epsilon \in \mathbb{R}$ we can efficiently check whether $e_{min}(x) \geq \epsilon$ or not. Since we can formulate the linear program $(LC)$ as the linear program $(P_1)$ described above, this comes down to check whether

$$x(M) \geq w(M) + \epsilon \quad (M \in \mathcal{M})$$

holds. This can be done by solving a maximum weight matching problem on $G = (N, E)$ with respect to the edge weights

$$\tilde{w}(i, j) := w(i, j) - x_i - x_j \quad ((i, j) \in E).$$

Hence applying Corollary 4.1 finishes the proof. □

Faigle, Kern, Fekete and Hochstättler [1998] showed that the nucleon of a matching game can be computed in polynomial time. For a matching game with a nonempty core this also holds for the nucleolus.

**Proposition 4.1** *Let $(N, v)$ be a matching game. If core$(N, v)$ is nonempty, then the nucleolus of $(N, v)$ can be computed in polynomial time.*

**Proof**: We define

$$\begin{array}{llrcll} (P_1^+) & \max & \epsilon & & & \\ & s.t. & x(e) & \geq & w(e)+\epsilon & (e \in E) \\ & & x_i & \geq & \epsilon & (i \in N) \\ & & x(N) & = & v(N) & \end{array}$$

with optimum value $\epsilon_1^+ \in \mathbb{R}$. (In fact $\epsilon_1^+ = 0$ holds, cf. Lemma 4.2). Next we identify

$$E_1 := \{e \in E \mid e \in Fix\, P_1^+(\epsilon_1^+)\} \text{ and } N_1 := \{i \in N \mid i \in Fix\, P_1^+(\epsilon_1^+)\}$$

and then solve

$$\begin{array}{lllll} (P_2^+) & \max & \epsilon & & \\ & s.t. & x(e) & = & w(e) + \epsilon_1^+ \quad (e \in E_1) \\ & & x_i & = & \epsilon_1^+ \quad (i \in N_1) \\ & & x(e) & \geq & w(e) + \epsilon \quad (e \in E \backslash E_1) \\ & & x_i & \geq & \epsilon \quad (i \in N \backslash N_1) \\ & & x(N) & = & v(N) \end{array}$$

with optimum value $\epsilon_2^+ = \epsilon_2$ etc. until we obtain a linear program $(P_r^+)$ that defines a unique solution $x^* \in \mathbb{R}^N$. Using the fact that the constraints $x \geq 0$ and $x(e) \geq w(e)$ for all $e \in E$ imply $x(S) \geq v(S)$ for all $S \subset N$, it is clear that $x^*$ is equal to the nucleolus of $(N, v)$. □

The above approach fails in case $\epsilon_1 < 0$. Whether there exists an efficient algorithm for computing the nucleolus of a general matching game is not known. In the next section we restrict ourselves to node matching games. We show that for this class of matching games the nucleolus can be computed in polynomial time. This generalizes the result of Kern and Paulusma [2000], where cardinality matching games are considered. We first give a polynomial description of the least core. Next, we use this description to present an efficient algorithm for computing the nucleolus.

## 4.3 Node matching games

Recall that node matching games arise when the edge weights $w(e)$ are the sum of certain *positive* weights on the incident nodes. In this section we consider a node matching game $(N, v)$ that is obtained from a graph $G = (N, E)$ with node weighting $w : N \to \mathbb{R}_+$. The edge weighting $w$ is defined by $w(i, j) = w_i + w_j$ for all $(i, j) \in E$.

We use the following standard notation: For $S \subseteq N$ we let $E(S) \subseteq E$ denote the set of edges joining nodes of $S$. Recall that $N(E')$ denotes the set of nodes

covered by a subset $E' \subseteq E$. Furthermore, for $S \subseteq N$ we use the shorthand notation

$$w(S) = \sum_{i \in S} w_i.$$

Note that $w(S) \geq v(S)$. Recall that we use the same notation for a matching $M \subset \mathcal{M}$, i.e.,

$$w(M) = \sum_{e \in M} w(e) = \sum_{(i,j) \in M} (w_i + w_j).$$

We assume that $T \subseteq N$ is the Tutte set satisfying the conditions (i) and (ii) in Theorem 4.4. The set $\tilde{\mathcal{M}}^*$ denotes the set of maximum weight matchings in $G$, and $\mathcal{M}^*$ denotes the set of maximum matchings in $G$. Each $M \in \mathcal{M}^*$ matches $T$ completely in $\mathcal{D}$. By condition (ii) of Theorem 4.4, given $D \in \mathcal{D}$, there is some $M \in \mathcal{M}^*$ matching $T$ into $\mathcal{D} \backslash \{D\}$. We say that $M$ *leaves* $D$ *uncovered*.

We will sometimes identify subsets of $N$ with the corresponding induced subgraphs. For example, if $i \in N$ is a node, we do not hesitate to write $i \in D$ to indicate that $i$ is a node of the component $D \in \mathcal{D}$. If $x \in \mathbb{R}^N$ is an allocation, we consequently write

$$x(D) = \sum_{i \in D} x_i.$$

Each maximum weight matching can be extended to a maximum matching in the following sense.

**Lemma 4.4** *Let $\tilde{M} \in \tilde{\mathcal{M}}^*$ be a maximum weight matching. Then there exists a matching $M \in \mathcal{M}^*$ in such a way that $N(\tilde{M}) \subseteq N(M)$.*

**Proof**: Suppose $\tilde{M} \in \tilde{\mathcal{M}}^*$ is a maximum weight matching not already in $\mathcal{M}^*$. By definition of the Tutte set $T$ every maximum matching $M \in \mathcal{M}^*$ covers $\bigcup \mathcal{C}$ and matches $T$ completely into $\bigcup \mathcal{D}$. Now choose a maximum matching $M \in \mathcal{M}^*$ in such a way that all nodes in each component $D \in \mathcal{D}$ that are covered by $\tilde{M}^*$, also are covered by $M$. That this is possible can be seen as follows. Suppose $M$ does not cover a node $i \in \bigcup \mathcal{D}$ that is covered by $\tilde{M}$. Consider the maximum alternating path $P \subseteq M \cup \tilde{M}$ starting in $i$ and ending in $j$. (Such a path exists according to Lemma 4.1.) Since $M \in \mathcal{M}^*$, $j$ is not covered by $\tilde{M}$. Reversing $M$ along $P$ results in a matching $\bar{M} \in \mathcal{M}^*$ covering

$N(M)\backslash j \cup i$. For other nodes that are covered by $\tilde{M}$, but no by $M$, we do the same. Hence we may assume that $M$ covers $(N(\tilde{M}) \cap \bigcup \mathcal{C}) \cup (N(\tilde{M}) \cap T) \cup (N(\tilde{M}) \cap \bigcup \mathcal{D}) = N(\tilde{M})$. □

If $\tilde{M} \in \tilde{\mathcal{M}}^*$ and $M \in \mathcal{M}^*$ with $N(\tilde{M}) \subseteq N(M)$, then obviously $w(M) = w(\tilde{M})$. So $\tilde{\mathcal{M}}^* \subseteq \mathcal{M}^*$ holds if the node weighting $w$ is strictly positive.

The following lemma gives a lower bound on the value $v(N\backslash i)$ for all $i \in \bigcup \mathcal{D}$.

**Lemma 4.5** *For all $i \in \bigcup \mathcal{D}$, $v(N\backslash i) \geq v(N) - w_i$. If $w > 0$ on $\mathcal{D}$ then $v(N\backslash i) > v(N) - w_i$ holds.*

**Proof**: Suppose $D \in \mathcal{D}$ and $i \in D$. If $M^* \in \tilde{\mathcal{M}}^*$ is a maximum weight matching not covering $i$, then $v(N\backslash i) = v(N) \geq v(N) - w_i$.

Otherwise assume that $i$ is covered by a matching $M \in \tilde{\mathcal{M}}^*$. By Lemma 4.4 we may assume that $M$ is a maximum matching. According to Theorem 4.4 there exists a maximum matching $M' \in \mathcal{M}^*$ not covering $D$. Since $D$ is factor-critical, we may assume that $M'$ does not cover $i$.

By Lemma 4.1 there exists a unique maximal alternating path $P \subseteq M \cup M'$ starting in $i$ and ending in a node $j$. Since both matchings $M$ and $M'$ cover $T \cup \bigcup \mathcal{C}$, $j$ must be a node in a component $D' \in \mathcal{D}$ (possibly equal to $D$). Reversing $M$ along $P$ results in a matching $\tilde{M} \in \mathcal{M}^*$ covering $N(M)\backslash i \cup j$. Hence $v(N\backslash i) \geq w(\tilde{M}) = w(M) - w_i + w_j = v(N) - w_i + w_j$, which proves the lemma. □

The next result gives a sufficient condition for the core to be nonempty. If the node weighting $w : N \to \mathbb{R}_+$ is strictly positive, then this is also a necessary condition.

**Theorem 4.8** *If $|D| = 1$ for all $D \in \mathcal{D}$, then core$(N, v)$ is nonempty. If the node weighting $w : N \to \mathbb{R}_+$ is strictly positive, then also the reverse statement is true.*

**Proof**: Suppose $|D| = 1$ for all $D \in \mathcal{D}$. Delete the components $C \in \mathcal{C}$ and all edges between nodes in $T$. In this way a graph $G'$ with node set $N' = T \cup \bigcup \mathcal{D}$ has been constructed. On $G'$ we define the same node weighting $w$ (restricted to $T \cup \bigcup \mathcal{D}$). Since $G'$ is bipartite, the corresponding node matching game $(N', v')$ has a nonempty core (Shapley and Shubik [1972]).

Choose an allocation $\tilde{x} \in \text{core}(N', v')$ in such a way that $\tilde{x}_i \geq w_i$ for all $i \in T$. Such an allocation exists by the following reasoning. Add to $G'$ an extra node $s$ with weight $w_s = 0$ and connect $s$ to each node in $T$. This new graph is also bipartite and the corresponding matching game $(N' \cup s, v'')$ has a nonempty core. By Lemma 4.4 a maximum weight matching in $G'$ exists that matches all nodes in $T$. Then the value of a maximum weight matching has not increased after adding the extra node $s$. Hence a core allocation $\tilde{x}$ of the node matching game $(N' \cup s, v'')$ (restricted to $N'$) lies in $\text{core}(N', v')$. By Theorem 4.6 we have $\tilde{x}_s = 0$. Then $\tilde{x}_i = \tilde{x}_i + \tilde{x}_s \geq w_i + w_s = w_i$ for all $i \in T$. Hence $\tilde{x}' \in \mathbb{R}^N$ defined by

$$\begin{array}{rcll} \tilde{x}' & \equiv & w_c & \text{on } \bigcup \mathcal{C} \\ \tilde{x}' & \equiv & \tilde{x} & \text{on } T \cup \bigcup \mathcal{D} \end{array}$$

is easily seen to be an element in $\text{core}(N, v)$.

Suppose $w > 0$ on $\mathcal{D}$ and $D \in \mathcal{D}$ with $|D| \geq 3$. Let $e = (i, j) \in E(D)$. By Lemma 4.5 we have $v(N \backslash i) > v(N) - w_i$ and $v(N \backslash j) > v(N) - w_j$. So if $x \in \mathbb{R}^N$ were in the core, then

$$x(N \backslash i) \geq v(N \backslash i) > v(N) - w_i \text{ and } x(N \backslash j) \geq v(N \backslash j) > v(N) - w_j.$$

Together with $x(N) = v(N)$ these inequalities imply $x_i + x_j < w_i + w_j$, a contradiction with the core constraint $x(e) = x_i + x_j \geq w_i + w_j$. Hence $\text{core}(N, v)$ must be empty. □

The following result shows that, according to a core allocation $x \in \mathbb{R}^N$, the amount that is allocated to a node $i \in \mathcal{D}$ does not exceed the weight $w_i$. This implies that $x_i = 0$ for a node $i \in \bigcup \mathcal{D}$ with weight $w_i = 0$.

**Proposition 4.2** *If core$(N, v)$ is nonempty, then for all $x \in$ core$(N, v)$*

$$x_i \leq w_i \text{ for all } i \in \bigcup \mathcal{D}.$$

*If $D \in \mathcal{D}$ has size $|D| > 1$, then $x_i = w_i$ holds for all $i \in D$.*

**Proof**: Suppose $D \in \mathcal{D}$ and $i \in D$. Let $x \in \mathbb{R}^N$ be a core allocation. By Lemma 4.5, $v(N \backslash i) \geq v(N) - w_i$. Then $x_i = x(N) - x(N \backslash i) \leq v(N) - v(N \backslash i) \leq w_i$.

If $|D| > 1$ then there is a node $j \in D$ such that $(i, j)$ is an edge in $E$. Since $x \in \text{core}(N, v)$, $x_i + x_j \geq w_i + w_j$. Since we have just deduced that $x_i \leq w_i$ and $x_j \leq w_j$, this implies that $x_i = w_i$ (and $x_j = w_j$). □

### 4.3.1 The least core of node matching games

Recall that, due to Lemma 4.3, the linear program defining the least core can be stated as the linear program

$$
\begin{array}{lllll}
(P_1) & \max & \epsilon & & \\
& s.t. & x(M) & \geq & w(M)+\epsilon \quad (M \in \mathcal{M}) \\
& & x(N) & = & v(N) \\
& & x & \geq & 0
\end{array}
$$

with optimal value $\epsilon_1 \leq 0$. In the following we assume that core$(N, v)$ is *empty*. Equivalently, by Lemma 4.2 $\epsilon_1 < 0$. Furthermore, by Theorem 4.8 there exists at least one component $D \in \mathcal{D}$ with size $|D| > 1$.

**Example 4.6** Consider the node matching game $(N, v)$ obtained from the graph $G = (N, E)$ with node weighting $w$ as indicated in Figure 4.6.

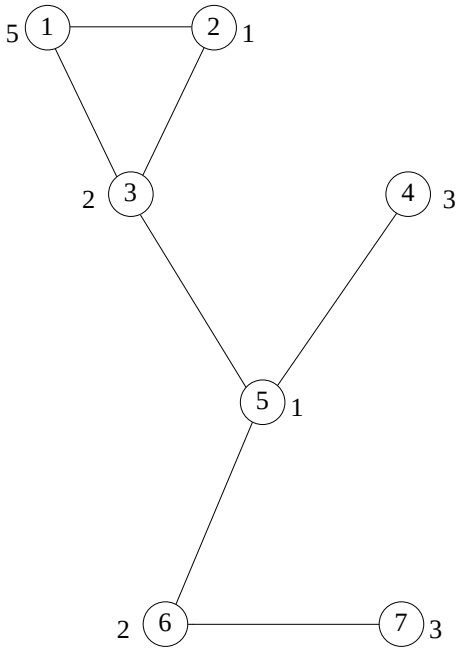


Figure 4.6

Clearly, the set $T = \{5\}$ is the Tutte set satisfying conditions (i) and (ii) of Theorem 4.4. The graph $G\backslash T$ has two odd components $D_1$ with node set $\{1, 2, 3\}$ and $D_2$ with node set $\{4\}$, and one even component $C_1$ with node set $\{6, 7\}$. By Theorem 4.8 core$(N, v)$ is empty.

The value of a maximum weight matching in $G$ is equal to $v(N) = 16$. Let $(x, \epsilon)$ be a feasible solution for $(P_1)$. Then $x_1 + x_2 \geq 6 + \epsilon$, $x(N\backslash 1) \geq 12 + \epsilon$ and $x(N\backslash 2) \geq 16 + \epsilon$. Together with $x(N) = 16$ we obtain the upper bound $\epsilon \leq -\frac{2}{3}$.

It is easy to check that $\tilde{x} = (4\frac{2}{3}, \frac{2}{3}, 1\frac{2}{3}, 0, 4, 2, 3)$ is an allocation in $P_1(-\frac{2}{3})$. Hence $\epsilon_1 = -\frac{2}{3}$. In fact, the allocations in leastcore$(N, v) = P_1(-\frac{2}{3})$ turn out to be exactly the allocations in the polyhedron determined by the constraints

$$\begin{array}{lcl} x_1 & = & 4\frac{2}{3} \\ x_2 & = & \frac{2}{3} \\ x_3 & = & 1\frac{2}{3} \\ x_3 + x_5 & \geq & 3 \\ x_4 + x_5 & = & 4 \\ x_5 + x_6 & \geq & 3 \\ x_6 + x_7 & = & 5 \\ x(N) & = & 16 \\ x & \geq & 0. \end{array}$$

□

By Theorem 4.7 we can efficiently compute an allocation in the least core of $(N, v)$. Here we aim for more, namely a concise description of $P_1(\epsilon_1)$.

Note that in Example 4.6 $x_i - w_i = x_j - w_j \leq 0$ for all nodes $i$, $j$ in the same component $D \in \mathcal{D}$. Also $x(e) \geq w(e)$ for all edges in $E$ with at least one end point not in $\bigcup \mathcal{D}$. Furthermore, large matchings such as the matching $M = \{(1, 2), (4, 5), (6, 7)\}$ and matchings completely contained in $\bigcup \mathcal{D}$ turn out to become tight, i.e., the corresponding constraints in $(P_1)$ become tight. Also for Example 4.4 and Example 4.5 this is true. Generally speaking, this holds for any node matching game, i.e., we will prove that the least core of $(N, v)$ can be described as the solution set of the following linear program.

$$\begin{array}{lrcll} \max & \epsilon & & & \\ s.t. & x_i - w_i & = & x_j - w_j & (i, j \in D, D \in \mathcal{D}) \\ & x_i & \leq & w_i & (i \in \bigcup \mathcal{D}) \\ & x(e) & \geq & w(e) & (e \in E\backslash E(\bigcup \mathcal{D})) \\ & x(N) & = & v(N) & \\ & x & \geq & 0 & \\ & \epsilon & = & \displaystyle\sum_{D \in \mathcal{D}} \frac{|D| - 1}{|D|} (x(D) - w(D)). & \end{array}$$

In this description the exponentially many constraints defining the least core have been replaced by a polynomially number.

If core$(N, v)$ would be nonempty, this is straightforward to see: By Theorem 4.6 core$(N, v)$ is equal to the polyhedron $P_c$. By Proposition 4.2 we may add the constraints $x \leq w$ on $\bigcup \mathcal{D}$, and $x_i = w_i$ for all $i \in D$ and $D \in \mathcal{D}$ with $|D| > 1$. So, if core$(N, v)$ would be nonempty, then the optimal solutions of the linear program as defined above describe the (least) core of $(N, v)$.

However, we have assumed that core$(N, v)$ is empty, and in the rest of this section we show that also in this case the linear program $(P_1)$ can be replaced by the linear program as described above. As a first step, we introduce a relaxation $(\hat{P}_1)$ of $(P_1)$ below, which is easier to analyze and, as we shall see, defines the same optimum value. To motivate this approach, note that, as shown in Example 4.6, rather large matchings and matchings completely contained in $\bigcup \mathcal{D}$ are expected to become tight when solving $(P_1)$ instead of small ones (single nodes and edges) as in the case of a nonempty core.

Let $\mathcal{M}_{\mathcal{D}}$ denote the set of matchings $M \subseteq E(\bigcup \mathcal{D})$ that are completely contained in the union of the odd components. We shall study the following relaxation of $(P_1)$:

$$
\begin{array}{lllll}
(\hat{P}_1) & \max & \epsilon & & \\
& s.t. & x(M) & \geq & w(M) + \epsilon \quad (M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}) \\
& & x(N) & = & v(N) \\
& & x & \geq & 0
\end{array}
$$

with optimal value $\hat{\epsilon}_1 \in \mathbb{R}$. Obviously $\hat{\epsilon}_1 \leq 0$. Below we will show that $\hat{\epsilon}_1 < 0$. Note that we do not explicitly consider maximum weight matchings in the description of $(\hat{P}_1)$. However, by Lemma 4.4 we know that, in case these matchings do not already belong to the set $\mathcal{M}^*$ of maximum matchings, there exists a corresponding matching $M \in \mathcal{M}^*$.

