UNIVERSITY OF SOUTHAMPTON

Computing the nucleolus of cooperative games

by

Márton Benedek

A thesis submitted in partial fulfillment for the
degree of Doctor of Philosophy

in the

Faculty of Social, Human and Mathematical Sciences
School of Mathematics

December 2019

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF SOCIAL, HUMAN AND MATHEMATICAL SCIENCES
SCHOOL OF MATHEMATICS

Doctor of Philosophy

by Márton Benedek

The computation of the nucleolus, one of the most widespread single valued solution concept of cooperative game theory has developed greatly, starting from an immensely complex lexicographical optimisation problem to solving a sequence of LPs. Focusing on the general case, in this thesis we provide an extensive review of the various existing computational methods in the literature, also providing a categorisation of these methods in a new aspect, mainly based on the process of updating in between LPs, that influences the required number of iterations. The disparity in the number of iterations between primal and dual methods lead to the classical primal-dual trade-off: one method requires minimal number of iterations (primal), while solving the large LPs in the other is easier (dual). We introduce a hybrid variant, that inherits good qualities from each method, dissolving the classical trade-off. After showing how to reduce the number of iterations for the dual sequence, we introduce a conceptually new, practical approach to one of the main tasks involved in the sequential LP framework, that is finding all coalitions in the span of some coalitions. We argue that it could be beneficial in practice not to do this, possibly leading to a decrease in computational time despite the increasing number of iterations, pivots and subroutine iterations. Through analysing the numerical results, it becomes clear, that the literature not yet considered every aspect of the primal-dual trade-off: namely *warm starting*, favouring primal over the dual. Balancedness is related to the nucleolus early on through the Kohlberg criterion verifying whether a solution is the nucleolus or not. Our results on balancedness lay the foundations for the main contribution of the thesis, a new constructive approach to compute the nucleolus. A primal based active-set method with a dual subroutine, that benefits from every advantage of primal methods, including efficient warm starting, however it does not suffer from it's disadvantage, by solving large LPs without explicitly formulating them. Results on balancedness further allow us to improve on the original Kohlberg criterion. The new active-set method outperforms other classical sequential LP models and the *dual counterpart* of the proposed method, based on extensive amount of numerical testing. Another important contribution is providing open-source codes for all algorithms and instances involved in creating those numerical results.

Statement of Authorship

I, Márton Benedek, declare that the thesis entitled "Computing the nucleolus of cooperative games" and the work presented in it are my own contribution. I confirm that:

- this work was done wholly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have quoted from the work of others, the source is always given;
- with the exception of such quotation, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Contents

List of Figures

List of Tables

Acknowledgements

This work could not have been achieved without the help of many.

First and foremost I am immensely grateful for the support of my family. In *Dorottya* I have been blessed with the partner I could not hope for. I am eternally thankful for her being there for me and for all her care and patience. *Sigma* is the companion casting light on even the darkest of our days. I thank him for being there for, and completing us. I am beyond words to express my gratitude towards *my mother*, who has set an example for me in all aspects of life, that I wish to be worthy of. Finally, I think of *Walton* as my brother, and could not consider myself luckier for our paths crossing. I am grateful for all the adventures together, and possibly even more so, for all the everydays.

One of the most joyful experiences during the years spent in Southampton working on this thesis was the camaraderie of my fellow PhD students. Getting to know them is a value that I carry with me forever through the binds of our friendships. Many of them directly or indirectly contributed toward this work being accomplished. In particular, I am especially grateful for Walton Pereira Coutinho, Ruth Walton, Simos Zachariades, Carlos Llamas Fernandez, Karl Steinborn-Busse, Laura Murray, and Hattie Hall.

This thesis would not have been completed without the invaluable help from my supervisors, Tri-Dung Nguyen and Jörg Fliege. I am immensely thankful for their work paving the way for my development. During my PhD I had the fortunate chance to have wonderful interactions with many other members of the academic staff as well. In particular I am especially grateful for getting to know Stefano Coniglio and the opportunity to learn from him, for the conversations with Christine Currie, as well as for Alain Zemkoho and Huifu Xu.

I would like to thank my examiners, Stefano Coniglio and Tamás Solymosi for their invaluable suggestions, the engaging discussion during the viva, and for all their efforts making the most out of this work.

Finally I would like to express my gratitude and acknowledge the help of everyone not mentioned, who directly or indirectly contributed towards this work. This research has been funded by EPSRC.

To my father.

Chapter 1

Introduction

In all kinds of aspects of our lives we experience that it is beneficial to cooperate. Whether we talk about groups of individuals, agents in the business environment, cities aligned in a network, companies with different resources available, objects following the laws of nature and physics, etc., we find that the *whole* can indeed be more than the *sum of the parts*.

The need of sharing the additional value created by cooperation between the constituting members arises automatically. A natural requirement from such an *allocation* is to be *fair*, achieving equality, proportionality and most importantly *stability*. In order to mathematically formalise these concepts, we start by a brief introduction to cooperative game theory, introducing most of the notions we need in Section 1.1, followed by a literature review in Section 1.2.

1.1 Cooperative game theory

We consider cooperative games with transferable utilities given in characteristic function form. That is, a cooperative game is an ordered pair (N, v) where N is the *player set* and $v : 2^N \to \mathbb{R}$ is the *characteristic function* mapping the power set $2^N = \{S \subseteq N\}$ of N into the reals. The value $v(S)$ assigns a real number to each *coalition* $S \subseteq N$, representing the payoff players in coalition S could achieve by cooperating, independently of other players. By assumption $v(\emptyset) = 0$. We also assume that N is finite and whenever not stated explicitly fixed as $N = \{1, \ldots, n\}$, thus simplify the notation of a game to v. Denote with $\mathcal{N} = \{S \subsetneq N, S \neq \emptyset\}$ the collection of *proper* coalitions that are non-empty proper subsets of the player set N. Generally in our notation subscript indices refer to coordinates of vectors, while superscript indices in parentheses refer to iteration numbers. The sole exception for the former being that we assume an arbitrarily fixed order of proper coalitions $\{S_1, \ldots, S_{2^n-2}\}$ in $\mathcal{N}$.

We are interested in a solution dividing the *grand coalition* value $v(N)$ among the players by a payoff vector $x \in \mathbb{R}^n$, where x_i represents the payoff allocated to player i. Introducing the notation $x(S) = \sum_{i \in S} x_i$, such solutions are called *preimputations* (or sometimes *efficient* payoffs), denoted by $PI(v) = \{x \in \mathbb{R}^n : x(N) = v(N)\}$. It is easy to see that the preimputation set $PI(v)$ is an $n-1$ dimension hyperplane of $\mathbb{R}^n$. We assume that v is a *profit game*, that is, the larger $x(S)$ is, the better solution x is for coalition S. Everything we cover in this thesis can be converted with suitable modifications to the case of a *cost game*, where a payoff vector is an allocation of costs, not profits.

There are many solution concepts in cooperative game theory. Among efficient solutions if we further impose the requirement of *individual rationality* we arrive to the set of *imputations* $I(v) = \{x \in PI(v) : x_i \geq v(\{i\}) \ \forall i \in N\}$, a polyhedron in the hyperplane $PI(v)$, possibly empty though. Shapley (1955) introduced the solution of payoff vectors that are not just individually, but *coalitionally rational*, that is the *core* $C(v) = \{x \in I(v) : x(S) \geq v(S) \ \forall S \subsetneq N\}$. The core, just like the imputation set can be empty. Whenever non-empty, core allocation vectors achieve fairness and stability by allocating to every coalition at least the payoff the coalition could achieve on its own.