To investigate the structure of optimal solutions of $(\hat{P}_1)$, we first introduce some notation. As before, $\hat{P}_1(\epsilon)$ denotes the set of $x \in \mathbb{R}^N$ such that $(x, \epsilon)$ is feasible for $(\hat{P}_1)$. If $x \in \hat{P}_1(\hat{\epsilon}_1)$ is an optimal solution, we say that $M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$ is *x-tight*, if $x(M) = w(M) + \hat{\epsilon}_1$. Given a feasible solution $x \in \hat{P}_1(\epsilon)$ and $D \in \mathcal{D}$, let

$$x_D := \frac{x(D)}{|D|}$$

denote the average value of $x$ on $D$. In the same way we define the average weight on a component $D \in \mathcal{D}$ :

$$w_D := \frac{w(D)}{|D|}.$$

Define $\bar{x} \in \mathbb{R}^N$ by *averaging $x$ with respect to the node weights $w$* on each component $D \in \mathcal{D}$, i.e.,

$$\bar{x}_i := x_D - w_D + w_i \qquad (i \in D,\, D \in \mathcal{D})$$

and leaving $x$ unchanged on $T \cup \bigcup \mathcal{C}$. Note that $\bar{x}_i(D) = x(D)$, and

$$\bar{x}_i - w_i = \bar{x}_j - w_j \quad \text{for all } i,\, j \in D.$$

First we show that $\bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$, if $x \in \hat{P}_1(\hat{\epsilon}_1)$ is an optimal solution. In order to this we need the following series of lemmas.

**Lemma 4.6** *Let $x \in \hat{P}_1(\epsilon)$ for some $\epsilon < 0$. If no $x$-tight matchings in $\mathcal{M}_{\mathcal{D}}$ exist, then $\epsilon$ is not optimal.*

**Proof**: Suppose $x(M) > w(M) + \epsilon$ for all matchings $M \in \mathcal{M}_{\mathcal{D}}$. If no $x$-tight matchings in $\mathcal{M}^*$ exist either then, obviously, $\epsilon$ is not optimal. If $M \in \mathcal{M}^*$ covers all nodes $i \in \bigcup \mathcal{D}$ with $x_i > 0$, then

$$x(M) = x(N) = v(N) > v(N) + \epsilon,$$

which means that $M$ is not $x$-tight. Hence each $x$-tight matching in $\mathcal{M}^*$ does not cover all nodes $i \in \bigcup \mathcal{D}$ with $x_i > 0$. Then we may slightly and uniformly decrease $x$ on the set $\{i \in \bigcup \mathcal{D} \mid x_i > 0\}$ and increase it by the same total amount on $T \cup \bigcup \mathcal{C}$. The resulting $\tilde{x}$ has no tight matchings in $\mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$, which implies that $\epsilon$ is not optimal. □

**Lemma 4.7** *If $x \in \hat{P}_1(\epsilon)$ then $\bar{x}(M) \geq w(M) + \epsilon$ for all $M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$.*

**Proof**: Let $x \in \hat{P}_1(\epsilon)$. It suffices to show that the constraints above are still satisfied after averaging $x$ with respect to the node weighting $w$ on some component $D \in \mathcal{D}$. Thus let $D \in \mathcal{D}$ and let $\tilde{x} \in \mathbb{R}^N$ be obtained by averaging $x$ on $D$, i.e.,

$$\tilde{x}_i = x_D - w_D + w_i \quad (i \in D).$$

Suppose $M \in \mathcal{M}^*$. Then either $M$ covers $D$ or $M \cap D = D \backslash i$ for some $i \in D$. In the first case $\tilde{x}(M) = x(M)$ and the claim follows. In the second case we may assume without loss of generality that $i \in D$ maximizes $x_i - w_i$ over $D$, otherwise we replace $M$ inside $D$ by some other near-perfect matching without changing $\tilde{x}(M) - w(M)$. (Recall that $D$ is factor-critical.) But then $x_i - w_i \geq x_D - w_D = \tilde{x}_i - w_i$ and consequently $\tilde{x}(M) - w(M) \geq x(M) - w(M)$, so the claim follows as $x \in \hat{P}_1(\epsilon)$.

Next consider $M \in \mathcal{M}_{\mathcal{D}}$ and assume $M$ minimizes $\tilde{x}(M) - w(M)$ over $\mathcal{M}_{\mathcal{D}}$. If $x_D - w_D > 0$ on $D$, then $M \cap D = \emptyset$. Hence $\tilde{x}(M) = x(M)$ and the claim follows. If $x_D - w_D \leq 0$ on $D$, then $M \cap D$ is without loss of generality a near-perfect matching in $D$ and we argue as we did for $M \in \mathcal{M}^*$. □

It is immediately clear that $\bar{x}(N) = x(N)$. We still have to show that $\bar{x} \geq 0$ for an allocation $x \in \hat{P}_1(\hat{\epsilon}_1)$.

**Lemma 4.8** *If $x \in \hat{P}_1(\hat{\epsilon}_1)$ then $\bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$.*

**Proof**: Let $x \in \hat{P}_1(\hat{\epsilon}_1)$. Besides $\bar{x}(N) = v(N)$ we have already shown, in Lemma 4.7, that $\bar{x}(M) \geq w(M) + \hat{\epsilon}_1$ for all $M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$. Let $w^D_{min}$ denote the minimum weight in a component $D \in \mathcal{D}$, i.e, $w^D_{min} = \min\{w_i \mid i \in D\}$ $(D \in \mathcal{D})$. We will prove that for all $D \in \mathcal{D}$

$$x_D \geq w_D - w^D_{min}.$$

Then $\bar{x} \geq 0$, and we are finished. Now suppose $D' \in \mathcal{D}$ and $x_{D'} < w_{D'} - w^{D'}_{min}$, or equivalently,

$$\sum_{i \in D'} x_i < \sum_{i \in D'} (w_i - w^{D'}_{min}).$$

Note that $|D'| > 1$. Let $j \in D'$ be a node with $w_j = w^{D'}_{min}$. It is straightforward to see that we can obtain an allocation $\tilde{x} \in \mathbb{R}^N$ such that

$$\begin{array}{lcll} \tilde{x}_i & \leq & w_i - w_j & (i \in D') \\ \tilde{x}(D') & = & x(D') & \\ \tilde{x}_i & = & x_i & (i \in N \backslash D') \\ \tilde{x} & \geq & 0. & \end{array}$$

We have chosen $\tilde{x}$ in such a way that $\tilde{x}_j = 0$ and $\tilde{x}_i - w_i \leq \tilde{x}_j - w_j$ for all $i \in D' \backslash j$.

• Below we show that $\tilde{x}$ is an allocation in $\hat{P}_1(\hat{\epsilon}_1)$. Besides $\tilde{x} \geq 0$, obviously $\tilde{x}(N) = v(N)$. So we are left to check whether $\tilde{x}(M) \geq w(M) + \hat{\epsilon}_1$ for all $M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$.

Suppose $M \in \mathcal{M}^*$. Then either $M$ covers $D'$ or $M \cap D' = D' \backslash i$ for some $i \in D'$. In the first case $\tilde{x}(M) = x(M) \geq w(M) + \hat{\epsilon}_1$. In the second case we deduce that $\tilde{x}(D' \backslash i) - w(D' \backslash i) \geq \tilde{x}(D' \backslash j) - w(D' \backslash j) \geq x(D' \backslash j) - w(D' \backslash j)$, since $x_i - w_j \leq x_j - w_j$ and $x_j = 0$. Because $D$ is factor-critical, we can replace the edges in $M$ covering $D' \backslash i$ by $\frac{1}{2}(|D| - 1)$ edges covering $D' \backslash j$. This results in a matching $M'$ such that

$$\tilde{x}(M) - w(M) \geq x(M') - w(M') \geq \hat{\epsilon}_1.$$

Suppose $M \in \mathcal{M}_{\mathcal{D}}$ and assume that $M$ minimizes $\tilde{x}(M) - w(M)$ over $\mathcal{M}_{\mathcal{D}}$. Since $\tilde{x}_i \leq w_i - w_j \leq w_i$, $M \cap D'$ is without loss of generality a near-perfect matching in $D'$ and we argue as we did for $M \in \mathcal{M}^*$.

• Since $\tilde{x}(D') = x(D') < \sum_{i \in D'}(w_i - w_j)$, there exists a node $k \in D'$ for which $\tilde{x}_k < w_k - w_j$, and consequently, $\tilde{x}_k - w_k < \tilde{x}_j - w_j$.

Claim: The node $k \in D'$ is covered by every $\tilde{x}$-tight matching $M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$.

Proof: Suppose $M \in \mathcal{M}^*$ is $\tilde{x}$-tight. If $M$ covers $D'$ then the claim follows. Otherwise $|M \cap D'| = |D'| - 1$. Suppose $M \cap D' = D' \backslash \{k\}$. Since $\tilde{x}_k - w_k < \tilde{x}_j - w_j$, we have $\tilde{x}(D' \backslash k) - w(D' \backslash k) > \tilde{x}(D' \backslash j) - w(D' \backslash j)$. Then a matching $M' \in \mathcal{M}^*$ would exist with $N(M') = N(M) \backslash j \cup k$ and $\tilde{x}(M') < w(M') + \hat{\epsilon}_1$, contradicting the feasibility of $\tilde{x}$.

Suppose $M \in \mathcal{M}_{\mathcal{D}}$ is $\tilde{x}$-tight. If $|M \cap D'| = |D'| - 1$ then, from the above, $M$ must cover $k$. Otherwise $|M \cap D'| < |D'| - 1$. Suppose $M$ does not cover $k$. Since $\tilde{x}_i \leq w_i$ on $D'$, we can extend $M$ to a tight matching $M'$ in $\mathcal{M}_{\mathcal{D}}$ that covers $D' \backslash k$, a contradiction.

• We now show that we can increase $\hat{\epsilon}_1$ by modifying $\tilde{x}$ a little, which would contradict the optimality of $\hat{\epsilon}_1$. We first deduce that $\hat{\epsilon}_1 \leq \tilde{x}_k + \tilde{x}_i - w_i - w_k < 0$, where $i \in N$ is chosen in such a way that $i \in D'$ and $(i, k) \in E$.

Suppose $\tilde{x}(T \cup \bigcup \mathcal{C}) > 0$. Then decrease $\tilde{x}$ on $T \cup \bigcup \mathcal{C}$ and increase $\bar{x}_k$ by the same amount in such a way that the resulting $\tilde{x}'$ is still in $\hat{P}_1(\hat{\epsilon}_1)$. Clearly, $\tilde{x}'$ has no tight matchings in $\mathcal{M}_{\mathcal{D}}$. By Lemma 4.6 $\hat{\epsilon}_1$ is not optimal, a contradiction.

Suppose $\tilde{x}(T \cup \bigcup \mathcal{C}) = 0$. Then $\tilde{x}(\bigcup \mathcal{D}) = x(N) - \tilde{x}(T \cup \bigcup \mathcal{C}) = x(N) = v(N) \geq v(\bigcup \mathcal{D}) = \sum_{D \in \mathcal{D}} (w(D) - w^D_{min})$. Since $\tilde{x}(D') < w(D') - |D'| w^{D'}_{min}$, there exists a component $\tilde{D} \in \mathcal{D}$ with $\tilde{x}(\tilde{D}) > w(\tilde{D}) - |\tilde{D}| w^{\tilde{D}}_{min}$. By Lemma 4.7, we may assume that

$$\tilde{x}_i = x_{\tilde{D}} - w_{\tilde{D}} + w_i > 0 \text{ on } \tilde{D}.$$

Then decrease $\tilde{x}$ on $\tilde{D}$ and increase $\tilde{x}_k$ by the same sufficient small amount. Again the resulting allocation has no tight matchings in $\mathcal{M}_{\mathcal{D}}$, and $\hat{\epsilon}_1$ would not be optimal.

So we have showed that for all $D \in \mathcal{D}$ $x_D \geq w_D - w^D_{min}$, implying that $\bar{x} \geq 0$. □

**Lemma 4.9** $\hat{\epsilon}_1 < 0$.

**Proof**: Suppose $x \in \hat{P}_1(\hat{\epsilon}_1)$. By Lemma 4.8 we may assume that $x = \bar{x}$. If $\bar{x}_D < w_D$ on some $D \in \mathcal{D}$, then consider an edge $e \in E(D)$. We have $\hat{\epsilon}_1 \leq \bar{x}(e) - w(e) = 2(\bar{x}_D - w_D) < 0$.

Suppose $\bar{x}_D \geq w_D$ for all $D \in \mathcal{D}$. Let $M$ be a maximum weight matching in $\mathcal{M}^*$. If $M$ covers all $i \in \bigcup \mathcal{D}$ with $w_i > 0$, then $x'$ given by $x'_i = w_i$ is easily seen to be a core allocation. This contradicts our assumption that core$(N, v)$ is empty. Hence there exists a node $i$ in some $D \in \mathcal{D}$ with $w_i > 0$ that is not covered by $M$. Then $\bar{x}_i = \bar{x}_D - w_D + w_i \geq w_i > 0$, and consequently, $\bar{x}(M) < \bar{x}(M \cup i) \leq \bar{x}(N) = v(N)$. Together with $\bar{x}(M) \geq v(N) + \hat{\epsilon}_1$, this implies $\hat{\epsilon}_1 < 0$. □

Summarizing we conclude that $\hat{\epsilon}_1 < 0$. If $x \in \hat{P}_1(\hat{\epsilon}_1)$ is an optimal allocation, so is $\bar{x}$. Furthermore, some matchings in $\mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$ must be $\bar{x}$-tight. These can in principle be found by minimizing $\bar{x}(M) - w(M)$ over $\mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$. Minimizing $\bar{x}(M) - w(M)$ over $\mathcal{M}^*$ amounts to solving a minimum weight maximum matching problem (cf. Corollary 4.2). Minimizing $\bar{x}(M) - w(M)$ over $\mathcal{M}_{\mathcal{D}}$ is even trivial: We simply choose a near-perfect matching in each component $D \in \mathcal{D}$ with $x_D < w_D$ (plus an arbitrary matching in all components on which $x_D = w_D$). So computing an $\bar{x}$-tight $M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$ for given $\bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$ is easy.

We aim at a more structural characterization of $\bar{x}$-tight matchings for given $\bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$. Let $\mathcal{D}_{max} = \mathcal{D}_{max}(\bar{x}) \subseteq \mathcal{D}$ be the set of odd components on which $\bar{x}_D - w_D$ is maximum (among all $D \in \mathcal{D}$).

**Lemma 4.10** *No $\bar{x}$-tight $M \in \mathcal{M}^*$ covers all $D \in \mathcal{D}_{max}$. If $\bar{x}$-tight matchings in $\mathcal{M}^*$ exist at all, then for each $D \in \mathcal{D}_{max}$ there is some $\bar{x}$-tight $M \in \mathcal{M}^*$ leaving $D$ uncovered.*

**Proof**: Suppose $\bar{M} \in \mathcal{M}^*$ is $\bar{x}$-tight and covers $D \in \mathcal{D}_{max}$. Let $\tilde{M} \in \mathcal{M}^*$ be any matching not covering $D$. (Recall Theorem 4.4). Let $P \subseteq \bar{M} \cup \tilde{M}$ be the unique maximal alternating path starting in $D$ (in a node $i$ uncovered by $\tilde{M}$) and ending in, say, $\tilde{D}$ (in a node $j$ uncovered by $\bar{M}$). Reversing $\bar{M}$ along $P$ results in a matching $M \in \mathcal{M}^*$ covering $N(\bar{M})\backslash i \cup j \in \tilde{D}$. Since $D \in \mathcal{D}_{max}$, we have $\bar{x}_i - w_i = \bar{x}_D - w_D \geq \bar{x}_{\tilde{D}} - w_{\tilde{D}} = \bar{x}_j - w_j$, hence

$$\bar{x}(M) - w(M) \leq \bar{x}(\bar{M}) - w(\bar{M}).$$

Thus $M$ must be $\bar{x}$-tight again, proving the second claim. The first claim follows by observing that if $\bar{M}$ would cover all $D \in \mathcal{D}_{max}$, then $\tilde{D} \notin \mathcal{D}_{max}$ (as it is uncovered by $\bar{M}$). But then $\bar{x}_D - w_D > \bar{x}_{\tilde{D}} - w_{\tilde{D}}$ and $\bar{x}(M) - w(M) < \bar{x}(\bar{M}) - w(\bar{M}) = \hat{\epsilon}_1$, contradicting $\bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$. □

Let $\mathcal{M}^*_{\mathcal{D}}$ denote the set of all maximum matchings in $\mathcal{M}_{\mathcal{D}}$.

**Lemma 4.11** *Let $x \in \hat{P}_1(\hat{\epsilon}_1)$. Then*

*(i)* $x = \bar{x}$

*(ii)* $x \leq w$ *on* $\bigcup \mathcal{D}$

*(iii) Each $M \in \mathcal{M}^*_{\mathcal{D}}$ is $x$-tight.*

**Proof**: Let $x \in \hat{P}_1(\hat{\epsilon}_1)$. We first prove (ii) and (iii) for $\bar{x}$ and then show that $x = \bar{x}$.

(ii) $\bar{x} \leq w$ on $\bigcup \mathcal{D}$, or equivalently, $\bar{x}_D \leq w_D$ for all $D \in \mathcal{D}$:
Suppose to the contrary that $\bar{x}_D > w_D \geq 0$ for some odd component $D \in \mathcal{D}_{max}$. Then $\bar{x}_i > w_i \geq 0$ for all $i \in \bigcup \mathcal{D}_{max}$.

We first consider the case $T \cup \bigcup \mathcal{C} = \emptyset$. If $\mathcal{D}_{max} = \mathcal{D}$, we had $\bar{x} > w_D$ for all $D \in \mathcal{D}$ and hence $\bar{x}(N) > v(N)$, a contradiction. Hence $\mathcal{D}_{max} \subset \mathcal{D}$. By Lemma 4.10 we may decrease $\bar{x}$ slightly and uniformly on $\bigcup \mathcal{D}_{max}$ and increase $\bar{x}$ on $\bigcup \mathcal{D} \backslash \bigcup \mathcal{D}_{max}$ resulting in some $\bar{\bar{x}} \in \hat{P}_1(\hat{\epsilon}_1)$ for which no $M \in \mathcal{M}^* \cup \mathcal{M}_{\mathcal{D}}$ is tight. This contradicts the optimality of $\hat{\epsilon}_1$.

Now suppose $T \cup \bigcup \mathcal{C} \neq \emptyset$. If $\mathcal{D}_{max} = \mathcal{D}$, we had $\bar{x}_D > w_D$ for all $D \in \mathcal{D}$ and hence no $M \in \mathcal{M}_{\mathcal{D}}$ were $\bar{x}$-tight. By Lemma 4.6, $\hat{\epsilon}_1 < 0$ is not optimal, a contradiction.

If $D_{max} \subset \mathcal{D}$, we proceed as follows. Chose $\delta > 0$ sufficiently small and let $\bar{x}^\delta$ arise from $\bar{x}$ by

- decreasing $\bar{x}_i$ by $\dfrac{\delta}{|D|}$ $\quad (i \in D, D \in \mathcal{D}_{max})$
- increasing $\bar{x}$ on $T$ by $\delta'$ in total, where $(|\mathcal{D}_{max}| - 1)\delta < \delta' < |\mathcal{D}_{max}|\delta$
- increasing $\bar{x}$ uniformly on $\bigcup \mathcal{D} \setminus \bigcup \mathcal{D}_{\max}$ by $|\mathcal{D}_{max}|\delta - \delta'$ in total.