A coalitions (dis)satisfaction with a solution $x \in \mathbb{R}^n$ can be measured by the *excess* $E(S, x) = v(S) - x(S)$. Denote the *excess function* by $\overline{E} : \mathbb{R}^n \to \mathbb{R}^{2^n-2}$ such that $\overline{E}(x)_i = E(S_i, x)$ for any $x \in \mathbb{R}^n$. The excess values $E(S, x)$ play a central role in establishing fairness of solutions. For example even though the core can be empty, there is always a large enough $\epsilon \in \mathbb{R}$ for which the *ϵ-core* $C_\epsilon(v) = \{x \in PI(v) : E(S, x) \leq \epsilon \ \forall S \in \mathcal{N}\}$ is non-empty. Naturally $C(v) = C_0(v)$ and $C_{\epsilon_1}(v) \subseteq C_{\epsilon_2}(v)$ whenever $\epsilon_1 \leq \epsilon_2$, so we are interested in the lowest value of ϵ, such that the ϵ-core is non-empty, that is $\epsilon^* = \min_{\epsilon \in \mathbb{R}} \{\epsilon : C_\epsilon(v) \neq \emptyset\}$. The ϵ^* attaining that minimum is called the *least core value* and $LC(v) := C_{\epsilon^*}(v)$ is called the *least core*. Evidently the core of a game is non-empty if and only if $\epsilon^* \leq 0$.

Our aim is to find a solution called the *nucleolus*, due to Schmeidler (1969). Nucleolus solutions lexicographically minimise among the imputations the (non-increasingly) ordered excess vector $\Theta(\overline{E}(x))$, where $\Theta(\overline{E}(x))_1 \geq \Theta(\overline{E}(x))_2 \geq \cdots \geq \Theta(\overline{E}(x))_{2^n-2}$. If ν is a nucleolus solution of game v then $\Theta(\overline{E}(\nu)) \leq_L \Theta(\overline{E}(x))$ for all imputations $x \in I(v)$ with $\leq_L$ being the usual lexicographic ordering[1] on $\mathbb{R}^{2^n-2}$. We arrive to the prenucleolus solution if we do not require individual rationality, that is lexicographically minimising the ordered excess vectors among preimputations. Schmeidler (1969) showed that if $I(v) \neq \emptyset$ then the nucleolus exists and consists of a single point. Also, the prenucleolus lies in the ϵ-core whenever $C_\epsilon(v) \neq \emptyset$, thus in the least core $LC(v)$ as well. If the imputation set $I(v)$ intersects the least core, then the same applies to the nucleolus ν.

[1] For $x, y \in \mathbb{R}^m : x \leq_L y \iff$ either $x = y$ or $\exists i \in \{0, 1, \ldots, m-1\} : x_{i+1} < y_{i+1}$, and if $i > 0$ then $x_j = y_j \ \forall j \in \{1, \ldots, i\}$.

The theory of cooperative games (with transferable utilities in our case) is a very powerful modelling tool for various kinds of fair division problems, whether they come from economics, business, finance, political science, among many other possible application areas.

As we have seen, the field not only offers means of modelling problems, but also a plethora of solution concepts. Among the most widely known are the core, the Shapley value (Shapley (1953)) and the nucleolus. Between the latter two, that is, among the most famous single-valued solution concepts only the nucleolus lies in the core, provided that it is not empty.

The reason for certain solution concepts being outstandingly widespread is the vast amount of attractive properties they possess, unfortunately however, computational ease is mostly not among these. There are certain classes of games for which calculating the above mentioned solutions can be done efficiently (in the sense of requiring polynomial time in the number of players), but in general, and in terms of challenging problem classes there are still unresolved computational problems.

With that mindset, in this thesis we are focusing on calculating the nucleolus of general cooperative games, that is, not supposing any special structure on the characteristic function of the game. Understanding the difficulty of the task one has to temper expectations, nevertheless, we are still able to significantly outperform existing methods with our proposed various new approaches. Therefore, throughout the remainder of the thesis, when we mention an algorithm being *efficient*, we refer to *computationally outperforming the competition.*

Our main contribution lies in providing a new algorithm, that one can see as the state-of-the-art implementation of the simplex method for the computation of the nucleolus. On the other hand, it equally can be viewed as an active set method, or a primal counterpart of the best known pivoting algorithm from Derks and Kuipers (1997). In parallel with the new algorithm, we introduce an efficient version of the Kohlberg (1971) criteria verifying whether a candidate solution is the nucleolus or not. Furthermore, we introduce a classification of the various computational methods from the previous 50 years, and dissolve a fundamental trade-off between classical primal and dual sequential LP methods by introducing new hybrid variants of these. We thoroughly investigate each task needs accomplishing in these algorithms, offering various solution methods and highlighting another, largely overlooked trade-off between primal and dual approaches. All of our findings are backed up by extensive amount of numerical results, and among one of the main targets of this project, we provide an open-source code Benedek (2018), that we believe carries significant added value both to this thesis and the field.

1.2 Literature review

We are interested in the computation of one of the most important solution concepts of cooperative game theory, the nucleolus. It consists of payoff vectors, that lexicographically minimise the largest dissatisfactions among all groups of players. Schmeidler (1969) showed not only that a cooperative game with a non-empty imputation set has a nucleolus solution, it is also unique, it lies in the core (if it is non-empty)[2] and it is a continuous function of the underlying characteristic function describing the game. Later on Sobolev (1975) axiomatised the closely related solution concept of the prenucleolus, having similarly attractive properties. With the only difference between the two solution concepts being that for the nucleolus we only consider efficient payoff vectors which are also individually rational, it is not surprising that the method of computation can be very similar. Most algorithms tailored to find one can be straightforwardly adapted to calculate the other. Thus while we present the computational approaches in terms of the nucleolus, we remark that, with suitable modifications, the results shown are easily extendable for the case of the prenucleolus.

While the breakthrough paper of Schmeidler (1969) focuses on fundamental theoretical results on the nucleolus of cooperative games, the solution concept has been already introduced by the author in 1966-67 research memoranda, as a result there were practical results preceding the publication regarding the computation of the solution. The calculation by means of a sequence of linear programs (LP), along with the closely related solution concept of the kernel, was first suggested by Kopelowitz (1967). The sequential LP model proved to be a very efficient tool for the lexicographical minimisation of the excess (dissatisfaction) levels and been refined through the following five decades. The major milestones are briefly discussed in the upcoming paragraphs.

In terms of the LPs forming the sequence, most of the methods, among which we are primarily focusing on the ones concerning general games (i.e. without assuming any structure in the characteristic function) can be classified as primal or dual methods. While the former works with the original (primal) problem of minimising dissatisfactions (or equivalently maximising satisfactions) of coalitions, the latter is concerned with the dual problem, looking for balanced collections of coalitions. The connection between the nucleolus and balancedness of certain collections of coalitions is established early on by Kohlberg (1971). The Kohlberg conditions provide a very powerful method to verify whether a payoff vector is the nucleolus solution or not. Unfortunately the nature of the method is not constructive and even checking the condition is a challenging task from a computational viewpoint.

Kohlberg (1972) also showed how one can calculate the nucleolus of any n-player cooperative game using a single LP with n variables and $(2^n)!$ constraints. Then Owen (1974)

[2] It also lies in the ϵ-core for all ϵ such that it is non-empty and $I(v) \cap C_\epsilon(v) \neq \emptyset$.

improved the single LP setting to $\mathcal{O}(2^n)$ variables and $\mathcal{O}(4^n)$ constraints, but coefficients in the LP increased greatly. Even though in the following years the sequential LP approach gained increased interest compared to the single-LP approaches (mainly due to the intractable size of the latter), recently Puerto and Perea (2013) showed another single LP approach calculating the nucleolus with $\mathcal{O}(4^n)$ constraints, $\mathcal{O}(4^n)$ decision variables and coefficients in $\{-1, 0, 1\}$ arising from a result in facility location problems (Ogryczak and Tamir (2003)).

Maschler et al. (1979) showed that their lexicographic center solution coincide with the nucleolus, which can be calculated by solving a sequence of $\mathcal{O}(4^n)$ (primal) LPs having $\mathcal{O}(2^n)$ constraints, $n+1$ variables and coefficients of also -1, 0 and 1. Bruyneel (1979)'s method can be viewed as the formers dual, calculating the nucleolus by means of minimal balanced collections of coalitions. Behringer (1981)'s method is a simplex based algorithm calculating the solution with a sequence of $\mathcal{O}(2^n)$ LPs likewise. Dragan (1981)'s dual method based on coalition arrays found the nucleolus solving only $n-1$ LPs at most, with $\mathcal{O}(n)$ constraints and $\mathcal{O}(2^n)$ variables[3]. Sankaran (1991) needed $\mathcal{O}(2^n)$ LPs with $\mathcal{O}(2^n)$ rows and $n+1$ columns, but the number of constraints were actually more than compared to Behringer's approach. In his PhD thesis Solymosi (1993) not only showed the first polynomial algorithm calculating the nucleolus of assignment games (later published in Solymosi and Raghavan (1994)), his primal LP sequence improved on the lexicographical center of Maschler et al. (1979) finding the solution with a sequence of at most $n-1$ LPs having $\mathcal{O}(2^n)$ rows and $n+1$ columns. He also unified the approach with a dual LP sequence of $n-1$ programs having $(n+1) \times \mathcal{O}(2^n)$ size, justifying Dragan (1981)'s dual sequence. Later on Fromen (1997) shows a method using Behringer's approach improving the number of LPs to $\mathcal{O}(n)$.

Despite the overwhelming literature on the *classical* sequential LP setting, there were different approaches as well. Justman (1977) showed a bargaining model considering *nucleolar* constraints. Potters et al. (1996) exploits that the LPs in the sequence are very closely related to each other, and provides a *prolonged* simplex method that calculates the nucleolus with $n-1$ LPs of size $\mathcal{O}(2^n) \times \mathcal{O}(2^n)$. Derks and Kuipers (1997) consider another implementation of the simplex method for the calculation of the prenucleolus, and also improves the complexity of one pivot step from $\mathcal{O}(n2^n)$ to $\mathcal{O}(2^n)$.

Kohlberg (1971)'s condition opened another stream of work concerning verification through balancedness considerations. Kohlberg's original criterion was expressed (among another, somewhat less practical way of *coalition arrays*) as checking the balancedness of collection of coalitions having dissatisfaction up to a certain level. Then one has to find balancing weights, *worst-case* for as many collections as there are distinctive dissatisfaction levels there are in a game at the nucleolus payoff point. Unfortunately this can lead up to checking $\mathcal{O}(2^n)$ collections, some of them possibly exponential in

[3]It is interesting to note, that Potters et al. (1996) argued that the number of LPs needed might be more.

Table 1.1: Categorisation of algorithms computing the nucleolus of general cooperative games

	type	# of LPs	# of constr.	# of var.
Kohlberg (1972)	primal	1	$(2^n)!$	n
Owen (1974)	primal	1	$\mathcal{O}(4^n)$	$\mathcal{O}(2^n)$
Puerto and Perea (2013)	primal	1	$\mathcal{O}(4^n)$	$\mathcal{O}(4^n)$
Maschler et al. (1979)	primal	$\mathcal{O}(4^n)$	$\mathcal{O}(2^n)$	$n+1$
Bruyneel (1979)	dual	$\mathcal{O}(4^n)$	$n+1$	$\mathcal{O}(2^n)$
Behringer (1981)	primal	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$
Dragan (1981)	dual	$n-1$	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$
Sankaran (1991)	primal	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$
Solymosi (1993)	primal dual	$n-1$	$\mathcal{O}(2^n)$	$n+1$
	dual	$n-1$	$n+1$	$\mathcal{O}(2^n)$
Fromen (1997)	primal	$\mathcal{O}(n)$	$\mathcal{O}(2^n)$	$n+1$
Potters et al. (1996)	primal-dual	$n-1$	$\mathcal{O}(2^n)$	$\mathcal{O}(2^n)$
Derks and Kuipers (1997)	dual	$n-1$	$n+1$	$\mathcal{O}(2^n)$
Nguyen and Thomas (2016)	primal primal	$n-1$	$\mathcal{O}(2^n)$	$n+1$
Benedek et al. (2018)	primal dual	$n-1$	$\mathcal{O}(2^n)$	$n+1$

size as well. Kido et al. (2008) introduced an equivalent criterion, which alone does not help in practice, but lead the author to find a *nonlinear approximation* of the nucleolus. Recently Benedek et al. (2018) showed an equivalent criteria verifying whether a payoff is the nucleolus by checking only at most $(n-1)$ collection of coalitions, and the size of the collections to store is at most n.

Another important area regarding the computation of the nucleolus is *characterisation sets*. It is of particular interest whether actually all the coalitions are needed to characterise the nucleolus of a game, and in certain cases which coalitions are important, or which coalitions can be neglected. A breakthrough result for balanced games is due to Huberman (1980), showing that only essential coalitions are needed. The notion of characterisation sets first appear in Oswald et al. (1995) and Granot et al. (1998). Later on Reijnierse and Potters (1998) shows that at most $2(n-1)$ coalition is enough to determine the nucleolus of any game. Recently Solymosi and Sziklai (2015) and Solymosi and Sziklai (2016) gave universal characterisation of balanced games.

When it comes to solving the LPs related to the least core or the nucleolus, naturally column/constraint generation is an option to consider. Early on Wolsey (1976) considered valid inequalities for simple games, Hallefjord et al. (1995) generated constraints in the case when the characterisation function is given implicitly (by an optimisation problem). Recently Nguyen and Thomas (2016) provides column generation framework applicable to various classes of combinatorial (e.g. flow) games, while Le et al. (2016) find least core allocations of generalised minimum spanning tree games by introducing a constraint generation algorithm.

Further related work includes Brune (1983) examining the linear independence of characteristic vectors of coalitions in *array-balanced coalition arrays*. Elkind et al. (2007) shows that finding the nucleolus of weighted voting games (WVGs) is NP-hard and subsequently Elkind and Pasechnik (2009) gives a pseudo-polynomial algorithm as well. Later on the claim that the latter algorithm is indeed pseudo-polynomial proves to be incorrect. However, recently Pashkovich (2018) showed another pseudo-polynomial algorithm. Leng and Parlar (2010) shows a good overview of some methods and provides an analytic solution for the nucleolus of 3-player games. Finding the nucleolus of a cooperative game is indeed not a trivial task (in general), Guajardo and Jörnsten (2015) highlights some common mistakes that occur in certain publications. This work aims to be self-contained in terms of cooperative game theoretic notions, but if needed we can refer the reader to Peleg and Sudhölter (2007) for a comprehensive introduction to the theory of cooperative games.

The outline of the remainder of the thesis is the following. Chapter 2 is devoted to introduce the sequential LP modelling framework, with Sections 2.1 and 2.2 building up the primal and dual LP sequence formulations, and identifies the main difficulties in the calculation. Sections 2.3 and 2.4 offer possibilities of handling some of these issues. Subsequently Chapter 3 is devoted entirely to addressing the main problem of solving the large LPs involved in the sequence. We provide a new implementation of the simplex method as a special kind of active set method in Section 3.1, and afterwards compare (theoretically) to the only existing similar type of algorithm in the literature. Detailed comparison in practice carried out on a small example can be found in Section 3.2, while Section 5.4 contains thorough numerical tests. Chapter 4 also includes a new and improved Kohlberg criterion (Section 4.1), and new sequential LP methods dissolving a classical primal-dual trade-off (Section 4.2). Chapter 5 is devoted to assess the computational performance of all algorithms covered, as well as a new conceptual approach handling a difficult task of most nucleolus computing algorithms (Section 5.5). Finally, Chapter 6 concludes, while Appendix A contains further numerical results.