For sufficiently small $\delta > 0$ the resulting $\bar{x}^\delta$ has $\bar{x}^\delta(M) > \bar{x}(M)$ for each $\bar{x}$-tight $M \in \mathcal{M}_{\mathcal{D}}$ (since none of these meets $\mathcal{D}_{max}$) and $\bar{x}^\delta(M) > \bar{x}(M)$ for all $\bar{x}$-tight $M \in \mathcal{M}^*$ by Lemma 4.10. Hence, again $\bar{x}^\delta \in \hat{P}_1(\hat{\epsilon}_1)$ has no tight matchings, contradicting the optimality of $\hat{\epsilon}_1$.

(iii) Each $M \in \mathcal{M}^*_{\mathcal{D}}$ is $\bar{x}$-tight: First note that for any $M_1, M_2 \in \mathcal{M}^*_{\mathcal{D}}$,

$$\bar{x}(M_1) - w(M_1) = \bar{x}(M_2) - w(M_2) = \sum_{D \in \mathcal{D}} (|D| - 1)(\bar{x}_D - w_D).$$

Since $\bar{x}_D \leq w_D$ for all $D \in \mathcal{D}$, each $M \in \mathcal{M}^*_{\mathcal{D}}$ minimizes $\bar{x}(M) - w(M)$ over $\mathcal{M}_{\mathcal{D}}$. Then the claim follows from Lemma 4.6.

(i) $x = \bar{x}$:
For each $D \in \mathcal{D}$ we chose a node $i \in \mathcal{D}$ with $x_i - w_i = \max\{x_j - w_j \mid j \in D\}$ and a near-perfect matching covering $D \setminus i$. Let $M \in \mathcal{M}^*_{\mathcal{D}}$ be the union of all these near-perfect matchings. By construction we have $x(M) - w(M) \leq \bar{x}(M) - w(M)$ with equality if and only if $x \equiv \bar{x}$ on $\bigcup \mathcal{D}$. But since $M$ is $\bar{x}$-tight,

$$x(M) - w(M) < \bar{x}(M) - w(M) = \hat{\epsilon}_1$$

would contradict $x \in \hat{P}_1(\hat{\epsilon}_1)$. □

**Lemma 4.12** *Let $x = \bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$. If $x(T \cup \bigcup \mathcal{C}) = 0$, then every $M \in \mathcal{M}^*$ is $x$-tight.*

**Proof**: If $T \cup \bigcup \mathcal{C}$ is empty, then $\mathcal{M}^* = \mathcal{M}^*_{\mathcal{D}}$ and the claim follows by Lemma 4.11(iii). Suppose $T \cup \bigcup \mathcal{C}$ is nonempty and $x(T \cup \bigcup \mathcal{C}) = 0$. Recall that any $M \in \mathcal{M}^*$ decomposes as

$$M = M_{\mathcal{C}} \cup M_{T,\mathcal{D}} \cup M_{\mathcal{D}}$$

with $M_{\mathcal{C}}$ a perfect matching of $\bigcup \mathcal{C}$, $M_{T,\mathcal{D}}$ matching $T$ completely into $\bigcup \mathcal{D}$ and $M_{\mathcal{D}} \in \mathcal{M}^*_{\mathcal{D}}$. Since $M_{\mathcal{D}}$ is $x$-tight (cf. Lemma 4.11(iii)), we have $x(M_{\mathcal{D}}) = w(M_{\mathcal{D}}) + \hat{\epsilon}_1$. Together with $x(M) \geq w(M) + \hat{\epsilon}_1$, this implies

$$x(M_{\mathcal{C}} \cup M_{T,\mathcal{D}}) \geq w(M_{\mathcal{C}}) + w(M_{T,\mathcal{D}}).$$

Let $B \subset \bigcup \mathcal{D}$ be the set of nodes that is joined to the nodes in $T$ by $M_{T,\mathcal{D}}$. Then $w(M_{\mathcal{C}}) + w(M_{T,\mathcal{D}}) = w(T \cup \bigcup \mathcal{C}) + w(B)$. By Lemma 4.11(ii), $x_i \leq w_i$ for all $i \in B$. Since we assumed that $x(T \cup \bigcup \mathcal{C}) = 0$, we have

$$w(B) + w(T \cup \bigcup \mathcal{C}) \leq x(B) \leq w(B).$$

Then $w \equiv 0$ on $T \cup \bigcup \mathcal{C}$ and $x_i = w_i$ on $B$. Hence we deduce that

$$\begin{aligned} x(M) &= x(M_{\mathcal{C}}) + x(T) + x(B) + x(M_{\mathcal{D}}) \\ &= w(B) + w(M_{\mathcal{D}}) + \hat{\epsilon}_1 \\ &= w(M) + \hat{\epsilon}_1. \end{aligned}$$

□

**Lemma 4.13** *Let $x = \bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$. Then there is some $x$-tight $M \in \mathcal{M}^*$. Moreover, if $D \in \mathcal{D}_{\max}$ or $|D| > 1$ then there is some $x$-tight $M \in \mathcal{M}^*$ not covering $D$.*

**Proof**: If $\bar{x}(T \cup \bigcup \mathcal{C}) = 0$ then, according to Lemma 4.12, every $M \in \mathcal{M}^*$ is $x$-tight. Since for each $D \in \mathcal{D}$ there is some matching $M \in \mathcal{M}^*$ not covering $D$, we have finished this case.

Assume that $T \cup \bigcup \mathcal{C} \neq \emptyset$ and $\bar{x}(T \cup \bigcup \mathcal{C}) > 0$. Suppose $\bar{x}(M) > w(M) + \hat{\epsilon}_1$ for all $M \in \mathcal{M}^*$. Then we could do as follows: Decrease (somehow) $\bar{x}$ on $T \cup \bigcup \mathcal{C}$ and increase $\bar{x}$ uniformly on $\bigcup \mathcal{D}$ by the same total (sufficiently small) amount. The resulting $\bar{\bar{x}}$ were still in $\hat{P}_1(\hat{\epsilon}_1)$ and would contradict Lemma 4.11(iii).

By Lemma 4.10 this implies that each $D \in \mathcal{D}_{\max}$ is left uncovered by some $\bar{x}$-tight $M \in \mathcal{M}^*$. We are left to prove a corresponding result for $D \in \mathcal{D}$ with

$|D| > 1$. Assume that $D \in \mathcal{D} \backslash \mathcal{D}_{\max}$ and $|D| > 1$. Then $\bar{x} < w$ on $D$ by Lemma 4.11(ii), so every $\bar{x}$-tight $M \in \mathcal{M}^*_{\mathcal{D}}$ contains a near-perfect matching of $D$. Now suppose $D$ is covered by every $\bar{x}$-tight $M \in \mathcal{M}^*$. Since $\bar{x}(T \cup \bigcup \mathcal{C}) > 0$, we may decrease $\bar{x}$ slightly on $T \cup \mathcal{C}$ and increase $x$ uniformly on $D$ by the same (sufficiently small) total amount. The resulting $\bar{\bar{x}}$ would again be in $\hat{P}_1(\hat{\epsilon}_1)$ and contradict Lemma 4.11(iii). This finishes the proof. □

We call an allocation $x = \bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$ *flexible* if the conclusion of Lemma 4.13 holds with respect to all $D \in \mathcal{D}$, i.e., if each $D \in \mathcal{D}$ is left uncovered by some $\bar{x}$-tight $M \in \mathcal{M}^*$.

**Lemma 4.14** *Flexible allocations exist.*

**Proof**: Let $x = \bar{x} \in P_1(\hat{\epsilon}_1)$. Suppose $\bar{x}$ is not already flexible. Then there exists a component $D = \{i\} \in D$ of size 1 such that every $\bar{x}$-tight $M \in \mathcal{M}^*$ covers $i$. Maximum matchings not covering $i$ are not tight. In particular, this implies that $T \neq \emptyset$ and, by Lemma 4.12, $\bar{x}(T \cup \bigcup \mathcal{C}) > 0$.

We may thus increase $\bar{x}_i$ and decrease $\bar{x}$ on $T \cup \bigcup \mathcal{C}$ by the same total amount $\delta$ until $\bar{x}$ becomes "flexible" with respect to $D = \{i\}$. By Lemma 4.12 we know that this will happen before $\delta$ exceeds the value $\bar{x}(T \cup \bigcup \mathcal{C})$. In other words, we choose $\delta > 0$ maximal such that the modification $\bar{x}^\delta$ is still in $\hat{P}_1(\hat{\epsilon}_1)$. Then $\bar{x}^\delta(M) = w(M) + \hat{\epsilon}_1$ holds for a matching $M \in \mathcal{M}^*$ that does not cover $i$ (and is not $\bar{x}$-tight). Because all matchings in $\mathcal{M}^*$ that were already $\bar{x}$-tight (and cover $i$) remain tight, the claim follows by induction. □

We are now ready to determine the structure of $x$-tight matchings in $\mathcal{M}^*$ for flexible $x = \bar{x} \in \hat{P}_1(\hat{\epsilon}_1)$. Suppose $\hat{x} \in \hat{P}(\hat{\epsilon}_1)$ is a given flexible allocation. Suppose that $\alpha_0 < \ldots < \alpha_p$ $(p \geq 0)$ are the different values $\hat{x} - w$ takes on $\bigcup \mathcal{D}$ and let

$$\mathcal{D} = \mathcal{D}_0 \cup \ldots \cup \mathcal{D}_p$$

be the corresponding partition of $\mathcal{D}$. Hence $\hat{x} - w \equiv \alpha_i$ on $\bigcup \mathcal{D}_i$ and $\mathcal{D}_p = \mathcal{D}_{\max}$.

**Proposition 4.3** *There exists a partition $T = T_0 \cup \ldots \cup T_p$ (with some of the $T_i$ possibly empty) such that $M \in \mathcal{M}^*$ is $\hat{x}$-tight if and only if $M$ matches each $T_i$ into $\mathcal{D}_i$.*

**Proof**: If $T = \emptyset$, the claim is true in the sense that nothing is matched into $\mathcal{D}$ and each $M \in \mathcal{M}^*$ is $\hat{x}$-tight. (By Lemma 4.13, some $\hat{x}$-tight $M \in \mathcal{M}^*$ exists and since $T = \emptyset$, all $M \in \mathcal{M}^*$ have the same $\hat{x}$-value.)

In general, recall that $\hat{x}$-tight matchings in $\mathcal{M}^*$ are exactly those that minimize $\hat{x}(M) - w(M)$ over $\mathcal{M}^*$. For given $\hat{x}$, the value $\hat{x}(M) - w(M)$ only depends on how many nodes of $T$ are matched into each $\mathcal{D}_i$. (This readily follows from the decomposition $M = M_{\mathcal{C}} \cup M_{T,\mathcal{D}} \cup M_{\mathcal{D}}$.) In other words, $\hat{x}(M) - w(M)$ only depends on the total $(\hat{x} - w)$-weight of nodes in $\bigcup \mathcal{D}$ that are matched with $T$. The claim therefore follows from Lemma 4.15 below. □

**Lemma 4.15** *Consider a bipartite graph $G$ with node classes $A$ and $B$. Suppose $B = B_0 \cup \ldots \cup B_p$ is a partition of $B$ and edges incident with $B_i$ have weight $\alpha_i$ ($\alpha_0 < \ldots < \alpha_p$). Assume that the set $\mathcal{M}^*$ of matchings that completely match $A$ into $B$ is nonempty and let $\mathcal{M}^*_{\min}$ be the set of $M \in \mathcal{M}^*$ with minimum weight. Suppose finally, that $\mathcal{M}^*_{\min}$ is "flexible" in the sense that each $b \in B$ is left uncovered by some $M \in \mathcal{M}^*_{\min}$. Then there is a partition $A = A_0 \cup \ldots \cup A_p$ of $A$ such that $M \in \mathcal{M}^*_{\min}$ if and only if $M$ matches $A_i$ into $B_i$ ($i = 0, \ldots, p$).*

**Proof**: Let $\mathcal{M}^*_0$ denote the set of maximum matchings in the subgraph $G_0$ induced by $A \cup B_0$. Each $M \in \mathcal{M}^*_{\min}$ induces a maximum matching $M_0 \subseteq M$ in $\mathcal{M}^*_0$. This can be seen as follows: Suppose $M$ does not induce a matching $M_0 \in \mathcal{M}^*_0$. Let $M'_0$ be a maximum matching in $\mathcal{M}^*_0$. For all nodes $b \in B_0$ that are covered by $M'_0$ but not by $M$ we do as follows. Let $P \subseteq M \cup M'_0$ be the unique maximal alternating path starting in node $b \in B_0$ uncovered by $M$ and ending in a node $a$ uncovered by $M'_0$. Reverse $M$ along $P$. Then in the end this results in a matching $M'$ with $w(M') < w(M)$, a contradiction. Hence we must have

$$(*) \quad \text{Each } b \in B_0 \text{ is left uncovered by some } M_0 \in \mathcal{M}^*_0.$$

Suppose $m^*_0$ is the maximum size of a matching in $G_0$. As $G_0$ is bipartite, Theorem 4.1 ensures the existence of a (minimum) node cover $A^*_0 \cup B^*_0$ ($A^*_0 \subseteq A$, $B^*_0 \subseteq B$) of size $m^*_0$. Since $|A^*_0 \cup B^*_0| = |M|$ for a matching $M \in \mathcal{M}^*_0$, each $M \in \mathcal{M}^*_0$ is incident with all nodes in $A^*_0 \cup B^*_0$. Hence, by (*) we conclude that $B^*_0 = \emptyset$. In other words, each $M \in \mathcal{M}^*_{\min}$ matches $A^*_0$ into $B_0$. Now let $\mathcal{M}^*_1$ denote the set of maximum matchings in the subgraph $G_1$ induced by $A \backslash A^*_0 \cup B_1$, and continue in the same way. So the claim follows by induction. □

We are now prepared to present our main result, a simple alternative description of the least core. Consider the LP

$$
\begin{array}{lllll}
(\hat{\hat{P}}_1) \quad \max & \epsilon & & & \\
\quad s.t. & x & = & \bar{x} & \\
& x_i & \leq & w_i & (i \in \bigcup \mathcal{D}) \\
& x(e) & \geq & w(e) & (e \in E \backslash E(\bigcup \mathcal{D})) \\
& x(N) & = & v(N) & \\
& x(M) & \geq & w(M) + \epsilon & (M \in \mathcal{M}^*_{\mathcal{D}}) \\
& x & \geq & 0. &
\end{array}
$$

Note that $x \equiv \bar{x}$ is just a shorthand notation for a number of linear equalities of the type $x_i - w_i = x_j - w_j$. Furthermore, note that for $x \equiv \bar{x}$ the value $x(M) - w(M)$ is independent of the particular choice of $M \in \mathcal{M}^*_{\mathcal{D}}$. Hence the exponentially many constraints for $M \in \mathcal{M}^*_{\mathcal{D}}$ reduce to one single inequality.

Again, we let $\hat{\hat{P}}_1(\epsilon) := \{x \mid (x, \epsilon) \text{is feasible for} (\hat{\hat{P}}_1)\}$ and denote the optimum value of $(\hat{\hat{P}}_1)$ by $\hat{\hat{\epsilon}}_1$.

**Theorem 4.9** *We have* $\epsilon_1 = \hat{\epsilon}_1 = \hat{\hat{\epsilon}}_1$ *and* $\mathit{leastcore}(N, v) = P_1(\epsilon_1) = \hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1)$.

**Proof**:

- We have $\epsilon_1 \leq \hat{\epsilon}_1$ by definition.
- $\hat{\epsilon}_1 \leq \hat{\hat{\epsilon}}_1$: Let $\hat{x} \in \hat{P}_1(\hat{\epsilon}_1)$ be flexible with corresponding partitions $\mathcal{D} = \mathcal{D}_0 \cup \ldots \cup \mathcal{D}_p$ and $T = T_0 \cup \ldots \cup T_p$. Define $\hat{\hat{x}} \in \mathbb{R}^N$ by

$$
\hat{\hat{x}}_i \;=\; \begin{cases} w_i & \text{if } i \in \bigcup \mathcal{C} \\ \hat{x}_i & \text{if } i \in \bigcup \mathcal{D} \\ w_i - \alpha_j & \text{if } i \in T_j \quad (0 \leq j \leq p). \end{cases}
$$

We show that $\hat{\hat{x}} \in \hat{\hat{P}}_1(\hat{\epsilon}_1)$ (proving that $\hat{\hat{\epsilon}}_1 \geq \hat{\epsilon}_1$). The only non-trivial constraints to check are $\hat{\hat{x}}(N) = v(N)$ and $\hat{\hat{x}}(e) \geq w(e)$ for $e \in E \backslash E(\bigcup \mathcal{D})$. All other constraints directly follow from Lemma 4.11.

Let $M \in \mathcal{M}^*$ be $\hat{x}$-tight and decompose it as

$$
M = M_{\mathcal{C}} \cup M_{T,\mathcal{D}} \cup M_{\mathcal{D}}
$$

as usual. Since $M_{\mathcal{D}} \in \mathcal{M}^*_{\mathcal{D}}$ is also $\hat{x}$-tight by Lemma 4.11, we conclude that $\hat{x}(M_{\mathcal{C}} \cup M_{T,\mathcal{D}}) = w(M_{\mathcal{C}}) + w(M_{T,\mathcal{D}}) = \hat{\hat{x}}(M_{\mathcal{C}} \cup M_{T,\mathcal{D}})$ by definition of $\hat{\hat{x}}$. Hence $\hat{\hat{x}}(N) = \hat{x}(N) = v(N)$.

Secondly, let us consider $e \in E \backslash E(\bigcup \mathcal{D})$. If $e \in E(T \cup \bigcup \mathcal{C})$ then $\hat{\hat{x}}(e) \geq w(e)$ by definition of $\hat{\hat{x}}$. (Recall that $\alpha_j = \hat{x}_i - w_i \leq 0$ for all $i \in \bigcup \mathcal{D}_j$.) Thus we are left with edges between $T$ and $\bigcup \mathcal{D}$. Suppose $\hat{\hat{x}}(e) < w(e)$ for such an edge joining, say, $D \in \mathcal{D}_i$ with $k \in T_j$. Then $\hat{\hat{x}}(e) < w(e)$ implies $\alpha_i < \alpha_j$. Since $\hat{x}$ is flexible, there exists an $\hat{x}$-tight matching $M \in \mathcal{M}^*$ not covering $D$. Since $D$ is factor-critical (and $\hat{x} - w$ is constant on $D$), we may assume that $M$ does not match the end point of $e$ in $D$. Since $M$ is $\hat{x}$-tight, $k \in T_j$ is matched into $D_j$ by some edge $f \in M$ (cf. Proposition 4.3). But then $M' = M \backslash f \cup e$ has $\hat{x}(M') - w(M') = \hat{x}(M) - w(M) + \alpha_i - \alpha_j < \hat{x}(M) - w(M) = \hat{\epsilon}_1$, a contradiction.