Chapter 2

Sequential LP model

In this chapter we start by formulating the sequential LP framework computing the nucleolus. Following the primal and dual LP sequences we focus on two of the main difficulties arising in these algorithms, finding the minimal tight set and exploring the linear span of collections of coalitions.

2.1 Primal LP sequence

A reasonable strategy to lexicographically minimise ordered excesses is to progress level by level in terms of dissatisfaction. Let us start by finding the least core, that is, first minimise the largest excess value $E(S, x)$, which can be formulated as an LP minimising an upper bound on all proper coalitions' excess while staying in the imputation set:

$$\mathbf{P^{(1)}} = \min_{x^{(1)} \in I(v), \epsilon^{(1)}} \{\epsilon^{(1)} : \epsilon^{(1)} + x^{(1)}(S) \geq v(S) \ \forall S \in \mathcal{N}\}.$$

Having exponential number of constraints, $\mathbf{P^{(1)}}$ is an LP already quite challenging to solve. In fact, if $I(v) \cap LC(v) \neq \emptyset$, then the optimal objective function value $\epsilon^{(1)*}$ is the least core value, which is NP-hard to find for many classes of games. Supposing that we found it though, the sequential LP framework proceeds by *settling* some coalitions in $\mathcal{N}$ assigned to have that largest excess $\epsilon^{(1)*}$, then we can minimise the second-largest excess among the remaining coalitions. Before turning to how to find coalitions S to settle, having $E(S, x) = \epsilon^{(1)*}$ for any solution x of $\mathbf{P^{(1)}}$, we underline the importance of the matter in the followings.

Let us denote the feasible points of $\mathbf{P^{(1)}}$ with $\mathcal{P}^{(1)}$ and the optimal solutions with $\mathcal{P}^{(1)*} \subseteq \mathcal{P}^{(1)}$. Further denote the *tight set* (or active set) by $\mathcal{T}^{(1)}(x, \epsilon)$ for any $(x, \epsilon) \in \mathcal{P}^{(1)}$, the set of coalitions that for the corresponding inequality constraints in $\mathbf{P^{(1)}}$ are tight, i.e. $\epsilon + x(S) = v(S)$ for all $S \in \mathcal{T}^{(1)}(x, \epsilon)$. Since $\epsilon^{(1)*}$ is unique for a given game v, with a

slight abuse of notation we write $x^{(1)} \in \mathcal{P}^{(1)*}$, and similarly $\mathcal{T}^{(1)}(x^{(1)}) = \mathcal{T}^{(1)}(x^{(1)}, \epsilon^{(1)*})$ for $x^{(1)} \in \mathcal{P}^{(1)*}$.

By construction, we have $\nu \in \mathcal{P}^{(1)*}$, unfortunately with possibly many other solutions. We might find for example $x, y \in \mathcal{P}^{(1)*}$ with $|\mathcal{T}^{(1)}(x)| < |\mathcal{T}^{(1)}(y)|$. This has great significance, as the following claim shows.

Remark 2.1. $|\mathcal{T}^{(1)}(\nu)| \leq |\mathcal{T}^{(1)}(y)|$ for all $y \in \mathcal{P}^{(1)*}$.

Proof. Suppose $x, y \in \mathcal{P}^{(1)*}$ and $j = |\mathcal{T}^{(1)}(x)| < |\mathcal{T}^{(1)}(y)|$. Then it follows that $\Theta(\overline{E}(x))_i = \Theta(\overline{E}(y))_i = \epsilon^{(1)*}$ for all $i \leq j$, while $\Theta(\overline{E}(x))_{i+1} < \epsilon^{(1)*} = \Theta(\overline{E}(y))_{i+1}$. Hence $\Theta(\overline{E}(x)) <_L \Theta(\overline{E}(y))$, yielding $y \neq \nu$. □

Remark 2.1 notes that any solution $x \in \mathcal{P}^{(1)*}$ must have the smallest possible tight set size $|\mathcal{T}^{(1)}(x)|$ in order to be a candidate for ν. Luckily, there exists a unique tight set with this property (cf. Elkind and Pasechnik (2009)).

Definition 2.2. Denote with $\mathcal{T}^{(1)*}$ the unique *minimal tight set* such that $|\mathcal{T}^{(1)*}| \leq |\mathcal{T}^{(1)}(x)|$ for all $x \in \mathcal{P}^{(1)}$ and $\mathcal{T}^{(1)*} = \mathcal{T}^{(1)}(x^*)$ for some $x^* \in \mathcal{P}^{(1)}$.

We find that not only $\mathcal{T}^{(1)*}$ is a unique tight set with minimal size, but it consists precisely of the constraints that are tight for all solutions in $\mathcal{P}^{(1)*}$.

Lemma 2.3. *For every* $x^{(1)} \in \mathcal{P}^{(1)*} : \mathcal{T}^{(1)*} \subseteq \mathcal{T}^{(1)}(x^{(1)})$ *and thus*

$$\mathcal{T}^{(1)*} = \bigcap_{x^{(1)} \in \mathcal{P}^{(1)*}} \mathcal{T}^{(1)}(x^{(1)}) \neq \emptyset$$

Proof. Let $x^{(1)}, x^{(1)'} \in \mathcal{P}^{(1)*}$ be arbitrary, $A = \mathcal{T}^{(1)}(x^{(1)}) \backslash \mathcal{T}^{(1)}(x^{(1)'})$ and $B = \mathcal{T}^{(1)}(x^{(1)'}) \backslash \mathcal{T}^{(1)}(x^{(1)})$. Suppose that $A \neq \emptyset$. By convexity $x^\lambda = \lambda x^{(1)} + (1-\lambda)x^{(1)'} \in \mathcal{P}^{(1)*}$ for all $\lambda \in [0, 1]$ and it is easy to see that

$$x^\lambda(S) + \epsilon^{(1)*} > v(S), \quad \forall S \in A$$

Hence $\mathcal{T}^{(1)}(x^\lambda) \subsetneq \mathcal{T}^{(1)}(x^{(1)})$. With similar argument for B, we can conclude that

$$\mathcal{T}^{(1)}(x^\lambda) = \mathcal{T}^{(1)}(x^{(1)}) \cup \mathcal{T}^{(1)}(x^{(1)'}) \setminus (A \cup B)\,.$$

Now if we have $\mathcal{T}^{(1)}(x^{(1)'}) = \mathcal{T}^{(1)*}$ then $B = \emptyset$ and $\mathcal{T}^{(1)}(x^{(1)'}) \subseteq \mathcal{T}^{(1)}(x^{(1)})$. If the latter would not hold, then $|\mathcal{T}^{(1)}(x^\lambda)| < |\mathcal{T}^{(1)*}|$, a contradiction. Hence $\mathcal{T}^{(1)*} \subseteq \mathcal{T}^{(1)}(x^{(1)})$ for any $x^{(1)} \in \mathcal{P}^{(1)*}$ and it is a proper subset if A is not empty. The equality follows.

Finally we remark that $\mathcal{T}^{(1)*}$ is not empty, as otherwise $\exists x^{(1)*} \in \mathcal{P}^{(1)*}$ such that $\epsilon^{(1)*} + x^{(1)*}(S) > v(S)$ for all $S \in \mathcal{N}$, contradicting $x^{(1)*} \in \mathcal{P}^{(1)*}$. □

It is straightforward to see, that the result of Lemma 2.3 holds for any LP, since we only needed the convexity of the set of optimal solutions and the linearity of the constraints. Thus the minimal tight set is the maximal (in size) set of constraints that are tight for all the solutions of an LP. In optimisation it also known as the *optimal partition*. Note that if one is aiming to solve a problem by constraint generation, generating constraints in the minimal tight set is the ultimate goal, as this set of constraint is necessary and sufficient to yield an optimal solution in the relaxation.

Even though $\epsilon^{(1)*}$ and the minimal tight set $\mathcal{T}^{(1)*}$ are unique, typically there are multiple solutions having minimal tight set, we distinguish them with the notation $x^{(1)*} \in \mathcal{P}^{(1)*} : \mathcal{T}^{(1)}(x^{(1)*}) = \mathcal{T}^{(1)*}$. Later on in Section 2.3 we address the question how to find $x^{(1)*}$ from an arbitrary $x^{(1)} \in \mathcal{P}^{(1)*}$ solution, or how to find coalitions belonging to the minimal tight set $\mathcal{T}^{(1)*}$ from the collection $\mathcal{T}^{(1)}(x^{(1)})$. For now let us suppose that we find $x^{(1)} \in \mathcal{P}^{(1)*}$ and the minimal tight set $\mathcal{T}^{(1)*}$. For all $S \in \mathcal{T}^{(1)*}$ let us define $y^*_S = E(S, x^{(1)}) = \epsilon^{(1)*}$. Satisfaction of coalitions S in the minimal tight set are constantly $-y^*_S$ through the solution space $\mathcal{P}^{(1)*}$ of $\mathbf{P^{(1)}}$, hence it does not make sense trying to further lower their upper bound. Therefore in order to proceed with minimising the second largest excess, we remove these coalitions from the set of inequality constraints, having an upper bound $\epsilon^{(2)}$ only on coalitions whose excess is less than $\epsilon^{(1)*}$ somewhere in $\mathcal{P}^{(1)*}$. In the same time we introduce the equality constraints $x^{(2)}(S) = v(S) - y^*_S$ for the *settled coalitions* in $\mathcal{T}^{(1)*}$ to ensure $E(S, x^{(2)}) = E(S, \nu)$ on these coalitions. Let us denote the collection of settled coalitions by Δ. Based on $\mathbf{P^{(1)}}$ we can say that N is settled at the beginning ($\Delta^{(0)} = N$) with $y^*_N = 0$, while after solving $\mathbf{P^{(1)}}$, $\mathcal{T}^{(1)*}$ gets settled, hence $\Delta^{(1)} = N \cup \mathcal{T}^{(1)*}$ with $y^*_S = \epsilon^{(1)*}$ for all $S \in \mathcal{T}^{(1)*}$.

To this point we examined the structure of tight sets of different solutions regarding the dissatisfaction constraints, however have not yet considered the imputation constraints $x^{(1)}_i \geq v(\{i\}), i \in N$. Therefore let us introduce *imputation-tight sets* $\widetilde{\mathcal{T}}^{(1)}(x^{(1)}) = \left\{\{i\} : i \in N, x^{(1)}_i = v(\{i\})\right\}$, collecting players as singleton coalitions, for which feasible solution $x^{(1)} \in \mathcal{P}^{(1)}$ is on the boundary of violating individual rationality. Analogously to the tight set case we can consider $\widetilde{\mathcal{T}}^{(1)*} = \left\{\{i\} : i \in N, x^{(1)}_i = v(\{i\}) \ \forall x^{(1)} \in \mathcal{P}^{(1)*}\right\}$. Naturally

- if $\{i\} \in \widetilde{\mathcal{T}}^{(1)}(x^{(1)})$ for some $x^{(1)} \in \mathcal{P}^{(1)*}$, then $\epsilon^{(1)*} \geq 0$,
- if $\epsilon^{(1)*} = 0$, then $\{i\} \in \mathcal{T}^{(1)*} \iff \{i\} \in \widetilde{\mathcal{T}}^{(1)*}$[1],
- if $\epsilon^{(1)*} > 0$, then $\{i\} \notin \mathcal{T}^{(1)}(x^{(1)})$ for all $x^{(1)} \in \mathcal{P}^{(1)*}$ and
- if $\epsilon^{(1)*} < 0$, then $\{i\} \notin \widetilde{\mathcal{T}}^{(1)}(x^{(1)})$ for all $x^{(1)} \in \mathcal{P}^{(1)*}$.

As a consequence, we include $\widetilde{\mathcal{T}}^{(1)*}$ in the settled coalitions $\Delta^{(1)} = N \cup \mathcal{T}^{(1)*} \cup \widetilde{\mathcal{T}}^{(1)*}$ with $y_{\{i\}} = 0$ for $\{i\} \in \widetilde{\mathcal{T}}^{(1)*}$.

[1]Therefore to avoid unnecessary duplication, we only consider $\{i\} \in \mathcal{T}^{(1)*}$ in this case.

Notice however that the additional set of equality constraints regarding coalitions in $\mathcal{T}^{(1)*}$ and $\widetilde{\mathcal{T}}^{(1)*}$ together with the preimputation constraint $E(N, x^{(2)}) = 0$ might not have full rank. In order to assess linear dependency between sets of coalitions, we define the *characteristic vector* of coalitions S as $e(S) \in \{0,1\}^n$ (assumed to be a row-vector) with $e(S)_i = 1$ if and only if $i \in S$. Then we say that coalition S is in the linear span of the collection of coalitions $\mathcal{T} \subseteq 2^N$, denoted by $S \in \text{span}(\mathcal{T})$, if $e(S) \in \text{span}(\{e(T) : T \in \mathcal{T}\})$, that is $\exists \mu \in \mathbb{R}^{|\mathcal{T}|} : e(S) = \sum_{T \in \mathcal{T}} \mu_T e(T)$.

The rank of a set of coalitions comes naturally with this definition: $\text{rank}(\mathcal{T})$ is the minimal number of coalitions that span every coalitions in $\mathcal{T}$. Then by the above observation we only have to include coalitions that actually increase the rank of Δ. Thus let us introduce a *representative set* of settled coalitions Δ_r, initialised in the same manner by $\Delta_r^{(0)} = N$. Then we expand this representative set to achieve $S \in \text{span}(\Delta_r^{(1)})$ for all $S \in \mathcal{T}^{(1)*} \cup \widetilde{\mathcal{T}}^{(1)*} \cup N$ and $|\Delta_r^{(1)}| = \text{rank}(\Delta_r^{(1)})$. As a result the excess of coalitions in $\text{span}(\mathcal{T}^{(1)*} \cup \widetilde{\mathcal{T}}^{(1)*} \cup N) = \text{span}(\Delta_r^{(1)})$ are completely determined by the current set of equality constraints:

$$\begin{aligned} e(S) &= \sum_{T \in \Delta_r^{(1)}} \mu_T e(T) \\ x(S) &= \sum_{T \in \Delta_r^{(1)}} \mu_T x(T), \end{aligned}$$

where the second equality arises from the first by multiplying with an arbitrary $x \in \mathbb{R}^n$. In light of this, there is also no reason to try minimising an upper bound regarding these coalitions excesses, hence we omit these coalitions as well from the inequality constraints, ending up with a constraint set $\epsilon^{(2)} + x^{(2)}(S) \geq v(S)\ \forall S \notin \text{span}(\Delta_r^{(1)})$. Analogously for the imputation constraints, we only require $x_i^{(2)} \geq v(\{i\})\ \forall \{i\} \notin \text{span}(\Delta_r^{(1)})$, as for considering the rest of the players, remaining in the imputation set is automatically satisfied by the equality constraints for $\Delta_r^{(1)}$.