- $\hat{\hat{\epsilon}}_1 \leq \epsilon_1$: We show that in general $\hat{\hat{P}}_1(\epsilon) \subseteq P_1(\epsilon)$. Suppose $x \in \hat{\hat{P}}_1(\epsilon)$. Then $x(M) \geq w(M) + \epsilon$ for all $M \in \mathcal{M}^*_{\mathcal{D}}$. Since $x \leq w$ on $\bigcup \mathcal{D}$, this also implies $x(M) \geq w(M) + \epsilon$ for all $M \in \mathcal{M}_{\mathcal{D}}$. (Use an augmenting path argument.) Since $x(e) \geq w(e)$ for all $e \in E \backslash E(\bigcup \mathcal{D})$, we further conclude that $x(M) \geq w(M) + \epsilon$ for all $M \in \mathcal{M}$.

- Finally, we verify that $P_1(\epsilon_1) = \hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1)$. We have just proved that "$\supseteq$" holds. Conversely let $x \in P_1(\epsilon_1)$. Then $x \in \hat{P}_1(\hat{\epsilon}_1)$ and by Lemma 4.11 $x$ satisfies all constraints of $\hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1)$ except possibly $x(e) \geq w(e)$ for $e \in E \backslash E(\bigcup \mathcal{D})$. Thus let $e \in E \backslash E(\bigcup \mathcal{D})$. Pick $M \in \mathcal{M}^*_{\mathcal{D}}$ not covering the end point of $e$ in $\bigcup \mathcal{D}$, so that $M \cup e$ is a matching again. Then, since $x \in P_1(\epsilon_1)$, we have $x(M \cup e) \geq w(M) + w(e) + \epsilon_1$, and since $M \in \mathcal{M}^*_{\mathcal{D}}$ is $x$-tight, we have $x(M) = w(M) + \hat{\epsilon}_1$. Since $\epsilon_1 = \hat{\epsilon}_1$, the claim follows. □

As mentioned before, the least core of a node matching game can also be described by the polyhedron $\hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1) = \hat{\hat{P}}_1(0)$ if the core is nonempty.

For general matching games it is not possible to characterize the least core by a polyhedron of the form $\hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1)$. The following example shows that already for a node matching game with *negative* node weights this is not possible.

**Example 4.7** Consider the matching game $(N, v)$ obtained from a graph $G$ with edge weighting $w$ as shown in Figure 4.7. This edge weighting can be obtained from only one node weighting, namely $w$ given by $w_1 = w_2 = -1$, $w_3 = 2$ and $w_4 = w_5 = 3$. So $w_1$ and $w_2$ are strictly negative.

The set $T = \{3\}$ is the Tutte set satisfying conditions (i) and (ii) of Theorem 4.4. Both odd components of $G \backslash T$, $D_1 = \{1\}$ and $D_2 = \{2\}$, contain only

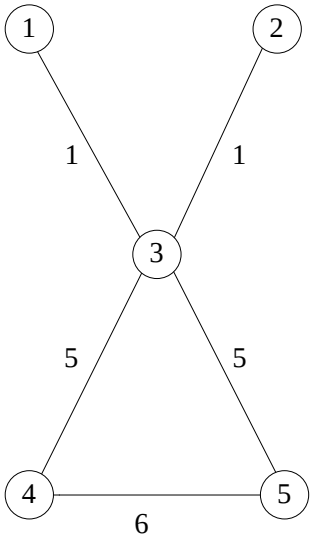

Figure 4.7

one node. However, core$(N, v)$ is empty and it is immediately clear that any allocation $x$ in the least core has $x(e) < 1$ for at least one edge $e \in E$. □

### 4.3.2 The nucleolus of node matching games

Recall from Section 2.3 that the nucleolus can be computed by solving the following sequence of linear programs:

$$\begin{array}{llrcl} (P_1) & \max & \epsilon & & \\ & s.t. & x(S) & \geq & v(S)+\epsilon \quad (S \notin \{\emptyset, N\}) \\ & & x(N) & = & v(N) \end{array}$$

with optimum value $\epsilon_1$,

$$\begin{array}{llrcl} (P_2) & \max & \epsilon & & \\ & s.t. & x & \in & P_1(\epsilon_1) \\ & & x(S) & \geq & v(S)+\epsilon \quad (S \notin Fix\, P_1(\epsilon_1)\ ) \end{array}$$

with optimum value $\epsilon_2$, etc. until the nucleolus $\eta(N, v)$ is finally determined as the unique solution $x^*$, $\epsilon^* = \epsilon_r$ of

$$\begin{array}{llrcl} (P_r) & \max & \epsilon & & \\ & s.t. & x & \in & P_{r-1}(\epsilon_{r-1}) \\ & & x(S) & \geq & v(S)+\epsilon \quad (S \notin Fix\, P_{r-1}(\epsilon_{r-1})\ ). \end{array}$$

By Theorem 4.9 $(P_1)$ is equivalent to $(\hat{\hat{P}}_1)$ in the sense that they define the same set of optimal solutions. As we shall see, similar equivalent formulations can be found for $(P_k)$, $k \geq 2$. Define recursively

$$
\begin{array}{llcll}
(\hat{\hat{P}}_k) & \max & \epsilon & & \\
& s.t. & x & \in & \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1}) \\
& & x(e) & \leq & w(e) + \epsilon_1 - \epsilon \quad (e \in E(\bigcup \mathcal{D}), e \notin Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1}) ) \\
& & x(e) & \geq & w(e) - \epsilon_1 + \epsilon \quad (e \in E \backslash E(\bigcup \mathcal{D}), e \notin Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1}) ) \\
& & x_i & \geq & -\epsilon_1 + \epsilon \qquad (i \in N, i \notin Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1}) ).
\end{array}
$$

As before, let $\hat{\hat{\epsilon}}_k$ denote the optimum value of $(\hat{\hat{P}}_k)$ and define $\hat{\hat{P}}_k(\epsilon)$ in the obvious way.

**Theorem 4.10** *We have $\epsilon_k = \hat{\hat{\epsilon}}_k$ and $P_k(\epsilon_k) = \hat{\hat{P}}_k(\hat{\hat{\epsilon}}_k)$ for $k = 1, \ldots, r$. In particular, $\hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1) \supset \ldots \supset \hat{\hat{P}}_r = \{x^*\}$ defines the nucleolus $\eta(N, v)$.*

**Proof**: For $k = 1$ the claim is equivalent to Theorem 4.9. We proceed by induction on $k$. Assume that $\epsilon_{k-1} = \hat{\hat{\epsilon}}_{k-1}$ and $P_{k-1}(\epsilon_{k-1}) = \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1})$. The induction step amounts to show the following two things.

(i) $P_k(\epsilon) \subseteq \hat{\hat{P}}_k(\epsilon)$ (implying that $\hat{\hat{\epsilon}}_k \geq \epsilon_k$) :
Let $x \in P_k(\epsilon)$. Then $x \in P_1(\epsilon_1) = \hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1)$, so $x$ satisfies $x \geq 0$, $x_i = x_D - w_D + w_i \leq w_i$ for all $i \in D$, $D \in \mathcal{D}$, and $x(M) = w(M) + \epsilon_1$ for all $M \in \mathcal{M}^*_{\mathcal{D}}$. We first consider $e \in E \backslash E(\bigcup \mathcal{D})$ and show that $x(e) \geq w(e) - \epsilon_1 + \epsilon$ unless $e \in Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1}) = Fix\, P_{k-1}(\epsilon_{k-1})$. Choose $M \in \mathcal{M}^*_{\mathcal{D}}$ such that $M \cup e$ is a matching. (Existence follows from the fact that each $D \in \mathcal{D}$ is factor-critical.) Since $M$ is fixed by $\hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1) = P_1(\epsilon_1)$, it is fixed by $\hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1})$. Hence $e \in Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1})$ if and only if $M \cup e \in Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1})$. Since we assume that $e \notin Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1})$, we have $M \cup e \notin Fix\, P_{k-1}(\epsilon_{k-1})$ and thus $x \in P_k(\epsilon)$ implies $x(M \cup e) \geq w(M \cup e) + \epsilon$. Together with $x(M) = w(M) + \epsilon_1$ this yields $x(e) \geq w(e) - \epsilon_1 + \epsilon$.

In the same way we can show that $x_i \geq -\epsilon_1 + \epsilon$ for a node $i \notin Fix\, \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1})$. Next consider $e \in E(\bigcup \mathcal{D})$, say $e \in E(D)$ for $D \in \mathcal{D}$. We show that $x(e) \leq w(e) + \epsilon_1 - \epsilon$ unless $e$ is already fixed by $\hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1}) = P_{k-1}(\epsilon_{k-1})$. Since

$x_i - w_i = x_j - w_j$ for all $i, j \in D$, we conclude that $x(e) - w(e)$ is independent of the particular choice of $e \in E(D)$. Choose any $M \in M^*_{\mathcal{D}}$ and assume without loss of generality that $e \in M \cap E(D)$ is not fixed by $P_{k-1}(\epsilon_{k-1})$. Since $x(M)$ is fixed (to $w(M) + \epsilon_1$), we conclude that $M \backslash e \notin Fix\, P_{k-1}(\epsilon_{k-1})$. Hence $x \in P_k(\epsilon)$ implies $x(M\backslash e) \geq w(M\backslash e) + \epsilon$. Together with $x(M) = w(M) + \epsilon_1$ we get $x(e) \leq w(e) + \epsilon_1 - \epsilon$.

(ii) $\hat{\hat{P}}_k(\epsilon) \subseteq P_k(\epsilon)$ (implying that $\epsilon_k \geq \hat{\hat{\epsilon}}_k$):
Let $x \in \hat{\hat{P}}_k(\epsilon)$. Again this implies $x \in \hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1)$, so $x \geq 0$, $x_i = x_D - w_D + w_i \leq w_i$ for all $i \in D$, $D \in \mathcal{D}$ and $x(M_{\mathcal{D}}) = w(M_{\mathcal{D}}) + \epsilon_1$ for all $M_{\mathcal{D}} \in \mathcal{M}^*_{\mathcal{D}}$.

We are to show that $x(S) \geq v(S) + \epsilon$ for $S \subset N$ not yet fixed by $P_{k-1}(\epsilon_{k-1}) = \hat{\hat{P}}_{k-1}(\hat{\hat{\epsilon}}_{k-1})$. Since $x \geq 0$, we may only consider $S = N(M)$ for $M \in \mathcal{M}$. Furthermore, since $x(e) \geq w(e)$ on $E\backslash E(\bigcup \mathcal{D})$, we may restrict ourselves to $M \subseteq E(\bigcup \mathcal{D})$. Finally, since $x_i = x_D - w_D + w_i$ for all $i \in D$, $D \in \mathcal{D}$, $x(M) - w(M)$ only depends on $|M \cap D|$ for each $D \in \mathcal{D}$. So we may without loss of generality assume that $M \subseteq M_{\mathcal{D}}$ for some $M_{\mathcal{D}} \in \mathcal{M}^*_{\mathcal{D}}$. Assume that $M$ is not fixed by $P_{k-1}(\epsilon_{k-1})$. Since $M_{\mathcal{D}}$ is fixed by $P_{k-1}(\epsilon_{k-1})$, we conclude that $M_{\mathcal{D}}\backslash M$ is not fixed by $P_{k-1}(\epsilon_{k-1})$. So at least some $e \in M_{\mathcal{D}}\backslash M$ is not fixed by $P_{k-1}(\epsilon_{k-1})$. Hence $x \in \hat{\hat{P}}_k(\epsilon)$ implies $x(e) \leq w(e) + \epsilon_1 - \epsilon$. All other edges $f \in M_{\mathcal{D}}\backslash M$ satisfy $x(f) \leq w(f)$ (as $x \leq w$ on $\bigcup \mathcal{D}$). Hence $x(M_{\mathcal{D}}) = w(M_{\mathcal{D}}) + \epsilon_1$ implies $x(M) \geq w(M) + \epsilon$ as required. □

Clearly, the number of constraints in each linear program $(\hat{\hat{P}}_k)$ is bounded by a polynomial in $|N|$. The size of the parameters $\hat{\hat{\epsilon}}_k$ $(1 = 1, \ldots, r)$ is bounded by a polynomial in $< N, v >$ (cf. Remark 2.1). Then we can conclude that

**Corollary 4.4** *The nucleolus of a node matching game $(N, v)$ can be computed in polynomial time.* □

We end this section by giving an explicit formula for the nucleolus in case the graph $(N, E)$ with node weighting $w$ satisfies some extra conditions. First consider the following observation.

**Proposition 4.4** *Let $i, j$ be two nodes in a component $D \in \mathcal{D}$ that are not connected. Let $(N, v')$ be the node matching game obtained from $(N, v)$ by adding the edge $(i, j)$ to $G$. Then $\eta(N, v') = \eta(N, v)$.*

**Proof**: The proof is straightforward, noting the fact that for all allocations $x \in P_1(\epsilon_1) = \hat{\hat{P}}_1(\hat{\hat{\epsilon}}_1)$

$$x_k - w_k = x_l - w_l \text{ for all } k, l \in D.$$

So adding the edge $(i, j)$ does not yield any new equations in the linear programs $(\hat{\hat{P}}_k)$ $(k = 1, \ldots, r)$. □

Now suppose $T \cup \bigcup \mathcal{C}$ is empty. Let

$$\mathcal{D} = \mathcal{S}_1 \cup \ldots \cup \mathcal{S}_l$$

be the partition of $\mathcal{D}$ such that all components $D \in \mathcal{D}$ with equal size $|D|$ are in the same subcollection $\mathcal{S}_i$, and $|D| < |D'|$ for $D \in \mathcal{S}_i$, $D' \in \mathcal{S}_j$ and $1 \leq i < j \leq l$.

We say that a node weighting is *symmetric* if we can order the nodes in any two components $D, D' \in \mathcal{D}$ with $|D| = |D'|$ as $i_1, \ldots, i_{|D|}$ and $j_1, \ldots, j_{|D|}$ such that

$$w_{i_r} = w_{j_r} \quad (r = 1, \ldots, |D|).$$

Below we give an expression for the nucleolus in case $T \cup \bigcup \mathcal{C}$ is empty and $w$ is symmetric.

**Theorem 4.11** *Let $T \cup \bigcup \mathcal{C}$ be empty and let $w : N \to \mathbb{R}_+$ be symmetric. Then the nucleolus of $(N, v)$ is equal to the allocation $x^* \in \mathbb{R}^N$ given by*

$$x_i^* = \begin{cases} w_i - w_{min}^D & \text{if } i \in \bigcup \mathcal{S}_1 \cup \ldots \cup \bigcup \mathcal{S}_{k-1} \\ w_i - \alpha & \text{if } i \in \bigcup \mathcal{S}_k \\ w_i & \text{if } i \in \bigcup \mathcal{S}_{k+1} \cup \ldots \cup \bigcup \mathcal{S}_l, \end{cases}$$

*where $1 \leq k \leq l$ is taken as small as possible in such a way that a number $\alpha \geq 0$ exists with the property that $x^* \in \mathbb{R}^N$ is of the form above, $x^* \geq 0$ and $x^*(N) = v(N)$.*

**Proof**: First suppose core$(N, v)$ is nonempty. Let $x \in$ core$(N, v)$. By Theorem 4.6 $x_i = 0$ if $i \in N$ is an isolated node. By Proposition 4.2 $x_i = w_i$ holds for all $D \in \mathcal{D}$ with $|D| > 1$. So core$(N, v)$ contains exactly one allocation, which must be the nucleolus. Choose $\alpha = 0$. If a component $D \in \mathcal{S}_1$ has size $|D| > 1$ then let $k = 1$. Otherwise let $k = 2$.

Suppose core$(N, v)$ is empty. As a first step in computing the nucleolus we have to determine the optimal value $\epsilon_1$ of $(P_1) = (\hat{\hat{P}}_1)$. Note that maximizing $\epsilon$ comes down to maximizing

$$\sum_{D \in \mathcal{D}} (|D| - 1) x_D.$$

So we maximize $x_D$ on components $D$ with $|D|$ as large as possible in such a way that the constraints of $(\hat{\hat{P}}_1):\ x(N) = v(N),\ x = \bar{x},\ x_D \leq w_D\ \ (D \in \mathcal{D})$ and $x \geq 0$, are met. Then it is clear that in this way $k$ has to be chosen as small as possible such that the resulting vector $x^*$ has

$$\begin{array}{rcll} x_i^* & = & w_i - w_{min}^D & \text{if } i \in \bigcup \mathcal{S}_1 \cup \ldots \cup \bigcup \mathcal{S}_{k-1} \\ x_i^* & = & w_i & \text{if } i \in \bigcup \mathcal{S}_{k+1} \cup \ldots \cup \bigcup \mathcal{S}_l, \end{array}$$

and $x^*(\bigcup \mathcal{S}_k) = x^*(N) - x^*(N \backslash \bigcup \mathcal{S}_k)$ such that $x_i^* - w_i = x_j^* - w_j$ for all $i, j \in D$, $D \in \mathcal{S}_k$ and $x^*(\bigcup \mathcal{S}_k) \geq \sum_{D \in \mathcal{S}_k}(w(D) - |D| w_{min}^D)$ (then $x^* \geq 0$). Also the nucleolus is of the same form as $x^*$.

By Proposition 4.4 we may without loss of generality assume that each $D \in \mathcal{D}$ is a complete graph. Suppose $D, D' \in \mathcal{S}_k$. Order the nodes of $D$ as $i_1, \ldots, i_{|D|}$ and the nodes of $D'$ as $j_1, \ldots, j_{|D|}$ such that

$$w_{i_r} = w_{j_r} \quad (r = 1, \ldots, |D|).$$

Recall that the nucleolus satisfies *Sym*. Define a permutation $\pi : N \to N$ as follows:

$$\pi(k) \ = \ \begin{cases} j_r & \text{if } k = i_r \quad (r = 1, \ldots, |D|) \\ i_r & \text{if } k = j_r \quad (r = 1, \ldots, |D|) \\ k & \text{otherwise.} \end{cases}$$

Then $v^\pi(S) = v(S)$ for all $S \subseteq N$. If we want the nucleolus to be equal to $x^*$ then, according to *Sym*, $x^*$ must be chosen such that $x_{i_r}^* = x_{j_r}^*$ for $1 \leq r \leq |D|$. This implies that for all $i \in \bigcup \mathcal{S}_k$ $x_i^* = w_i - \alpha$ for some $0 \leq \alpha \leq w_i$. □

### 4.3.3 Cardinality matching games

As mentioned before, cardinality matching games are a subclass of the class of node weight matching games with node weighting equal to a half for all nodes.

Below we assume that $(N, v)$ is a cardinality matching game defined by a graph $(N, E)$. Again $T \subseteq N$ is the Tutte set satisfying the conditions (i) and (ii) in Theorem 4.4. Since $w \equiv \frac{1}{2} > 0$, Theorem 4.8 reduces to the following simple observation (for a characterization, see also Deng, Ibaraki and Nagamochi [1999]).

**Corollary 4.5** *The cardinality matching game $(N, v)$ has nonempty core if and only if $|D| = 1$ for all $D \in \mathcal{D}$.* □

Because all nodes $i \in N$ have the same positive weight $w_i = \frac{1}{2}$, the set $\tilde{\mathcal{M}}^*$ coincides with the set $\mathcal{M}^*$. Furthermore $w_D = \frac{1}{2}$ for all $D \in \mathcal{D}$. Then the linear program $(\hat{\hat{P}}_1)$ describing the least core can be written as

$$
\begin{array}{lllll}
(\hat{\hat{P}}_1) & \max & \epsilon & & \\
& s.t. & x_i = x_D & & (i \in D, D \in \mathcal{D}) \\
& & x_i \leq \frac{1}{2} & & (i \in \bigcup \mathcal{D}) \\
& & x(e) \geq 1 & & (e \in E \backslash E(\bigcup \mathcal{D})) \\
& & x(N) = v(N) & & \\
& & x(M) \geq |M| + \epsilon & & (M \in \mathcal{M}^*_{\mathcal{D}}) \\
& & x \geq 0, & &
\end{array}
$$

where $v(N) = |\bigcup \mathcal{C}| + |T| + \sum_{D \in \mathcal{D}} \frac{1}{2}(|D| - 1)$. This way we have obtained the characterization of the least core as presented in Kern and Paulusma [2000]. Because $w \equiv \frac{1}{2}$ is symmetric, we can apply Theorem 4.11 (cf. Example 4.5).