This yields the next level LP, finding coalitions with the minimal second-worst excess among the remaining, *unsettled* coalitions:

$$\mathbf{P^{(2)}} = \min_{x^{(2)}, \epsilon^{(2)}} \Big\{ \epsilon^{(2)} : \begin{aligned} \epsilon^{(2)} + x^{(2)}(S) &\geq v(S) && \forall S \notin \text{span}(\Delta_r^{(1)}), \\ x_i^{(2)} &\geq v(\{i\}) && \forall \{i\} \notin \text{span}(\Delta_r^{(1)}), \\ x^{(2)}(S) &= v(S) - y_S^* && \forall S \in \Delta_r^{(1)} \Big\}, \end{aligned}$$

where $y_S^* = \epsilon^{(1)*}$ for all $S \in \mathcal{T}^{(1)*}$ and $y_N^* = 0$. In order to generalise this idea we define $\mathcal{T}^{(0)*} = N$, $\widetilde{\mathcal{T}}^{(0)*} = \emptyset$ and for all k the representative set of settled coalitions $\Delta_r^{(k)}$ such that $S \in \text{span}(\Delta_r^{(k)})$ for all $S \in \bigcup_{i=0}^{k} \left(\mathcal{T}^{(i)*} \cup \widetilde{\mathcal{T}}^{(i)*} \right)$ and $|\Delta_r^{(k)}| = \text{rank}(\Delta_r^{(k)})$, the *unsettled coalitions* $\Sigma^{(k)} = 2^N \setminus \text{span}(\Delta_r^{(k)})$, the *unsettled players* $\Gamma^{(k)} = \{\{i\} : i \in N\} \setminus \text{span}(\Delta_r^{(k)})$ and $y_S^* = \epsilon^{(k)*}$ for all $S \in \mathcal{T}^{(k)*}$. Then the k-th level LP can be formulated

as

$$\begin{array}{lrll} \min\limits_{x^{(k)},\epsilon^{(k)}} & \epsilon^{(k)} & & \\ \text{s.t.} & \epsilon^{(k)} + x^{(k)}(S) & \geq v(S) & \forall S \in \Sigma^{(k-1)} \\ & x_i^{(k)} & \geq v(\{i\}) & \forall \{i\} \in \Gamma^{(k-1)} \\ & x^{(k)}(S) & = v(S) - y_S^* & \forall S \in \Delta_r^{(k-1)} \end{array} \qquad (\mathbf{P^{(k)}})$$

The sequence of (primal) LPs ($\mathbf{P^{(k)}}$) are almost the same as Solymosi (1993)'s primal LP sequence. The main difference is the less number of equalities due to $\Delta_r^{(k)}$ being just a representative set of the settled coalitions $\Delta^{(k)}$ compared to Solymosi (1993)'s $x^{(k)}(S) \geq v(S) - y_S^*$ inequalities for all settled coalitions $S \in \Delta^{(k)}$. However, those inequalities are tight (active) in any optimal solution of the LP, thus even though the feasible space in Solymosi (1993)'s primal LP sequence is larger, the set of optimal solutions are exactly the same. Consequently, the following claim is not surprising.

Proposition 2.4. *After solving* ($\mathbf{P^{(k)}}$) *at most* $n-1$ *times* ν *is the unique solution to* $x(S) = v(S) - y_S^*$ $\forall S \in \Delta_r^{(k)}$.

Proof. It follows straightforward from $\text{rank}(\Delta_r^{(0)}) = 1$ and that $\text{rank}(\Delta_r^{(k)}) > \text{rank}(\Delta_r^{(k-1)})$ for all k such that $\text{rank}(\Delta_r^{(k-1)}) < n$, because $\Sigma^{(k-1)} \supseteq \mathcal{T}^{(k)*} \neq \emptyset$ (furthermore $\widetilde{\mathcal{T}}^{(k)*} \subseteq \Gamma^{(k-1)}$) given $\text{rank}(\Delta_r^{(k-1)}) < n$. □

2.2 Dual LP sequence

Naturally there is a dual counterpart of ($\mathbf{P^{(k)}}$).

$$\begin{array}{lllll} \max\limits_{u^{(k)},w^{(k)},z^{(k)}} & \sum\limits_{S\in\Sigma^{(k-1)}} u_S^{(k)} v(S) + & \sum\limits_{\{i\}\in\Gamma^{(k-1)}} w_{\{i\}}^{(k)} v(\{i\}) + & \sum\limits_{S\in\Delta_r^{(k-1)}} z_S^{(k)}(v(S) - y_S^*) & \\ \text{s.t.} & \sum\limits_{S\in\Sigma^{(k-1)}} u_S^{(k)} e(S)_i + & w_{\{i\}}^{(k)} \widetilde{e}(\Gamma^{(k-1)})_i + & \sum\limits_{S\in\Delta_r^{(k-1)}} z_S^{(k)} e(S)_i = 0 & \forall i \in N \\ & \sum\limits_{S\in\Sigma^{(k-1)}} u_S^{(k)} & = 1 & & \\ & u^{(k)}, & w^{(k)} & \geq 0 & \end{array} \qquad (\mathbf{D^{(k)}})$$

where $u^{(k)}$ and $w^{(k)}$ are non-negative dual variables corresponding to the unsettled inequalities for $\Sigma^{(k-1)}$ and $\Gamma^{(k-1)}$ respectively, $z^{(k)}$ are dual variables for the representative set of settled equalities for $\Delta_r^{(k)}$, $\widetilde{e}(\Gamma^{(k-1)}) \in \{0,1\}^n$ such that $\widetilde{e}(\Gamma^{(k-1)})_i = 1$ if and only if $\{i\} \in \Gamma^{(k-1)}$, $y_N^* = 0$ and $y_T^* = \sum_{S\in\Sigma^{(l-1)}} u_S^{(l)} v(S) + \sum_{\{i\}\in\Gamma^{(l-1)}} w_{\{i\}}^{(l)} v(\{i\}) + \sum_{S\in\Delta_r^{(l-1)}} z_S^{(l)}(v(S) - y_S^*)$ if for some $1 \leq l \leq k-1 : T \in \mathcal{T}^{(l)*}$ and $(u^{(l)}, w^{(l)}, z^{(l)}) \in \mathcal{D}^{(l)*}$ ($\mathbf{D^{(l)}}$)-optimal.

Just like the linear programs, tight sets and the corresponding results can be extended for the dual too. However, instead of tight sets we are more interested in the complement,

the *non-tight sets* (or support sets) in the dual regarding the sign-restriction inequalities $u_S^{(k)} \geq 0$ for $S \in \Sigma^{(k-1)}$ and $w_{\{i\}}^{(k)} \geq 0$ for $\{i\} \in \Gamma^{(k-1)}$. Similarly to above let us denote with $\mathcal{D}^{(k)}$ and $\mathcal{D}^{(k)*}$ the feasible and optimal solutions of ($\mathbf{D^{(k)}}$), as well as $\mathcal{NT}^{(k)}(u) = \left\{S \in \Sigma^{(k-1)} : u_S > 0\right\}$ and $\widetilde{\mathcal{NT}}^{(k)}(w) = \left\{\{i\} \in \Gamma^{(k-1)} : w_{\{i\}} > 0\right\}$ for any $(u, w, z) \in \mathcal{D}^{(k)}$. Applying the complement of the minimal tight set to ($\mathbf{D^{(k)}}$) we find that there exists a unique *maximal non-tight set* in the dual, denoted by $\mathcal{NT}^{(k)*}$ and $\widetilde{\mathcal{NT}}^{(k)*}$ regarding inequalities $u^{(k)} \geq 0$ and $w^{(k)} \geq 0$ respectively. Furthermore, Lemma 2.3 tells us that $\mathcal{NT}^{(k)}(u^{(k)}) \subseteq \mathcal{NT}^{(k)*}$ (as well as $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) \subseteq \widetilde{\mathcal{NT}}^{(k)*}$) for all dual solutions $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$ and the union of the non-tight sets yields the maximal non-tight set. The following proposition ties together the primal-dual results.

Proposition 2.5. *The minimal tight set in the primal* ($\mathbf{P^{(k)}}$) *coincide with the maximal non-tight set in the dual* ($\mathbf{D^{(k)}}$)*, that is* $\mathcal{NT}^{(k)*} = \mathcal{T}^{(k)*}$ *and* $\widetilde{\mathcal{NT}}^{(k)*} = \widetilde{\mathcal{T}}^{(k)*}$.

Proof. We are going to show $\mathcal{NT}^{(k)*} = \mathcal{T}^{(k)*}$, as proving $\widetilde{\mathcal{NT}}^{(k)*} = \widetilde{\mathcal{T}}^{(k)*}$ can be done using the same technique. Fix an arbitrary $S \in \Sigma^{(k-1)}$. First we are going to show that $\mathcal{NT}^{(k)*} \subseteq \mathcal{T}^{(k)*}$. We claim that if $u_S^{(k)} > 0$ for some $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$, then $\epsilon^{(k)*} + x^{(k)}(S) = v(S)$ for all $x^{(k)} \in \mathcal{P}^{(k)*}$, that is, $S \in \mathcal{T}^{(k)*}$. Suppose the opposite, that $\epsilon^{(k)*} + x^{(k)}(S) > v(S)$ for some $x^{(k)} \in \mathcal{P}^{(k)*}$. Then $u_S^{(k)}\left(\epsilon^{(k)*} + x^{(k)}(S)\right) > u_S^{(k)} v(S)$, and because of primal-dual feasibility $u_T^{(k)}\left(\epsilon^{(k)*} + x^{(k)}(T)\right) \geq u_T^{(k)} v(T)$ for all $T \in \Sigma^{(k-1)}$ and $T \neq S$. Combining these we get

$$\sum_{T \in \Sigma^{(k-1)}} u_T^{(k)} \left(\epsilon^{(k)*} + x^{(k)}(T)\right) > \sum_{T \in \Sigma^{(k-1)}} u_T^{(k)} v(T) \tag{2.1}$$

Multiplying dual equality constraints by $x_i^{(k)}$ and $\epsilon^{(k)*}$ respectively, we get

$$\begin{aligned} \sum_{T \in \Sigma^{(k-1)}} u_T^{(k)} e(T)_i x_i^{(k)} + w_{\{i\}}^{(k)} \widetilde{e}(\Gamma^{(k-1)})_i x_i^{(k)} + \sum_{S \in \Delta_r^{(k-1)}} z_S^{(k)} e(S)_i x_i^{(k)} &= 0 \quad \forall i \in N \\ \sum_{T \in \Sigma^{(k-1)}} u_T^{(k)} \epsilon^{(k)*} &= \epsilon^{(k)*} \end{aligned}$$

Summing up all these equalities we arrive at

$$\sum_{T \in \Sigma^{(k-1)}} u_T^{(k)} \left(\epsilon^{(k)*} + x^{(k)}(T)\right) + \sum_{\{i\} \in \Gamma^{(k-1)}} w_{\{i\}}^{(k)} x_i^{(k)} + \sum_{S \in \Delta_r^{(k-1)}} z_S^{(k)} x^{(k)}(S) = \epsilon^{(k)*} \tag{2.2}$$

Because of (2.1), the left-hand side of (2.2) is strictly greater than

$$\sum_{T \in \Sigma^{(k-1)}} u_T^{(k)} v(T) + \sum_{\{i\} \in \Gamma^{(k-1)}} w_{\{i\}}^{(k)} x_i^{(k)} + \sum_{S \in \Delta_r^{(k-1)}} z_S^{(k)} x^{(k)}(S),$$

but because of primal feasibility this is at least

$$\sum_{T\in\Sigma^{(k-1)}} u_T^{(k)} v(T) + \sum_{\{i\}\in\Gamma^{(k-1)}} w_{\{i\}}^{(k)} v(\{i\}) + \sum_{S\in\Delta_r^{(k-1)}} z_S^{(k)}(v(S) - y_S^*).$$

Hence we contradict strong duality and so $\mathcal{NT}^{(k)*} \subseteq \mathcal{T}^{(k)*}$.

For $\mathcal{T}^{(k)*} \subseteq \mathcal{NT}^{(k)*}$, we show that the dual minimal tight set $\Sigma^{(k-1)}\backslash\mathcal{NT}^{(k)*}$ is contained in the primal maximal non-tight set $\Sigma^{(k-1)} \setminus \mathcal{T}^{(k)*}$. For that we are going to use the result from Goldman and Tucker (1956) (based on Tucker (1956)), namely that every LP has a strict complementary solution. If $S \in \Sigma^{(k-1)} \setminus \mathcal{NT}^{(k)*}$, then $u_S^{(k)} = 0$ for all $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$ by Lemma 2.3, and thus there must be some $x^{(k)} \in \mathcal{P}^{(k)*}$ with $\epsilon^{(k)*} + x^{(k)}(S) > v(S)$, otherwise there would be no strict complementary pair $(u^{(k)}, w^{(k)}, z^{(k)})$ and $(x^{(k)}, \epsilon^{(k)*})$. Again, by Lemma 2.3 this means that $S \in \Sigma^{(k-1)}\backslash\mathcal{T}^{(k)*}$, and the proof is complete. □

Corollary 2.6. *For any optimal pair* $x^{(k)} \in \mathcal{P}^{(k)*}$ *and* $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$

$$\mathcal{NT}^{(k)}(u^{(k)}) \subseteq \mathcal{NT}^{(k)*} = \mathcal{T}^{(k)*} \subseteq \mathcal{T}^{(k)}(x^{(k)}),$$
$$\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) \subseteq \widetilde{\mathcal{NT}}^{(k)*} = \widetilde{\mathcal{T}}^{(k)*} \subseteq \widetilde{\mathcal{T}}^{(k)}(x^{(k)})$$

Apart from the differences in line with the primal case, that is having less and unrestricted dual variables z, the sequence of (dual) LPs ($\mathbf{D^{(k)}}$) are Solymosi (1993)'s dual LP sequence. As expected the dual LPs also find the nucleolus solution in at most $n-1$ iterations. Note however an important consequence of Corollary 2.6.

Remark 2.7. Suppose an algorithm computing the nucleolus using the sequence of dual LPs ($\mathbf{D^{(k)}}$) update $\Delta_r^{(k)}$ with $\mathcal{NT}^{(k)}(u^{(k)})\cup\widetilde{\mathcal{NT}}^{(k)}(w^{(k)})$ for an arbitrary $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$, while another algorithm based on the sequence of primal LPs ($\mathbf{P^{(k)}}$) update $\Delta_r^{(k)}$ using $\mathcal{T}^{(k)*} \cup \widetilde{\mathcal{T}}^{(k)*}$. Then the former method requires at least the same number of iterations to find ν, than the latter, and it requires strictly more if

$$\operatorname{rank}(\Delta_r^{(k-1)} \cup \mathcal{NT}^{(k)}(u^{(k)}) \cup \widetilde{\mathcal{NT}}^{(k)}(w^{(k)})) < \operatorname{rank}(\Delta_r^{(k-1)} \cup \mathcal{T}^{(k)*} \cup \widetilde{\mathcal{T}}^{(k)*})$$

for any k.