**Corollary 4.6** *If $T \cup \bigcup \mathcal{C}$ is empty, then the nucleolus of $(N, v)$ is equal to the allocation $x^* \in \mathbb{R}^N$ given by*

$$
x_i^* = \begin{cases} 0 & \text{if } i \in \bigcup \mathcal{S}_1 \cup \ldots \cup \bigcup \mathcal{S}_{k-1} \\ \frac{1}{2} - \alpha & \text{if } i \in \bigcup \mathcal{S}_k \\ \frac{1}{2} & \text{if } i \in \bigcup \mathcal{S}_{k+1} \cup \ldots \cup \bigcup \mathcal{S}_l, \end{cases}
$$

*where $1 \leq k \leq l$ is taken as small as possible in such a way that a number $\alpha \geq 0$ exists with the property that $x^* \in \mathbb{R}^N$ is of the form above, $x^* \geq 0$ and $x^*(N) = v(N)$.* □

# Chapter 5

# Competition games

We consider a class of games related to sports competitions, in which various teams play matches against each other in pairs according to a previously determined schedule. At some stage of the competition a team may try to bribe some other teams in order to win the competition. Those other teams form the player set in the competition game introduced in Section 5.1. The difficulty here is deciding whether bribing yields the desired result or not. Therefore, in Section 5.2 we first try to solve the problem whether it is possible at all that at some stage of a competition a certain team can end up with the highest final score. The computational complexity of this problem turns out to depend on the way scores are allocated according to the outcome of a match. We determine the complexity for all possible score allocation rules. In Section 5.3 we return to our competition game and give an algorithm to compute the value of a coalition.

## 5.1 Introduction

Consider a sports competition like a national soccer league, in which all participating teams play against each other in pairs (matches) according to a prefixed schedule. Initially all teams have total score zero. When a team participates in a match, its total score is increased by $\alpha \in \mathbb{R}$ if it loses the match, by $\beta \in \mathbb{R}$ if the match ends in a draw, and by $\gamma \in \mathbb{R}$ if it wins the match. We always assume that $\alpha \leq \beta \leq \gamma$ and call the triple $(\alpha, \beta, \gamma)$ the *rule* (score allocation rule) of the competition. In case of a soccer competition, the former FIFA rule was $(\alpha, \beta, \gamma) = (0, 1, 2)$, but this has been changed into the new

rule $(\alpha, \beta, \gamma) = (0, 1, 3)$. Other sports like chess or draughts still use the rule $(\alpha, \beta, \gamma) = (0, 1, 2)$, while stratego, also a strategic board game, has as score allocation rule $(\alpha, \beta, \gamma) = (0, 1, 6)$.

At some stage of the competition a team may ask whether it still has a (theoretical) chance of "winning" the competition, i.e, ending up with the highest final total score. If this is possible it may try to bribe some other participating teams. Here we assume that both teams have to be involved if the end result of a match must be set to a draw and that in case of a win/loss match only the losing team has to be bribed.

We use the following notations. $\bar{N}$ denotes the (finite) *set of teams* participating in the competition. The team in this set that wants to bribe its opponents is denoted by $t_0$. At a certain stage of the competition the position of each team in the ranking is determined by the *current score vector* $s \in \mathbb{R}^{\bar{N}}$, where $s_i$ is the current score of team $i \in \bar{N}$ . The set of remaining matches is denoted by $M$. It is possible that two matches $m_1$ and $m_2$ in $M$ have the same pair of teams $(i, j)$ playing against each other. The *state of a competition* is given by a triple $(\bar{N}, s, M)$.

**Definition 5.1** Given a rule $(\alpha, \beta, \gamma)$ and a state of a competition $(\bar{N}, s, M)$, a *competition game* $(N, v)$ consists of a set $N = \bar{N} \backslash t_0$ and characteristic function $v : 2^N \to \mathbb{R}_+$ that is of the form

$$v(S) \quad = \quad \begin{cases} v^* & \text{if } t_0 \text{ wins the competition after bribing } S \\ 0 & \text{otherwise,} \end{cases}$$

where $v^* \geq 0$ is called the *bribe* of the competition. □

Clearly, a value $v(S)$ does not only depend on the state of the competition but also on its rule. A coalition $S$ is called *influential* if $v(S) = v^*$. In that case $v(S)$ can be interpreted as the price that is agreed by both $t_0$ and coalition $S$ for cheating the competition. If bribing $S$ does not guarantee $t_0$ to win, then $t_0$ has no interest in coalition $S$ and $v(S) = 0$. If another influential coalition is cheaper, then $t_0$ would not do business with $S$. Therefore, all prices are set equal to the same amount $v^*$.

**Example 5.1** Consider some sports competition with rule $(0, 1, 2)$ and state $(\bar{N}, s, M)$ given by the tables below.

| teams | scores |
|---|---|
| $t_0$ | 10 |
| $t_1$ | 9 |
| $t_2$ | 6 |
| $t_3$ | 3 |
| $t_4$ | 2 |

| remaining matches |
|---|
| $t_0 - t_2$ |
| $t_0 - t_3$ |
| $t_3 - t_0$ |
| $t_1 - t_4$ |
| $t_4 - t_1$ |

Then a competition game $(N, v)$ is determined by $N = \{t_1, t_2, t_3, t_4\}$ and $v : 2^N \to \mathbb{R}_+$ given by

$$v(S) = \begin{cases} v^* & \text{if } t_1 \in S \text{ or } t_3 \in S \\ 0 & \text{otherwise.} \end{cases}$$

□

Since we would have a trivial game if $v(N) = 0$, we only consider competition games $(N, v)$ with $v(N) = v^* > 0$. In those competitions team $t_0$ still has a chance to achieve the highest final score. Then the question arises how difficult it is to check whether this condition holds or not. In the next section we solve this problem.

## 5.2 The sports competition problem

### 5.2.1 The model

In this section we determine the computational complexity of the problem

*"Given a competition with state $(\bar{N}, s, M)$ and rule $(\alpha, \beta, \gamma)$ has team $t_0$ still a chance to win the competition?"*

To analyze this we may without loss of generality assume that $t_0$ wins all its remaining matches, resulting in a final total score $\tilde{s}_0$ for $t_0$. The current score $s_i$ of a team $t_i \neq t_0$ only has to be adjusted, if $t_i$ had to play against $t_0$ and $\alpha \neq 0$. As we shall see later on, we may assume $\alpha = 0$. So we still denote the (possibly adjusted) scores of the other teams by $s_i$. The problem is now

whether the teams $t_i \neq t_0$ can finish the remaining matches in such a way that each $t_i$ collects at most $c_i := \tilde{s}_0 - s_i$ additional score points. (We permit the situation that $t_0$ shares the first place in the final ranking with some other teams.)

This can be modeled by a multigraph $G = (V, E)$, whose nodes correspond to teams $t_i \neq t_0$ and edges are in 1-1 correspondence with remaining matches. Each node $i \in V$ has capacity $c_i \in \mathbb{R}$, as defined above. We represent the outcome of a match $e = (i, j)$ by directing the edge from the winner to the loser (and leaving the edge undirected in case of a draw). This way we obtain a *partially oriented graph*, i.e., a graph with an edge set that contains both directed and undirected edges.

Because it will turn out that the computational complexity of our sports competition problem (SC) depends on the rule $(\alpha, \beta, \gamma)$ of the competition, we subdivide our problem into the classes SC$(\alpha, \beta, \gamma)$.

**Definition 5.2** SPORTS COMPETITION (SC$(\alpha, \beta, \gamma)$)

*Instance*: A multigraph $G = (V, E)$ and node capacities $c \in \mathbb{R}^V$.

*Question*: Can $G$ be partially oriented such that for each node $i \in V$:

$$\alpha\delta^-(i) + \beta\delta^0(i) + \gamma\delta^+(i) \leq c_i \ ? \tag{5.1}$$

Here, as usual, $\delta^+$ and $\delta^-$ denote the outdegree and indegree of a node, whereas $\delta^0$ denotes the number of incident unoriented edges. A partial orientation of $G$ satisfying the capacity constraints (5.1) is called a *solution* of the instance $(G, c)$. □

Below we give an example of an instance of SC$(0, 1, 2)$.

**Example 5.2** Consider some sports competition with rule $(0, 1, 2)$ and state $(\tilde{N}, s, M)$ given by the tables below.

| teams | scores |
|---|---|
| $t_0$ | 8 |
| $t_1$ | 7 |
| $t_2$ | 7 |
| $t_3$ | 5 |
| $t_4$ | 3 |

| remaining matches |
|---|
| $t_0 - t_2$ |
| $t_1 - t_2$ |
| $t_2 - t_1$ |
| $t_1 - t_3$ |
| $t_3 - t_4$ |

If we assume that $t_0$ wins its remaining match $t_0 - t_2$, then the final score for $t_0$ is $\tilde{s}_0 = 10$ and the capacity vector $c \in \mathbb{R}^4$ is given by $c = (3, 3, 5, 7)$. The corresponding instance of SC(0,1,2) is the (unoriented) graph $G$ with node capacity vector $c$ (cf. Figure 5.1). Figure 5.2 shows a solution of $(G, c)$. So $t_1$

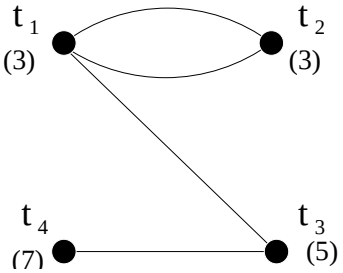


Figure 5.1

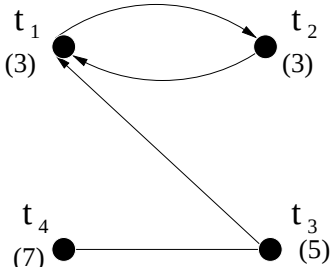


Figure 5.2

beats $t_2$ and vice versa. Furthermore, $t_3$ wins its remaining match against $t_2$, and the match between $t_3$ and $t_4$ ends in a draw. This results in the final table

| teams | scores |
|---|---|
| $t_0$ | 10 |
| $t_1$ | 9 |
| $t_2$ | 9 |
| $t_3$ | 8 |
| $t_4$ | 4 |

Hence at the given stage of the competition it is still possible for team $t_0$ to end up with the highest final total score. □

In the literature a simplified version of the sports competition problem (disallowing draws) is known. In that case the problem reduces to a flow problem, cf. Cook, Cunningham, Pulleyblank and Schrijver [1998] or the section below. As we shall see, however, the question becomes more interesting if draws may occur. Our main result, which has been proved independently by Bernholt, Gülich, Hofmeister and Schmitt [1999], implies that in case $\alpha < \beta < \gamma$ the problem is polynomially solvable if and only if $\alpha + \gamma = 2\beta$ (assuming $\mathcal{P} \neq \mathcal{NP}$). This means that for games like draughts and chess the problem is polynomially solvable.

A lot of sports competitions, such as the FIFA soccer competitions, have the property that each participating team has to play the same number of matches against every other team. We call this type of competitions *closed*. We can split a class SC($\alpha, \beta, \gamma$) into a subclass of closed competitions with rule ($\alpha, \beta, \gamma$) and a subclass of *open competitions*, where the property mentioned above does not hold (e.g., large one- or two-day open tournaments). A couple of years ago the FIFA exchanged the rule $(0, 1, 2)$ for the current rule $(0, 1, 3)$. Therefore we are especially interested in the computational complexity of SC restricted to closed competitions. We show that the complexity does not change under this restriction. This implies that for soccer competitions, by changing the score allocation rule into the rule $(\alpha, \beta, \gamma) = (0, 1, 3)$, the problem has become $\mathcal{NP}$-complete. Also for stratego competitions the problem is $\mathcal{NP}$-complete.

We end our introduction with the following simple observation. Given an instance $(G, c)$ of SC($\alpha, \beta, \gamma$), we can derive an equivalent instance $(G, c')$ of SC($0, \beta - \alpha, \gamma - \alpha$) by setting $c'_i := c_i - \alpha\delta(i)$. (Here, $\delta$ refers to the degree in $G$.) So with respect to computational complexity of SC($\alpha, \beta, \gamma$) we may always assume that ($\alpha, \beta, \gamma$) is *normalized*, in the sense that $\alpha = 0 \leq \beta \leq \gamma$.

### 5.2.2 Complexity results

Our main result (cf. Kern and Paulusma [2001]) completely determines the computational complexity of the sports competition problem. In cases where SC($\alpha, \beta, \gamma$) turns out to be $\mathcal{NP}$-complete we prove this by reduction from 3-dimensional matching. This is a well-known $\mathcal{NP}$-complete problem (cf. Garey and Johnson [1979]).

3-DIMENSIONAL MATCHING (3DM)

*Instance*: three disjoint sets $X$, $Y$, $W$ with the same number of elements $q$ and a subset $R \subseteq X \times Y \times W$.

*Question*: Does $R$ contain a *(3-dimensional) matching*, i.e., is there a subset of triples $R' \subseteq R$ such that $R'$ covers each element of $X \cup Y \cup W$ exactly once?

(In Bernholt, Gülich, Hofmeister and Schmitt [1999] $\mathcal{NP}$-completeness is proved by a reduction from the $\mathcal{NP}$-complete problem 3-SATISFIABILITY, see Garey and Johnson [1979].)

**Theorem 5.1** *SC($\alpha$, $\beta$, $\gamma$) is polynomially solvable in each of the following three cases:*

*(i)* $\alpha = \beta$

*(ii)* $\beta = \gamma$

*(iii)* $\alpha + \gamma = 2\beta$.

*In all other cases the problem is $\mathcal{NP}$-complete.*

**Proof**: First recall that we may assume that $(\alpha, \beta, \gamma)$ is normalized, so $\alpha = 0$. (Note that normalization does not affect cases (i)-(iii).) **Case (i)** is then trivial. Indeed, an instance $(G, c)$ has a solution if and only if $c \geq 0$. (Leave all edges unoriented.)

In all other cases we have $\beta > 0$. By scaling we may assume that $\beta = 1$. (Divide $\beta$, $\gamma$ as well as $c$ by $\beta$.)

**Case (ii)** $\beta = \gamma = 1$ (corresponding with the exclusion of draws).

Consider an instance given by $G = (V, E)$ and $c \in \mathbb{R}^V$. Construct a directed bipartite graph with node sets $V$ and $E$ and arcs linking each $i \in V$ to all edges in $E$ incident with $i$ in $G$. Then add an additional source $s$ and sink $t$ as indicated in Figure 5.3. The arcs from $s$ to $V$ all get lower capacity 0 and upper capacity $\lfloor c_i \rfloor$ $(i \in V)$. The arcs from $V$ to $E$ get lower capacity 0 and upper capacity 1. The arcs from $E$ to $t$ get lower capacity and upper capacity 1. The resulting network has a feasible $s$-$t$ flow $x \in \mathbb{R}^{|V|+3|E|}$ if and only if our instance $(G, c)$ has a solution. Indeed, as all capacities are integral, a feasible flow may also be assumed to be integral (cf. Theorem 1.4). Given an integral feasible flow we can interpret an arc $(i, (i, j))$ from $V$ to $E$ that

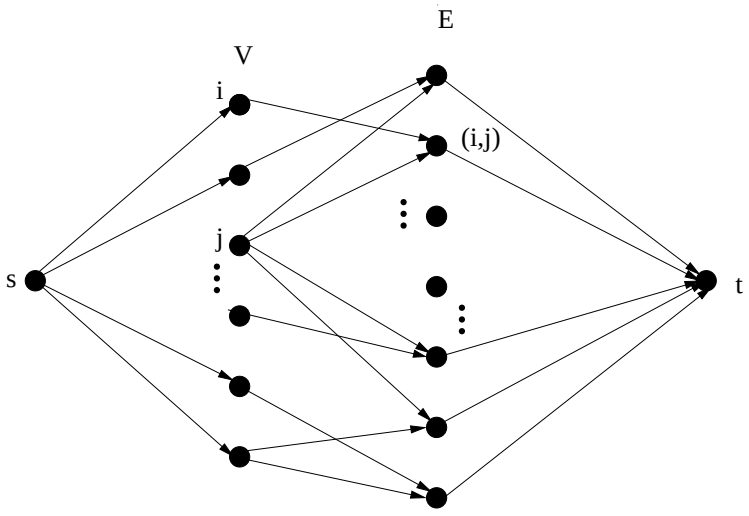


Figure 5.3

carries 1 unit of flow as $i$ winning the match $e = (i, j)$ and conversely (cf. also Cook, Cunningham, Pulleyblank and Schrijver [1998]).

**Case (iii)** $\beta = 1, \gamma = 2$ (ancient FIFA rule).

This can be solved similarly. In the network of Figure 5.3 we simply redefine the upper capacities of all arcs from $V$ to $E$ to be 2. The lower and upper capacities of arcs from $E$ to $t$ are also set to 2. Again, feasible integral flows are in 1-1 correspondence with solutions of our instance $(G, c)$. Each node $e \in E$ in our network has two incoming arcs that carry a total flow of 2 units, distributed as 2 : 0 or 1 : 1, corresponding to a win/loss match or a draw.

To prove the last part of the theorem we make a distinction between the case "$\beta = 1, \gamma > 2$" and "$\beta = 1, 1 < \gamma < 2$". Since we can guess an outcome of the remaining matches and compute the final scores, in both cases SC$(\alpha, \beta, \gamma)$ is a member of $\mathcal{NP}$.

**Case (iv)** $\beta = 1, \gamma > 2$.

We prove $\mathcal{NP}$-completeness by reduction from 3DM. Suppose $|X| = |Y| = |W| = q$ and $R \subseteq X \times Y \times W$ is given. We are to determine whether $R$ contains a matching $R' \subseteq R$. Assume without loss of generality that each element $z \in X \cup Y \cup W$ actually occurs in some triple $r \in R$. We write $z \in r$ to indicate that $z$ occurs in $r \in R$.

Given $R \subseteq X \times Y \times W$, we construct a graph $G = (V, E)$ as follows. We first make one copy of each element $z \in X \cup Y \cup W$ for each occurrence of $z$ in $R$, i.e., we define

$$\begin{array}{lcl} \bar{X} & := & \{(x, r) \mid x \in X, r \in R, x \in r\} \\ \bar{Y} & := & \{(y, r) \mid y \in Y, r \in R, y \in r\} \\ \bar{W} & := & \{(w, r) \mid w \in W, r \in R, w \in r\}. \end{array}$$

Construct a graph $G = (V, E)$ with node set $V = X \cup Y \cup W \cup \bar{X} \cup \bar{Y} \cup \bar{W} \cup R$ and edges as defined by the incidence relations in a straightforward way, i.e.,

$$\begin{array}{lcll} E & = & \{(x, (x, r)) \mid (x, r) \in \bar{X}\} & \\ & \cup & \{(y, (y, r)) \mid (y, r) \in \bar{Y}\} & \\ & \cup & \{(w, (w, r)) \mid (w, r) \in \bar{W}\} & \\ & \cup & \{(r, (x, r)) \mid (x, r) \in \bar{X}\} & \\ & \cup & \{(r, (y, r)) \mid (y, r) \in \bar{Y}\} & \\ & \cup & \{(r, (w, r)) \mid (w, r) \in \bar{W}\} & \text{(cf. Figure 5.4 below).} \end{array}$$

Next define node capacities $c \in \mathbb{R}^V$ as follows:

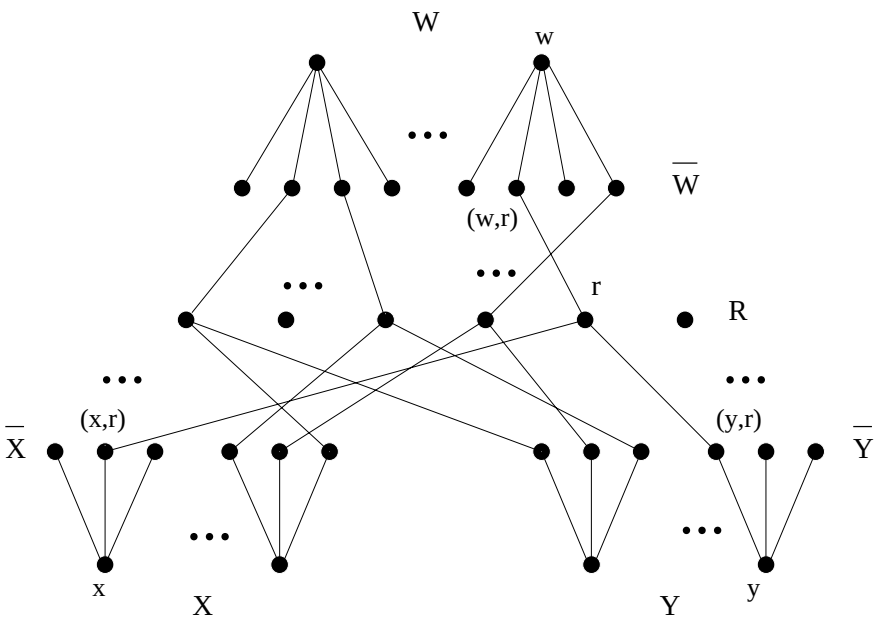


Figure 5.4

$$\begin{array}{lcll} c & \equiv & 1 & \text{on } X \cup Y \\ c & \equiv & 1+\gamma & \text{on } \bar{X} \cup \bar{Y} \\ c & \equiv & \max\{\gamma, 3\} & \text{on } R \\ c & \equiv & 1 & \text{on } \bar{W} \\ c & \equiv & \gamma(\delta-1)+1 & \text{on } W. \end{array}$$

(Again, $\delta$ refers to the degree function of $G$.)

We claim that this instance $(G, c)$ has a solution if and only if $R$ contains a matching.

"$\Leftarrow$" Suppose $R' \subseteq R$ is a matching. Define a corresponding partial orientation of $G$ as follows. For each $w \in W$ choose the unique $r' \in R'$ with $(w, r') \in \bar{W}$. We leave the edge $(w, (w, r'))$ unoriented and orient all other edges from $w$ to $\bar{W}$. This way the capacity constraints of $w$ are met. For each $r' = (x, y, w) \in R'$ we orient the edge $(r', (w, r'))$ from $r'$ towards $(w, r')$ and the edges $(r', (x, r'))$ and $(r', (y, r'))$ from $\bar{X}$ respectively $\bar{Y}$ towards $r'$. All edges incident with $r \in R \backslash R'$ remain unoriented. This way we ensure that the capacity constraints on $\bar{W}$ and $R$ are respected. Finally, orient all edges between $\bar{X}$ and $X$ from $\bar{X}$ towards $X$ except those that correspond to an element in $R'$ (these remain unoriented). This way the capacity constraints for $X$ and $\bar{X}$ are met. We orient edges between $\bar{Y}$ and $Y$ in the same way. This partial orientation gives a solution of the instance $(G, c)$.

"$\Rightarrow$" Conversely, suppose we are given a partial orientation of $G$ respecting the capacity constraints. The latter imply that for $x \in X$ we have $\delta^-(x) \geq \delta(x) - 1$ and $\delta^+(x) = 0$. We may assume without loss of generality that actually $\delta^-(x) = \delta(x) - 1$. (Otherwise, i.e., if $\delta^-(x) = \delta(x)$, pick an arbitrary edge incident with $x$ and make it unoriented. The modified orientation will still respect all capacity constraints.) A similar argument holds for elements $y \in Y$. Nodes in $\bar{X}$ have degree 2. In view of their capacity bound $1 + \gamma$, we may assume without loss of generality that each $(x, r) \in \bar{X}$ has $\delta^0 = 1$ and $\delta^+ = 1$. (Otherwise, again modify the solution without violating the capacity constraints.) As each $x \in X$ has $\delta^-(x) = \delta(x) - 1$ and $\delta^0(x) = 1$, we conclude that

• There are exactly $|X|$ arcs directed from $\bar{X}$ to $R$. Moreover, if $((x, r), r)$ is directed towards $r$ and $((x', r'), r')$ is directed towards $r'$, then $x \neq x'$.

The same holds for the directed arcs from $\bar{Y}$ to $R$.

Arguing similarly for nodes in $W$, we find that each $w \in W$ has without loss of generality $\delta^+(w) = \delta(w) - 1$ and $\delta^0(w) = 1$. (Otherwise modify the orientation such that $w$ actually uses its full capacity.) Nodes in $\bar{W}$ have degree 2 and capacity bound 1. Then we may without loss of generality assume that these nodes have $\delta^0 = 1$ and $\delta^- = 1$. The above implies that

• There are exactly $|W|$ arcs directed from $R$ towards $\bar{W}$. Moreover, if the edge $(r, (w, r))$ is directed from $r$ towards $(w, r)$ and $(r', (w', r'))$ is directed from $r'$ towards $(w', r')$, then $w \neq w'$.

Finally, the capacity constraints on $R$ imply that a node $r \in R$ can have $\delta^+ \geq 1$ only if $\delta^- \geq 2$. From this and the above observations, it is straightforward to check that

$$R' = \{r \in R \mid \delta^+(r) = 1\}$$

actually is a matching.

**Case (v)** $\beta = 1 < \gamma < 2$.

Again, we prove $\mathcal{NP}$-completeness by reduction from 3DM. In the graph $G$ of Figure 5.4 we replace the node capacities $c \in \mathbb{R}^V$ by node capacities $\tilde{c}$ as follows:

$$\begin{array}{lcll} \tilde{c} & \equiv & \gamma(\delta - 1) + 1 & \text{on } X \cup Y \\ \tilde{c} & \equiv & 1 & \text{on } \bar{X} \cup \bar{Y} \\ \tilde{c} & \equiv & \max\{2\gamma, 3\} & \text{on } R \\ \tilde{c} & \equiv & 1 + \gamma & \text{on } \bar{W} \\ \tilde{c} & \equiv & 1 & \text{on } W. \end{array}$$

Analogously to Case (iv) one can prove that the instance $(G, \tilde{c})$ has a solution if and only if $R$ contains a matching. □

Theorem 5.1 is related to the class of all possible sports competitions. Recall that this class contains both closed competitions and open competitions. Because a lot of sports competitions such as the FIFA soccer competitions are closed, we want to determine the computational complexity of our sports competition problem restricted to this kind of competition as well. Obviously for score allocation rules, for which the general sports competition can be solved in polynomial time, the problem for closed competitions can be solved efficiently. The following theorem shows that also for the other rules the computational complexity does not change. We call the number of times that each team plays against every other team the *order* $\rho$ of the competition.

**Theorem 5.2** *Consider a class of closed competitions with fixed rule $(\alpha, \beta, \gamma)$ and fixed order $\rho$. Then the problem of determining whether at a given stage a certain team can win the competition is $\mathcal{NP}$-complete if*

$$\alpha < \beta < \gamma \ \text{ and } \ \alpha + \gamma \neq 2\beta.$$

**Proof**: For the rule $(\alpha, \beta, \gamma)$ we again assume $\alpha = 0$ and $\beta = 1$. If $\gamma > 2$ then we only have to show that for the instance $(G, c)$, as defined in Case (iv) of the proof of Theorem 5.1, there exists a closed competition of order $\rho$ with a state $(\bar{N}, s, M)$ such that $(G, c)$ is obtained from this state. If $1 < \gamma < 2$ we have to find an appropriate competition for instance $(G, \tilde{c})$ of Case (v) in the proof of Theorem 5.1. Because such a closed competition can be constructed in the same way we only prove the case $\gamma > 2$.

Hence let $\gamma > 2$ and consider (first) the case $\rho = 1$.

In Case (iv) of the proof of Theorem 5.1 we have a graph $G = (V, E)$ with node set $V = X \cup Y \cup W \cup \bar{X} \cup \bar{Y} \cup \bar{W} \cup R$ and edges as defined in Figure 5.4. Furthermore, the node capacities $c \in \mathbb{R}^V$ are given by

$$\begin{array}{lcll} c & \equiv & 1 & \text{on } X \cup Y \\ c & \equiv & 1+\gamma & \text{on } \bar{X} \cup \bar{Y} \\ c & \equiv & \max\{\gamma, 3\} & \text{on } R \\ c & \equiv & 1 & \text{on } \bar{W} \\ c & \equiv & \gamma(\delta - 1) + 1 & \text{on } W. \end{array}$$

We construct a competition that contains (besides team $t_0$) the teams corresponding with $V$ and, in addition, the following teams:

- for all $x \in X$ a set $H(x)$ of $\delta(x) - 1$ teams;
- for all $y \in Y$ a set $H(y)$ of $\delta(y) - 1$ teams;
- for all $\bar{w} \in \bar{W}$ a set $H(\bar{w})$ of one team;
- for all $r \in R$ a set $H(r)$ of two teams.

Let $H_1$ denote the collection of all teams as defined above ,i.e.,

$$H_1 = \bigcup_{x \in X} H(x) \cup \bigcup_{y \in Y} H(y) \cup \bigcup_{\bar{w} \in \bar{W}} H(\bar{w}) \cup \bigcup_{r \in R} H(r).$$

Besides $H_1$, we add a set of teams $H_2$ of size $|H_2|$ sufficiently large (as explained later). Then the set of teams in this competition is

$$\bar{N} = V \cup H_1 \cup H_2 \cup \{t_0\}.$$

The total set of matches between pairs of teams in $\bar{N}$ has size $\frac{1}{2}|\bar{N}|(|\bar{N}|-1)$. We want the set of remaining matches $M$ to correspond exactly with the set of edges $E$ and a current score vector $s$ in such a way that every team $i \in V$ may collect at most $c_i$ additional score points. For this purpose we define the following outcomes of matches not in $E$.

. Team $t_0$ wins all its matches against teams in $H_2$, its matches against teams in $V$ all end in a draw, and $t_0$ loses all its matches against the teams in $H_1$.

. Matches between teams in $V$ that do not correspond with an edge in $E$ end in a draw.

. Every team $i \in X \cup Y \cup \bar{W}$ wins all its matches against teams in $H_2$, its matches against teams in $H(i)$ all end in a draw, and $i$ loses all matches against teams in $H_1 \backslash H(i)$.

. Every team $i \in W$ wins $|H_2| - \delta(i) + 1$ matches against teams in $H_2$, all its other matches against teams in $H_2$ end in a draw, and $i$ loses all matches against teams in $H_1$.

. Every team $i \in \bar{X} \cup \bar{Y}$ wins $|H_2| - 1$ matches against teams in $H_2$, its remaining match in $H_2$ ends in a draw, and $i$ loses all matches against teams in $H_1$.

. If $\gamma \leq 3$, then every team $i \in R$ wins all matches against teams in $H_2$ and loses its matches against $H(i)$. If $\gamma > 3$, then every team $i \in R$ wins $|H_2| - 1$ matches against teams in $H_2$ and its remaining match in $H_2$ plus the matches against the teams in $H(i)$ end in a draw. In both cases $i$ loses its matches against teams in $H_1 \backslash H(i)$.

. Every team $i \in H_1$ loses all matches against teams in $H_2$. Matches between teams in $H_1$ all end in a draw.

. Matches between teams in $H_2$ end in a draw.

This way $t_0$ has no remaining matches and $\tilde{s}_0 = s_0 = |V| + \gamma|H_2|$. Furthermore we have made sure that $c_i = \tilde{s}_0 - s_i$ holds for all teams $t_i \in V$.

Because all teams in $H_1 \cup H_2$ have no remaining matches, they will correspond with isolated nodes if they are added to $G$. So we have a set of remaining matches $M$ in 1-1 correspondence with edges in $E$. However, we still have to check that teams in $H_1 \cup H_2$ do not have a final score greater than $\tilde{s}_0$. For all $i \in H_1$ we deduce that its score

$$s_i \leq |H_1| + \gamma + \gamma + (|V| - 1)\gamma = |H_1| + \gamma + \gamma|V|,$$

and for all $i \in H_2$ we have

$$s_i \leq |H_2| + \gamma|H_1| + |V|.$$

It is easy to see that both upper bounds will be smaller than $|V| + \gamma|H_2| = \tilde{s}_0$ provided that $H_2$ is sufficiently large. Hence we have constructed a competition with a state $(\tilde{N}, s, M)$ such that $(G, c)$ is obtained from this state (omitting the isolated nodes).

Above we have assumed that each team plays exactly one time against all other teams. For a class of competitions with order $\rho > 1$ we let $\rho - 1$ matches between every pair of teams end in a draw. Then we use the same construction as in the case $\rho = 1$. □

Theorem 5.2 implies that for soccer competitions with the current FIFA rule $(0, 1, 3)$ and order for instance 2 or 4 it has indeed become $\mathcal{NP}$-complete to decide whether at a given stage a certain soccer team wins the competition.

### 5.2.3 Related problems

Examining the proof of Theorem 5.1 we see that the network model we used for solving cases (ii) and (iii) of our main theorem does not apply for cases in which the rule $(\alpha, \beta, \gamma)$ has the property that $\alpha < \beta < \gamma$ and $\alpha + \gamma \neq 2\beta$. Take for instance the rule $(\alpha, \beta, \gamma) = (0, 1, 3)$. If we increase the upper capacities to 3 on all arcs from $V$ to $E$ and from $E$ to $t$ in the network of Figure 5.3, then a feasible flow does no longer necessarily represent a solution of our instance. (A total flow of 2 entering a node $e = (i, j) \in E$ distributed as $2 : 0$ on the two entering arcs does not correspond to a win/loss or a draw.) If we "repair" this by introducing a "capacity gap" $]1, 3[$ on all arcs from $V$ to $E$ we get a flow problem with capacity gaps, which again nicely describes our sports competition problem. So as a consequence of our result, the following class of problems is also $\mathcal{NP}$-complete.

**Definition 5.3** FLOWS WITH CAPACITY GAPS (FCG)

*Instance*: A digraph $D = (V, A)$ with source $s$ and sink $t$ and for each arc $a \in A$ two disjoint capacity intervals $I_1(a) = [c_1(a), c_2(a)]$ and $I_2(a) = [c_3(a), c_4(a)]$ $(c_i(a) \in \mathbb{Z},\ i = 1, \ldots, 4)$.

*Question*: Does a (integral) $s$-$t$ flow $x \in \mathbb{Z}^A$ exist with $x(a) \in I_1(a) \cup I_2(a)$ $(a \in A)$? □

**Corollary 5.1** *FCG is $\mathcal{NP}$-complete.* □

As to sports competitions, we would like to remark that also other questions can be treated in the same way. For example "Is there a chance that team $t_0$ ends up with the lowest final score?" turns out to be of exactly the same complexity as SC: Assume that $t_0$ has a current total score $s_0$ and loses all remaining matches. This results in a current total score $s_i$ for all other teams $t_i \neq t_0$. The question is now whether the teams $t_i \neq t_0$ can finish the remaining matches in such a way that each $t_i$ collects *at least* $c_i := s_0 - s_i$ additional score points. Again we model this by a multigraph $G = (V, E)$ whose nodes correspond to teams $t_i \neq t_0$ and edges are in 1-1 correspondence with remaining matches. Each node $i \in V$ has a (lower) capacity $c_i \in \mathbb{R}$. Our "reverse" sports competition problem (RSC) can now be formulated as follows.

**Definition 5.4** REVERSE SPORTS COMPETITION (RSC($\alpha, \beta, \gamma$))

*Instance*: A multigraph $G = (V, E)$ and node capacities $c \in \mathbb{R}^V$.

*Question*: Can $G$ be partially oriented such that for each node $i \in V$:

$$\alpha\delta^-(i) + \beta\delta^0(i) + \gamma\delta^+(i) \geq c_i \ ? \tag{5.2}$$

□

It is easy to see that for $i \in V$, (5.2) is equivalent to

$$(\gamma - \beta)\delta^0(i) + (\gamma - \alpha)\delta^-(i) \leq \gamma\delta(i) - c_i.$$

Hence an instance $(G, c)$ of RSC($\alpha, \beta, \gamma$) corresponds to an instance $(G, \gamma\delta - c)$ of SC($0, \gamma - \beta, \gamma - \alpha$), and the corollary below immediately follows from Theorem 5.1.

**Corollary 5.2** *RSC($\alpha, \beta, \gamma$) is polynomially solvable in each of the following three cases:*

*(i)* $\alpha = \beta$

*(ii)* $\beta = \gamma$

*(iii)* $\alpha + \gamma = 2\beta$.

*In all other cases the problem is $\mathcal{NP}$-complete.* □

Questions such as "Is there a chance that $t_0$ ends up being one of the three teams that have the three lowest final scores?" can also be treated in a similar way. Again, assume that $t_0$ has a current total score $s_0$ and loses all remaining matches. Choose two teams $t_i, t_j \neq t_0$, and let $t_i$ and $t_j$ lose their remaining matches against teams $t_k$ $(k \neq 0, i, j)$. (Choose, if necessary, an arbitrary outcome for the matches between $t_i$ and $t_j$.) These outcomes result in final total scores $\tilde{s}_0 = s_0$, $\tilde{s}_i$ and $\tilde{s}_j$, and current total scores $s_k$ for all other teams $t_k$ $(k \neq 0, i, j)$. If it is possible that the teams $t_k$ $(k \neq 0, i, j)$ can finish the remaining matches in such a way that each $t_k$ collects at least $c_k := \tilde{s}_0 - s_k$ additional score points, then $t_0$ can indeed end up being one of the three lowest teams. If this is not possible for any pair $t_i, t_j$, then $t_0$ can never end up being one of the three lowest teams. So one has to solve at most $\frac{1}{2}|N|(|N|-1)$ problem instances in RSC$(\alpha, \beta, \gamma)$. Hence also this question is of the same complexity as SC.

## 5.3 Competition games

Consider a competition game $(N, v)$ obtained from a sports competition with state $(\bar{N}, s, M)$ and rule $(\alpha, \beta, \gamma)$. Clearly computing the value $v(N)$ comes down to solve the corresponding instance $(G, c)$ of SC$(\alpha, \beta, \gamma)$. Below we give an algorithm for computing the other values $v(S)$ $(S \neq N)$. Denote the set of remaining matches of team $t_i$ by $M_i$. For $S \subseteq N$ let $G^S = (V, E)$ be the multigraph whose nodes correspond with the teams in $S$ and whose edges are in 1-1 correspondence with the remaining matches between teams in $S$. In step (1) of the algorithm we assume that, after bribing, $t_0$ will win its remaining matches against teams in $S$. However, in our model $t_0$ will only bribe a coalition $S$ if it has an absolute guarantee on winning the competition. Because there is a probability that $t_0$ loses its matches in $M$ against teams outside $S$, we reward these kind of matches only with $\alpha$ points resulting in a lower bound $\tilde{s}_0$ on its final total score. Obviously there is no guarantee on winning, if it is possible for a team outside $S$ to achieve a higher final score than $\tilde{s}_0$. If this situation cannot occur, we proceed with step (4). Recall that we assume that both teams have to be involved if the end result of a match

must be set to a draw and that in case of a win/loss match only the losing team has to be bribed. Under this assumption we let every team in $S$ lose its remaining matches against teams outside $S$. Then it remains to check if the corresponding instance of SC($\alpha$, $\beta$, $\gamma$) has a solution or not.

---

**Algorithm COMP**:

(**1**) Compute the final total score $\tilde{s}_0$ for $t_0$

$$\begin{aligned}\tilde{s}_0 \;:=\; s_0 \;&+\; \alpha|\{m \in M_0 \mid m = (t_0, t_i) \text{ with } t_i \notin S\}| \\ &+\; \gamma|\{m \in M_0 \mid m = (t_0, t_i) \text{ with } t_i \in S\}|.\end{aligned}$$

(**2**) Compute the highest final score $\tilde{s}$ that a team $t_i \notin S$ might achieve

$$\tilde{s} := \max\{s_i + \gamma|M_i| \mid t_i \notin S\}.$$

(**3**) IF $\tilde{s}_0 < \tilde{s}$ THEN output $v(S) = 0$ and STOP.

(**4**) Adjust the current score $s_i$ for each team $t_i \in S$

$$s_i := s_i + \alpha|\{m \in M_i \mid m = (t_i, t_j) \text{ with } t_j \in \bar{N}\backslash S\}| \quad (t_i \in S).$$

(**5**) Compute the amount $c_i$ of additional score points that each team $t_i \in S$ may collect by

$$c_i := \tilde{s}_0 - s_i \quad (t_i \in S).$$

(**6**) Solve the instance $(G^S, c)$ of SC($\alpha$, $\beta$, $\gamma$).

(**7**) IF the answer for $(G^S, c)$ is "yes" THEN output $v(S) := v^*$ OTHERWISE output $v(S) := 0$. STOP.

---

The computational complexity of the algorithm depends on the complexity of solving an instance of SC($\alpha$, $\beta$, $\gamma$). COMP is a polynomial time algorithm if and only if SC($\alpha$, $\beta$, $\gamma$) is polynomially solvable.

Let $N^*$ denote the set of *indispensable teams*, which contains all teams that have to be included in any coalition $S$ that is bribed by $t_0$. Because obviously

$v(S) \leq v(T)$ for all $S \subseteq T \subseteq N$, we have

$$N^* := \{i \in N \mid v(N\backslash i) = 0\}.$$

Note that $N^*$ can be computed in polynomial time if the corresponding sports competition problem SC($\alpha$, $\beta$, $\gamma$) is polynomially solvable.

The following result gives an easy characterization of the core of a competition game.

**Theorem 5.3** *Let $(N, v)$ be a competition game with $v(N) = v^*$. If $N^*$ is empty, then core$(N, v)$ is empty. Otherwise*

$$core(N, v) = \{x \in I(N, v) \mid x(N^*) = v^*\}.$$

**Proof**: First suppose $N^* = \emptyset$. Then $v(N\backslash i) = v^*$ for all $i \in N$. So if $x \in \mathbb{R}^N$ were in the core, then

$$|N|v^* \leq \sum_{i \in N} x(N\backslash i) = (|N| - 1)x(N),$$

implying $x(N) > v^*$, a contradiction.

Suppose $N^* \neq \emptyset$. Then $v(S) = 0$ if $N^*\backslash S$ is nonempty. Hence any positive allocation $x \in \mathbb{R}^N$ with $x(N^*) = v^*$ is a core vector.

Now suppose $x \in \text{core}(N, v)$. Then $x \geq 0$ and $x(N) = v^*$. If $N^* = N$ we are finished. Suppose $i \notin N^*$. Then we deduce from $x(N\backslash i) \geq v(N\backslash i) = v^*$, $x \geq 0$ and $x(N) = v^*$ that $x_i = 0$. □

In Example 5.1 $v(N\backslash i) = v^*$ for all $i \in N$. Hence $N^*$ and core$(N, v)$ are empty. The example below shows a competition game with a nonempty core.

**Example 5.3** Consider a competition with rule $(0, 1, 2)$ and state $(\bar{N}, s, M)$ given by the tables below.

| teams | scores |
|---|---|
| $t_0$ | 10 |
| $t_1$ | 13 |
| $t_2$ | 6 |
| $t_3$ | 5 |
| $t_4$ | 0 |

| remaining matches |
|---|
| $t_0 - t_2$ |
| $t_0 - t_3$ |
| $t_1 - t_4$ |

The competition game $(N, v)$ is determined by $N = \{t_1, t_2, t_3, t_4\}$ and $v : 2^N \to \mathbb{R}_+$ given by

$$v(S) = \begin{cases} v^* & \text{if } S = \{t_1, t_2, t_3\} \text{ or } S = N \\ 0 & \text{otherwise.} \end{cases}$$

Then $N^* = \{t_1, t_2, t_3\}$ and

$$\text{core}(N, v) = \{x \in \mathbb{R}^N \mid x_1 + x_2 + x_3 = v^*,\ x_4 = 0 \text{ and } x \geq 0\}.$$

□

In the next theorem we prove that the Shapley value of a competition game $(N, v)$ can easily be computed if $t_0$ only has to bribe the indispensable teams.

**Theorem 5.4** *Let $(N, v)$ be a competition game with $N^* \neq \emptyset$. If $v(N^*) = v^*$, then the Shapley value $\phi$ is given by*

$$\phi_i(N, v) = \begin{cases} \dfrac{v^*}{|N^*|} & \textit{if } i \in N^* \\ 0 & \textit{otherwise.} \end{cases}$$

**Proof**: (i) Suppose $i, j \in N^*$. Then

$$v(S \cup i) = 0 = v(S \cup j) \text{ for all } S \subseteq N \backslash \{i, j\}.$$

By Theorem 2.4 we know that the Shapley value satisfies *Sym*. Now define a permutation $\pi : N \to N$ as follows:

$$\pi(k) = \begin{cases} j & \text{if } k = i \\ i & \text{if } k = j \\ k & \text{otherwise.} \end{cases}$$

It follows that $v^\pi(S) = v(S)$ for all $S \subseteq N$. Then, according to *Sym*,

$$\phi_i(N, v) = \phi_j(N, v) \text{ for all } i, j \in N^*. \tag{5.3}$$

Also, by Theorem 2.4 the Shapley value satisfies *Dum*. Suppose $k \notin N^*$. First note that $v(N^*) = v^*$ and $v(T) \leq v(S)$ for all $T \subseteq S$ imply that $v(S) = v^*$ if $N^* \subseteq S$. Then $v(S) - v(S\backslash k) = 0 = v(k)$ for all $S \subseteq N$. So $k$ is a dummy in $(N, v)$. By property *Dum* we have

$$\phi_k(N, v) = 0 \text{ for all } k \in N \backslash N^*. \tag{5.4}$$

(5.3) and (5.4) imply that

$$\phi_i(N, v) \quad = \quad \begin{cases} \dfrac{v^*}{|N^*|} & \text{if } i \in N^* \\ 0 & \text{otherwise.} \end{cases}$$

□

Note that the Shapley value as defined above is a core allocation. However, the following example makes clear that in case $v(N^*) = 0$ the Shapley value is not necessarily of this form.

**Example 5.4** We consider a competition with rule $(0, 1, 2)$ and state $(\bar{N}, s, M)$ given by the tables below.

| teams | scores |
|---|---|
| $t_0$ | 8 |
| $t_1$ | 7 |
| $t_2$ | 7 |
| $t_3$ | 5 |
| $t_4$ | 1 |

| remaining matches |
|---|
| $t_0 - t_3$ |
| $t_0 - t_4$ |
| $t_1 - t_3$ |
| $t_3 - t_1$ |
| $t_2 - t_3$ |
| $t_3 - t_2$ |

Then the competition game $(N, v)$ is defined by set of teams $N = \{t_1, t_2, t_3, t_4\}$ and characteristic function $v$ given by

$$v(S) \quad = \quad \begin{cases} v^* & \text{if } t_3 \text{ and } t_4 \text{ in } S \text{ or } S = \{t_1, t_2, t_3\} \\ 0 & \text{otherwise.} \end{cases}$$

Hence $N^* = \{t_3\}$. Since $v(t_3) = 0$, Theorem 5.4 cannot be applied. Computing the Shapley value yields $\phi_1 = \phi_2 = \frac{1}{12}v^*$, $\phi_3 = \frac{7}{12}v^*$ and $\phi_4 = \frac{3}{12}v^*$. Note that the Shapley value of this competition game is not a core allocation. □

Also with respect to the problems mentioned in the previous section (e.g., RSC) we can define similar competition games, where $t_0$ tries to bribe some other teams. Furthermore, our model can be extended by letting the value $v(S)$ depend on the (size of) coalition $S$ or by including some other teams that also want to win the competition by means of bribing.

# Bibliography


AARTS, H. [1994], *Minimum Cost Spanning Tree Games and Set Games*, Ph.D. thesis, University of Twente, Enschede, the Netherlands.

BERNHOLT, T., A. GÜLICH, T. HOFMEISTER, AND N. SCHMITT [1999], Football elimination is hard to decide under the 3-point-rule, in: M. Kutylowski, L. Pacholski, and T. Wierzbicki (eds.), *Mathematical Foundations of Computer Science 1999*, Lecture Notes in Computer Science 1672, 410–418.

BILBAO, J.M. [2000], *Cooperative Games on Combinatorial Structures*, Kluwer Academic, Norwell, Massachusetts.

BIRD, C.G. [1976], On cost allocation for a spanning tree. a game theoretic approach, *Networks* **6**, 335–350.

BONDY, J.A., AND U.S.R. MURTY [1976], *Graph Theory with Applications*, Macmillan, London and Elsevier, New York.

COOK, S.A. [1971], The complexity of theorem-proving procedures, *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ACM, New York, 151–158.

COOK, W.J., W.H. CUNNINGHAM, W.R. PULLEYBLANK, AND A. SCHRIJVER [1998], *Combinatorial Optimization*, John Wiley & Sons, New York.

DAVIS, M., AND M. MASCHLER [1965], The kernel of a cooperative game, *Naval Research Logistics Quarterly* **12**, 223–259.

DENG, X., T. IBARAKI, AND H. NAGAMOCHI [1999], Algorithmic aspects of the core of combinatorial optimization games, *Mathematics of Operations Research* **24**, 751–766.

DENG, X., AND C.H. PAPADIMITRIOU [1994], On the complexity of cooperative game solution concepts, *Mathematics of Operations Research* **19**, 257–266.

DRIESSEN, T.S.H. [1991], A survey of consistency properties in cooperative

game theory, *SIAM Review* **33**, 43–59.

DRIESSEN, T.S.H., AND D. PAULUSMA [2001], Two extensions of the Shapley value for cooperative games, *Mathematical Methods of Operations Research* **53**, 35–49.

EDMONDS, J. [1965a], Maximum matching and a polyhedron with $0,1$-vertices, *Journal of Research of the National Bureau of Standards Section B* **69B**, 125–130.

EDMONDS, J. [1965b], Paths, trees, and flowers, *Canadian Journal of Mathematics* **17**, 449–467.

EVANS, R.A. [1996], Value, consistency, and random coalition formation, *Games and Economic Behavior* **12**, 68–80.

FAIGLE, U., AND W. KERN [1992], The Shapley value for cooperative games under precedence constraints, *International Journal of Game Theory* **21**, 249–266.

FAIGLE, U., AND W. KERN [1993], On some approximately balanced combinatorial cooperative games, *ZOR - Methods and Models of Operations Research* **38**, 141–152.

FAIGLE, U., W. KERN, S.P. FEKETE, AND W. HOCHSTÄTTLER [1997], On the complexity of testing membership in the core of min-cost spanning tree games, *International Journal of Game Theory* **26**, 361–366.

FAIGLE, U., W. KERN, S.P. FEKETE, AND W. HOCHSTÄTTLER [1998], The nucleon of cooperative games and an algorithm for matching games, *Mathematical Programming* **83**, 195–211.

FAIGLE, U., W. KERN, AND J. KUIPERS [1998a], Computing the nucleolus of min-cost spanning tree games is $NP$-hard, *International Journal of Game Theory* **27**, 443–450.

FAIGLE, U., W. KERN, AND J. KUIPERS [1998b], *An efficient algorithm for nucleolus and prekernel computation in some classes of TU-games*, Memorandum 1464, University of Twente, Enschede, the Netherlands.

FAIGLE, U., W. KERN, AND D. PAULUSMA [2000], Note on the computational complexity of least core concepts for min-cost spanning tree games, *Mathematical Methods of Operations Research* **52**, 23–38.

GALLAI, T. [1963], Kritische Graphen II, *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* **8**, 373–395.

GALLAI, T. [1964], Maximale Systeme unabhängiger Kanten, *A Magyar Tudományos Akadémia Matematikai Kutató Intézetének Közleményei* **9**, 401–413.

GAREY, M.R., AND D.S. JOHNSON [1979], *Computers and Intractability:*

*A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, California.

GILLIES, D.B. [1959], Solutions to general non-zero-sum games, in: A.W. Tucker and R.D. Luce (eds.), *Contributions to the Theory of Games IV*, Annals of Mathematics Studies 40, Princeton University Press, Princeton, New Jersey, 47–85.

GRANOT, D., AND F. GRANOT [1992], On some network flow games, *Mathematics of Operations Research* **17**, 792–841.

GRANOT, D., F. GRANOT, AND W.R. ZHU [1998], Characterization sets for the nucleolus, *International Journal of Game Theory* **27**, 359–374.

GRANOT, D., AND G. HUBERMAN [1981], Minimum cost spanning tree games, *Mathematical Programming* **21**, 1–18.

GRANOT, D., AND G. HUBERMAN [1984], On the core and nucleolus of minimum cost spanning tree games, *Mathematical Programming* **29**, 323–347.

GRANOT, D., M. MASCHLER, G. OWEN, AND W.R. ZHU [1996], The kernel/nucleolus of a standard tree game, *International Journal of Game Theory* **25**, 219–244.

GRÖTSCHEL, M., L. LOVÁSZ, AND A. SCHRIJVER [1981], The ellipsoid method and its consequences in combinatorial optimization, *Combinatorica* **1**, 169–197, [corrigendum: Combinatorica 4 (1984), 291-295].

GRÖTSCHEL, M., L. LOVÁSZ, AND A. SCHRIJVER [1993], *Geometric Algorithms and Combinatorial Optimization* (Second ed.), Springer-Verlag, Berlin.

HOFFMAN, A.J., AND J.B. KRUSKAL [1956], Integral boundary points of convex polyhedra, in: H.W. Kuhn and A.W. Tucker (eds.), *Linear Inequalities and Related Systems*, Annals of Mathematics Studies 38, Princeton University Press, Princeton, New Jersey, 223–246.

KERN, W., AND D. PAULUSMA [2000], *Matching games: the least core and the nucleolus*, Memorandum 1541, University of Twente, Enschede, the Netherlands.

KERN, W., AND D. PAULUSMA [2001], The new FIFA rules are hard: complexity aspects of sports competitions, *Discrete Applied Mathematics* **108**, 317–323.

KHACHIYAN, L.G. [1979], A polynomial algorithm in linear programming, *Soviet Mathematics Doklady* **20**, 191–194.

KÖNIG, D. [1931], Graphok és matrixok, *Matematikai és Fizikai Lapok* **38**, 116–119, (Hungarian).

KRUSKAL, J.B. [1956], On the shortest spanning subtree of a graph and the traveling salesman problem, *Proceedings of the American Mathematical Society* **7**, 48–50.

KUIPERS, J. [1996], *A polynomial time algorithm for computing the nucleolus of convex games*, Report M 96-12, Maastricht University, Maastricht, the Netherlands.

LOVÁSZ, L., AND M.D. PLUMMER [1986], *Matching Theory*, North-Holland Mathematics Studies 121, North-Holland, Amsterdam.

MASCHLER, M., B. PELEG, AND L.S. SHAPLEY [1979], Geometric properties of the kernel, nucleolus, and related solution concepts, *Mathematics of Operations Research* **4**, 303–338.

MEGIDDO, N. [1978], Computational complexity of the game theory approach to cost allocation for a tree, *Mathematics of Operations Research* **3**, 189–196.

MICALI, S., AND V.V. VAZIRANI [1980], An $\mathcal{O}(V^{\frac{1}{2}}E)$ algorithm for finding maximum matching in general graphs, *21st Annual Symposium on Foundations of Computer Science*, IEEE, New York, 17–27.

NEMHAUSER, G.L., AND L.A. WOLSEY [1999], *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, Reprint of the 1988 original.

NEUMANN, J. VON [1928], Zur Theorie der Gesellschaftsspielle, *Mathematische Annalen* **100**, 295–320.

NEUMANN, J. VON, AND O. MORGENSTERN [1944], *Theory of Games and Economic Behavior*, Princeton University Press, Princeton, New Jersey.

O'NEILL, B. [1982], A problem of rights arbitration from the Talmud, *Mathematical Social Sciences* **2**, 345–371.

OWEN, G. [1995], *Game Theory* (Third ed.), Academic Press, San Diego, California.

PAPADIMITRIOU, C.H., AND K. STEIGLITZ [1982], *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, New Jersey.

POTTERS, J.A.M, J.H. REIJNIERSE, AND M. ANSING [1996], Computing the nucleolus by solving a prolonged simplex algorithm, *Mathematics of Operations Research* **21**, 757–768.

PRIM, R.C. [1957], Shortest connection networks and some generalizations, *Bell Systems Technical Journal* **36**, 1389–1401.

QUINT, T. [1991], Characterization of cores of assignment games, *International Journal of Game Theory* **19**, 413–420.

RANSMEIER, J.S. [1942], *The Tennessee Valley Authority: A Case Study in the Economics of Multiple Purpose Stream Planning*, Vanderbilt University Press, Nashville, Tennessee.

SCHMEIDLER, D. [1969], The nucleolus of a characteristic function game, *SIAM Journal on Applied Mathematics* **17**, 1163–1170.

SCHRIJVER, A. [1986], *Theory of Linear and Integer Programming*, John Wiley & Sons, Chichester.

SHAPLEY, L.S. [1953], A value for $n$-person games, in: H.W. Kuhn and A.W. Tucker (eds.), *Contributions to the Theory of Games II*, Annals of Mathematics Study 28, Princeton University Press, Princeton, New Jersey, 307–317.

SHAPLEY, L.S., AND M. SHUBIK [1966], Quasi-cores in a monetary economy with nonconvex preferences, *Econometrica* **34**, 805–827.

SHAPLEY, L.S., AND M. SHUBIK [1972], The assignment game I: the core, *International Journal of Game Theory* **1**, 111–130.

SHUBIK, M. [1982], *Game Theory in the Social Sciences: Concepts and Solutions*, MIT Press, Cambridge, Massachusetts.

SOLYMOSI, T., AND T.E.S. RAGHAVAN [1994], An algorithm for finding the nucleolus of assignment games, *International Journal of Game Theory* **23**, 119–143.

SPRUMONT, Y. [1990], Population monotonic allocation schemes for cooperative games with transferable utility, *Games and Economic Behavior* **2**, 378–394.

TIJS, S.H., AND T.S.H. DRIESSEN [1986], Extensions of solution concepts by means of multiplicative $\epsilon$-tax games, *Mathematical Social Sciences* **12**, 9–20.

WALLMEIER, E. [1983], *Der f-Nukleolus und ein Dynamisches Verhandlungsmodell als Lösungskonzepte für kooperative n-Personenspiele*, Ph.D. thesis, Skripten zur Mathematischen Statistik 5, Westfälische Wilhelms-Universität Münster, Institut für Mathematische Statistik, Münster.

YOUNG, H.P., N. OKADA, AND T. HASHIMOTO [1982], Cost allocation in water resources development, *Water Resources Research* **18**, 463–475.

# Notations

The numbers refer to the page of first occurrence.

### General

A real number $a$ is called positive, if $a \geq 0$.

| | | |
|---|---|---|
| $\mathbb{R}$ | set of real numbers | 2 |
| $\mathbb{R}_+$ | set of positive real numbers | 2 |
| $\mathbb{Q}$ | set of rational numbers | 4 |
| $\mathbb{Z}$ | set of integers | 115 |
| $\mathbb{N}$ | set of positive integers | 4 |
| $\mathbb{R}^n$ | set of $n$-dimensional vectors with real components | 7 |
| $\mathbb{R}^n_+$ | set of $n$-dim. vectors with positive real components | 75 |
| $\mathbb{Q}^n$ | set of $n$-dim. vectors with rational components | 4 |
| $\mathbb{Z}^n$ | set of $n$-dim. vectors with integer components | 115 |
| $\|N\|$ | cardinality of $N$ | 11 |
| $2^N$ | power set of $N$ | 2 |
| $\mathbb{R}^N$ | set of all functions from $N$ to $\mathbb{R}$ | 2 |
| $S \subset T$ | if $S$ is a subset of $T$ and $S \neq T$ | 22 |
| $S \subseteq T$ | if $S$ is a subset of $T$ | 2 |
| $S \backslash T$ | $\{i \in S \mid i \notin T\}$ | 3 |
| $S \triangle T$ | $S \backslash T \cup T \backslash S$ | 66 |
| $S \cup i$ | $S \cup \{i\}$ | 31 |
| $S \backslash i$ | $S \backslash \{i\}$ | 16 |
| $\pi$ | permutation of $\{1, \ldots, n\}$ | 16 |
| $f(n) = \mathcal{O}(g(n))$ | $f(n) \leq cg(n)$ for $n$ sufficiently large | 4 |
| $t(\xi)$ | running time | 4 |

| | | |
|---|---|---|
| $x$ | vector in $\mathbb{R}^n$ or function in $\mathbb{R}^N$ denoted as vector | 2 |
| $x \geq 0$ | if $x_i \geq 0$ for $i = 1, \ldots, n$ | 34 |
| $x^T$ | transpose of $x$ | 8 |
| $< x >$ | size of $x$ if $x \in \mathbb{Q}^n$ | 4 |
| $x^\pi$ | vector given by $x^\pi_{\pi(i)} = x_i$ for $i = 1 \ldots, n$ | 17 |
| $X^\pi$ | $\{z \in \mathbb{R}^N \mid z = x^\pi \text{ for some } x \in X\}$ | 17 |
| $x \leq y$ | if $x_i \leq y_i$ for $i = 1, \ldots, n$ | 7 |
| $x \preceq y$ | if $x$ lexicographically smaller than $y$ | 20 |
| $\lambda X$ | $\{\lambda x \mid x \in X\}$ | 16 |
| $X + Y$ | $\{x + y \mid x \in X, y \in Y\}$ | 16 |

**Game theory**

| | | |
|---|---|---|
| $(N, v)$ | cooperative game (profit game) | 2 |
| $N$ | player set | 2 |
| $v$ | characteristic function (profit function) | 2 |
| $< N, v >$ | size of $(N, v)$ | 18 |
| $(N, c)$ | cost game | 2 |
| $c$ | cost function | 2 |
| $\lambda v + a$ | $(\lambda v + a)(S) = \lambda v(S) + a(S)$ for all $S \subseteq N$ | 17 |
| $v + w$ | $(v + w)(S) = v(S) + w(S)$ for all $S \subseteq N$ | 17 |
| $v^\pi$ | $v^\pi(\pi(S)) = v(S)$ for all $S \subseteq N$ | 17 |
| $x(S)$ | $\sum_{i \in S} x_i$ | 2 |
| $\Phi$ | solution concept | 15 |
| $\Phi(N, v)$ | if $\lvert\Phi(N, v)\rvert = 1$ also the allocation itself | 17 |
| $I(N, v)$ | set of imputations | 15 |
| $I^*(N, v)$ | set of pre-imputations | 16 |
| $\eta(N, v)$ | nucleolus of $(N, v)$ | 21 |
| $\phi(N, v)$ | Shapley value of $(N, v)$ | 31 |
| $(LC)$ | linear program describing the least core | 19 |
| $(f\text{-}LC)$ | linear program describing the $f$-least core | 24 |
| $\mathcal{F}_0$ | $\{\emptyset, N\}$ | 21 |
| $e(S, x)$ | excess of $S$ with respect to $x$ | 20 |
| $e_{min}(x)$ | $\min\{e(S, x) \mid \emptyset \neq S \neq N\}$ | 20 |
| $\theta(x)$ | excess vector with respect to $x$ | 20 |
| $e^f(S, x)$ | $f$-excess of $S$ with respect to $x$ | 26 |
| $e^f_{min}(x)$ | $\min\{e^f(S, x) \mid \emptyset \neq S \neq N\}$ | 26 |

| | | |
|---|---|---|
| $\theta^f(x)$ | $f$-excess vector with respect to $x$ | 28 |

**Polyhedral theory**

| | | |
|---|---|---|
| $P = P(A, b)$ | if $Ax \leq b$ describes $P$ | 7 |
| $Fix\, P$ | set of coalitions fixed by $P$ | 21 |
| $\varphi = \varphi(P)$ | facet complexity of $P$ | 8 |
| $P(\epsilon)$ | $\{x \in \mathbb{R}^N \mid (x, \epsilon) \in P\}$ | 21 |

**Graph theory**

| | | |
|---|---|---|
| $(G, w)$ | discrete structure | 17 |
| $G = (V, E)$ | graph | 10 |
| $V = V(G)$ | node set | 10 |
| $E = E(G)$ | edge set | 10 |
| $\|G'\|$ | size of component $G'$ | 11 |
| $G \backslash E'$ | subgraph obtained by removing $E'$ | 11 |
| $G \backslash V'$ | subgraph obtained by removing $V'$ | 11 |
| $G\|_{V'}$ | subgraph induced by $V'$ | 11 |
| $\delta(i)$ | degree of $i$ | 10 |
| $w$ | node and/or edge weighting | 10 |
| $< w >$ | maximum size of the $w$-values | 18 |
| $w(E')$ | $\sum_{e \in E'} w(e)$ | 10 |
| $w(V')$ | $\sum_{i \in V'} w_i$ | 78 |
| $K_n$ | complete graph on $n$ nodes | 10 |
| $C_n$ | cycle on $n$ nodes | 11 |
| $P = v_0 v_1 \ldots v_k$ | path with end points $v_0$ and $v_k$ | 11 |
| $(G, l, u)$ | network | 12 |
| $G = (V, A)$ | directed graph | 12 |
| $A = A(G)$ | arc set | 11 |
| $l$ | lower capacity function defined on $A$ | 12 |
| $u$ | upper capacity function defined on $A$ | 12 |
| $\delta^-(i)$ | indegree of $i$ | 12 |
| $\delta^+(i)$ | outdegree of $i$ | 12 |
| $h(a)$ | head of $a$ | 11 |
| $t(a)$ | tail of $a$ | 11 |

**Chapter 3**

| | | |
|---|---|---|
| $s$ | supply | 36 |
| $(N, c)$ | MCST-game | 36 |
| $c(S)$ | weight of an MST in $G\|_{S\cup\{s\}}$ | 36 |
| $x^*$ | standard core allocation corresponding to $T^*$ | 37 |
| $F_i(T^*)$ | set of immediate followers of $i$ in $T^*$ | 39 |
| $S_i(x)$ | set of allocations by strong demand operation on $x$ | 41 |
| $f^i(S)$ | $f^i(S) = 1$ if $S = \{i\}$, otherwise $f^i(S) = 0$ | 42 |
| $U$ | set of $k \geq q$ points | 45 |
| $W$ | set of $3q$ points | 45 |
| $St$ | Steiner node | 46 |
| $g$ | guardian | 46 |
| $\delta(S)$ | $x(S) - c(S) + \epsilon f(S)$ | 53 |
| $f^u$ | $f(N\backslash u) \quad (u \in U)$ | 48 |
| $f^w$ | $f(N\backslash w) \quad (w \in W)$ | 48 |
| $\epsilon^f(D)$ | $\frac{\|D\|+2q-1}{\|D\| f^u+3qf^w+f(N\backslash g)+f(D\cup W\cup g)}$ | 48 |
| $\epsilon^f$ | $\min\{\epsilon^f(D) \mid D \subseteq U \text{ covers } W\}$ | 49 |

**Chapter 4**

| | | |
|---|---|---|
| $(N, v)$ | matching game | 70 |
| $v(S)$ | value of a maximum weight matching in $G\|_S$ | 70 |
| $m^*$ | size of a maximum matching in $G$ | 67 |
| $T$ | Tutte set satisfying conditions Theorem 4.4 | 68 |
| $\mathcal{C} = \mathcal{C}(T)$ | set of even components of $G\backslash N'$ | 67 |
| $\mathcal{D} = \mathcal{D}(T)$ | set of odd components of $G\backslash N'$ | 67 |
| $\bigcup\mathcal{C}$ | union of all even components | 67 |
| $\bigcup\mathcal{D}$ | union of all odd components | 67 |
| $M_{\mathcal{C}}$ | perfect matching in $\bigcup\mathcal{C}$ | 67 |
| $M_{\mathcal{D}}$ | matching inducing a near-perf. matching in all $D \in \mathcal{D}$ | 67 |
| $M_{T,\mathcal{D}}$ | matching that matches $T$ completely into $\bigcup\mathcal{D}$ | 67 |
| $E(S)$ | set of edges joining nodes of $S$ | 77 |
| $N(E')$ | set of nodes covered by $E'$ | 73 |

| | | |
|---|---|---|
| $x(M)$ | $x(N(M))$ | 74 |
| $i \in D$ | $i \in N(D)$ | 78 |
| $x(D)$ | $\sum_{i \in D} x_i$ | 78 |
| $x_D$ | $\frac{x(D)}{\lvert D \rvert}$ | 83 |
| $w_D$ | $\frac{w(D)}{\lvert D \rvert}$ | 84 |
| $w^D_{min}$ | minimum weight in a component $D \in \mathcal{D}$ | 85 |
| $\bar{x}$ | allocation obtained after averaging $x$ w.r.t. $w$ | 84 |
| $\mathcal{D}_{max}$ | set of odd components on which $\bar{x}_D - w_D$ is max. | 87 |
| $\mathcal{M}$ | set of all matchings in $G$ | 76 |
| $\tilde{\mathcal{M}}^*$ | set of all maximum weight matchings in $G$ | 78 |
| $\mathcal{M}^*$ | set of all maximum matchings in $G$ | 78 |
| $\mathcal{M}_{\mathcal{D}}$ | set of all matchings contained in $E(\bigcup \mathcal{D})$ | 83 |
| $\mathcal{M}^*_{\mathcal{D}}$ | set of all maximum matchings in $\mathcal{M}_{\mathcal{D}}$ | 88 |
| $T_0 \cup \ldots \cup T_p$ | partition of $T$ | 91 |
| $\mathcal{D}_0 \cup \ldots \cup \mathcal{D}_p$ | partition of $\mathcal{D}$ (flexible) | 91 |
| $\mathcal{S}_1 \cup \ldots \cup \mathcal{S}_l$ | partition of $\mathcal{D}$ (same size) | 98 |

**Chapter 5**

| | | |
|---|---|---|
| $(\alpha, \beta, \gamma)$ | rule of the competition | 101 |
| $v^*$ | bribe of the competition | 102 |
| $\rho$ | order of the competition | 111 |
| $(\bar{N}, s, M)$ | state of the competition | 102 |
| $\bar{N}$ | set of teams | 102 |
| $s$ | current score vector | 102 |
| $M$ | set of remaining matches | 102 |
| $t_0$ | team that wants to bribe | 102 |
| $(N, v)$ | competition game | 102 |
| $N$ | $\bar{N} \backslash t_0$ | 102 |
| $v(S)$ | $v^*$ if $S \subseteq N$ is influential, otherwise $v(S) = 0$ | 102 |
| $N^*$ | set of indispensable teams | 117 |
| $G$ | multigraph $G$ corresponding to $N$ | 104 |
| $G^S$ | multigraph $G$ corresponding to $S$ | 116 |
| $c$ | node capacity function | 104 |
| $\delta^0(i)$ | number of unoriented edges incident with $i$ | 104 |

# Author Index

# Subject Index

# Summary

In game theory situations of conflict are modeled and analyzed. In such a situation two or more individuals (the players) with similar or different interests are taking actions or making decisions. In cooperative game theory players may form coalitions in order to optimize their profits (or costs).

It is often reasonable to assume that the players decide to work all together. Then the question arises how to split up the joint profit or cost. A solution concept suggests for each game a set of possible pay-offs (allocations).

The usefulness of a solution concept is not only determined by its modeling adequacy but also by its computational complexity. In this thesis we study the complexity of several solution concepts with respect to various classes of cooperative games. In all games we consider, the cost or profit is computed as the optimal value of some discrete optimization problem. More precisely, a game is defined by a graph with an associated node and/or edge weighting. The profit or cost of a coalition is determined as the value of a discrete optimization problem on this graph.

After some preliminaries in Chapter 1, in Chapter 2 we discuss a number of solution concepts for cooperative games, in particular the core and the nucleolus. We generalize these two concepts to obtain the $f$-least core and the $f$-nucleolus (special cases: the nucleon and the per-capita nucleolus).

Chapter 3 concentrates on minimum cost spanning tree games (MCST-games). In an MCST-game the players are represented by nodes in a complete graph and the cost of a coalition is equal to the weight of the corresponding minimum spanning tree. MCST-games have nonempty core, and certain core allocations can be computed in polynomial time. However, for certain reasons these core allocations may not be acceptable. We therefore study the $f$-least core of a minimum cost spanning tree game for various priority functions $f$. By a reduction from minimum cover problems we prove that for a large class

of priority functions computing an allocation in the $f$-least core of a general MCST-game is $\mathcal{NP}$-hard. As a consequence also computing $f$-nucleoli, such as the nucleolus, the nucleon and the per-capita nucleolus of MCST-games, is in general $\mathcal{NP}$-hard.

Chapter 4 deals with matching games. In a matching game the players are represented by nodes in a graph and the profit of a coalition is equal to the weight of the corresponding maximum matching. In particular, we study cardinality matching games and a generalization thereof (node matching games). We first present a simple characterization of the least core. We then use this description to construct a polynomial time algorithm for computing the nucleolus of these games. The case of general (weighted) matching games remains open.

In Chapter 5 we study complexity aspects of sports competitions like national football leagues and related games. In such a competition various teams play matches against each other in pairs according to a previously determined schedule. The central problem is the so-called elimination problem, i.e., to determine at a given intermediate state of the competition whether a particular team still has a chance of winning the competition. Our main result states that the new FIFA-rules (3 : 0 for a win) have complicated this problem considerably. We completely characterize the complexity of this problem, and relate it to the core complexity of a corresponding game, in which a team tries to bribe some other teams in order to win the competition.

# Samenvatting

Speltheorie modelleert en analyseert conflictsituaties, waarin twee of meer individuen (de spelers) optreden, die gelijke of verschillende belangen hebben. In coöperatieve speltheorie kunnen spelers coalities vormen om hun winst (of kosten) te optimaliseren.

Het is vaak redelijk om aan te nemen, dat alle spelers met elkaar gaan samenwerken. Dan doet zich de vraag voor, hoe de gezamenlijke winst of kosten verdeeld moeten worden. Een oplossingsconcept geeft voor elk spel een verzameling mogelijke uitbetalingen (allocaties).

De bruikbaarheid van een oplossingsconcept wordt niet alleen bepaald door de mate waarop het concept aansluit op de gemodelleerde situatie, maar ook door de complexiteit van het berekenen van een allocatie volgens dat concept.

In dit proefschrift bestuderen we de complexiteit van verscheidene oplossingsconcepten met betrekking tot verschillende klassen van coöperatieve spelen. In alle bestudeerde spelen worden de kosten (of de winst) berekend als de optimale waarde van een zeker discreet optimaliseringsprobleem. We definiëren een spel door middel van een graaf met gewichten op de punten en/of lijnen. De winst of de kosten van een coalitie wordt dan bepaald als de waarde van een discreet optimaliseringsprobleem op deze graaf.

Na een inleiding in Hoofdstuk 1 bespreken we in Hoofdstuk 2 een aantal oplossingsconcepten voor coöperatieve spelen. In het bijzonder besteden we aandacht aan de nucleolus en de core. We generaliseren deze twee concepten en verkrijgen op die manier de $f$-least core en de $f$-nucleolus (speciale gevallen: de nucleon en de per-capita nucleolus).

Hoofdstuk 3 behandelt minimum opspannende boomspelen (MOB-spelen). In een MOB-spel worden de spelers gerepresenteerd als punten in een complete graaf en de kosten van een coalitie worden gelijkgesteld aan het gewicht van de overeenkomstige minimum opspannende boom. MOB-spelen hebben

een niet-lege core en bepaalde core-allocaties kunnen in polynomiale tijd berekend worden. Echter deze core-allocaties zijn niet onder alle omstandigheden acceptabel. Daarom bestuderen we de $f$-least core van een MOB-spel voor verscheidene prioriteitsfuncties $f$. We bewijzen dat het berekenen van een allocatie in de $f$-least core van een algemeen MOB-spel $\mathcal{NP}$-moeilijk is voor een grote klasse van prioriteitsfuncties. Uit dit resultaat volgt dat ook het berekenen van $f$-nucleoli, zoals de nucleolus, nucleon en de per-capita nucleolus van MOB-spelen, $\mathcal{NP}$-moeilijk is.

In Hoofdstuk 4 behandelen we de matching-spelen. In een matching-spel worden de spelers gerepresenteerd als knopen in een graaf en de winst van een coalitie wordt gelijkgesteld aan het gewicht van de overeenkomstige maximale matching. We bestuderen in het bijzonder cardinaliteitsmatching-spelen en een generalisatie hiervan (puntmatching-spelen). Met behulp van een eenvoudige karakterisering van de least core construeren we een efficiënt algoritme voor het berekenen van de nucleolus van deze spelen. Het algemene geval van (gewogen) matching spelen blijft een open probleem.

In Hoofdstuk 5 gaan we in op complexiteitsaspecten van sportcompetities zoals de nationale voetbalcompetities. In een dergelijke competitie spelen verscheidene teams wedstrijden tegen elkaar volgens een van te voren vastgelegd wedstrijdschema. We bestuderen het zogenaamde eliminatieprobleem, d.w.z., het bepalen of in een gegeven tussenstand een zeker team nog steeds een kans maakt op het winnen van de competitie. Ons resultaat houdt in, dat de nieuwe FIFA-regeles (3 punten voor een overwinning) dit probleem aanzienlijk vermoeilijkt hebben. We geven een volledige karakterisering van de complexiteit van dit probleem. Ook brengen we het in verband met de complexiteit van het berekenen van een core-allocatie in een zogenaamd competitiespel, waarin een team probeert de andere teams om te kopen om zo de competitie te winnen.

# About the author

Daniël Paulusma was born on May 23, 1974, in Leeuwarden. In 1992 he began to study Applied Mathematics at the University of Twente. He wrote his master's thesis entitled "Extensions of the Shapley value for TU-games defined on posets" in 1997. In May 1997 he graduated. He had already decided to stay a little longer in Enschede, and in June 1997 he started his Ph.D. research, which resulted in this thesis.