While Table 1.1 considers worst case number of iterations, which is at most $n-1$ for state-of-the-art sequential LP methods, Remark 2.7 highlights the difference between the actual number of iterations needed for primal and dual methods. While primal methods require minimal number of iterations by nature (at the price of including an iterative subroutine in most cases), dual methods can require more iterations to compute the nucleolus. However, dual LPs ($\mathbf{D^{(k)}}$) are generally easier to solve than their primal counterparts ($\mathbf{P^{(k)}}$), mainly due to the alignment of the coefficient matrix causing dual

bases to be of compact size compared to primal bases. This is a well-known trade-off between traditional primal and dual sequential LP methods.[2]

Before turning to the problem of how to find the minimal tight set (e.g. introducing the dual subroutine used by primal methods), let us summarise the general framework of calculating the nucleolus in Algorithm 1.

Algorithm 1: Finding the nucleolus ν of cooperative game v with $I(v) \neq \emptyset$

1. Initialisation: $k = 1, \Delta_r^{(0)} = N, \Sigma^{(0)} = \mathcal{N}, \Gamma^{(0)} = \{\{i\} : i \in N\}, y_N^* = 0$;
while $\text{rank}(\Delta_r^{(k-1)}) < n$ **do**
  2. Solve ($\mathbf{P^{(k)}}$) (or ($\mathbf{D^{(k)}}$)), finding $(x^{(k)}, \epsilon^{(k)*}) \in \mathcal{P}^{(k)*}$ (or $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$);
  3. Find (the entire, or some subset of) the minimal tight set: $\mathcal{T} \subseteq \mathcal{T}^{(k)*}$, $\widetilde{\mathcal{T}} \subseteq \widetilde{\mathcal{T}}^{(k)*}$;
  4. Update $\Delta_r^{(k-1)}$: $\text{span}\left(\Delta_r^{(k)}\right) = \text{span}\left(\Delta_r^{(k-1)} \cup \mathcal{T} \cup \widetilde{\mathcal{T}}\right)$, $\text{rank}\left(\Delta_r^{(k)}\right) = |\Delta_r^{(k)}|$, $k = k + 1$;
  5. Find all $S \in \Sigma^{(k-1)} \setminus \text{span}(\Delta_r^{(k-1)})$, $\{i\} \in \Gamma^{(k-1)} \setminus \text{span}(\Delta_r^{(k-1)})$ to formulate ($\mathbf{P^{(k)}}$) (or ($\mathbf{D^{(k)}}$));
end
6. Output: ν is the unique solution of $x(S) = v(S) - y_S^* \; \forall S \in \Delta_r^{(k-1)}$;

So far we neglected how to tackle several tasks included in Algorithm 1. First of all, solving ($\mathbf{P^{(k)}}$) (or ($\mathbf{D^{(k)}}$)) in *Step 2* is already challenging; we return to this in Chapter 3. Once we solved the large LP, identifying elements of the minimal tight set $\mathcal{T}^{(k)*}$ in *Step 3* is the next problem, Section 2.3 reviews several possibilities for this. Finally, it is not straightforward how to find unsettled coalitions belonging to $\text{span}(\Delta_r^{(k)})$. *Steps 4* and *5* of the process remains mostly unaddressed in the literature. Section 2.4 explores some of the possible approaches, while we revisit this task from a new, conceptually different approach in Section 5.5.

2.3 Finding the minimal tight set

In this section we address the problem of how to find the entire, or some part of the minimal tight set $\mathcal{T}^{(k)*}$. First of all, Lemma 2.3 allows the practically often unusable method of finding all the optimal solutions and the corresponding tight sets in order to find the minimal tight set. Such brute force method in general does not get more attractive if we try to use it from a dual perspective: finding all the dual optimal solutions and taking the union of the corresponding non-tight sets can be just as hard as finding all the primal optimal solutions and the intersection of the tight sets.

[2] While Solymosi (1993) claims to favour the dual setting, based on numerical results (Tables 5.6-5.10) we find that it varies which setting is better. Furthermore, while this phenomenon is governed by another primal-dual trade-off that we examine in Section 5.5, in Section 4.2 we introduce sequential LP method variants that dissolve the former, classical trade-off.

Nevertheless, Corollary 2.6 allows us to find at least a part of the minimal tight set as $\mathcal{NT}^{(k)}(u^{(k)}) \subseteq \mathcal{T}^{(k)*}$ and $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) \subseteq \widetilde{\mathcal{T}}^{(k)*}$, given we find a dual optimal solution $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$. Essentially this is how dual methods find the nucleolus, with the evident downside of possibly needing to perform more iterations than primal methods (cf. Remark 2.7).

2.3.1 Improving subroutine of Solymosi (1993)

On the other hand Solymosi (1993) uses a (dual based) subroutine which can be incorporated in the primal LP sequence in order to find $\mathcal{T}^{(k)*} \cup \widetilde{\mathcal{T}}^{(k)*}$ from any $\mathcal{T}^{(k)}(x^{(k)}) \cup \widetilde{\mathcal{T}}^{(k)}(x^{(k)})$, given we found $x^{(k)} \in \mathcal{P}^{(k)*}$. In Solymosi (1993)'s dual sequence, after solving a similar LP to ($\mathbf{D^{(k)}}$) in *Step 2* of Algorithm 1 finding $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$, the update in *Step 4* is done by $\Delta^{(k)} = \Delta^{(k-1)} \cup \left\{S : u_S^{(k)} > 0\right\} \cup \left\{\{i\} : w_{\{i\}}^{(k)} > 0\right\}$. While in the primal sequence a similar LP to ($\mathbf{P^{(k)}}$) is solved finding solution $(x^{(k)}, \epsilon^{(k)*}) \in \mathcal{P}^{(k)*}$ with tight sets $\mathcal{T}^{(k)}(x^{(k)})$ and $\widetilde{\mathcal{T}}^{(k)}(x^{(k)})$. Then in *Step 3* the minimal tight set $\mathcal{T}^{(k)*} \subseteq \mathcal{T}^{(k)}(x^{(k)})$, $\widetilde{\mathcal{T}}^{(k)*} \subseteq \widetilde{\mathcal{T}}^{(k)}(x^{(k)})$ is found by repeatedly solving the following system of equations:

$$\begin{array}{llllll} \sum\limits_{S\in\mathcal{U}} & \lambda_S e(S)_i + & \gamma_{\{i\}}\widetilde{e}(\widetilde{\mathcal{U}})_i + \sum\limits_{S\in\Delta^{(k-1)}} & \mu_S e(S)_i & -\kappa & = 0 \ \forall i \in N \\ \sum\limits_{S\in\mathcal{U}} & \lambda_S & & & & = 1 \\ & \lambda, & \gamma, & & \mu, & \kappa \ \geq 0 \end{array} \tag{2.3}$$

where $\widetilde{e}(\widetilde{\mathcal{T}}) \in \{0,1\}^n$ such that $\widetilde{e}(\widetilde{\mathcal{T}})_i = 1 \iff \{i\} \in \widetilde{\mathcal{T}}$, initially $\mathcal{T}^{(k)}(x^{(k)}) = \mathcal{U}$, $\widetilde{\mathcal{T}}^{(k)}(x^{(k)}) = \widetilde{\mathcal{U}}$, and $\mathcal{T}, \widetilde{\mathcal{T}} = \emptyset$. While (2.3) is feasible we find a solution $(\lambda^*, \gamma^*, \mu^*)$ and we expand $\mathcal{T}$ and $\widetilde{\mathcal{T}}$ with coalitions

$$\begin{array}{c} \left\{S \in \mathcal{U} : S \in \text{span}\left(\{T \in \mathcal{U} : \lambda_T^* > 0\} \cup \Delta^{(k-1)}\right)\right\}, \\ \left\{\{i\} \in \widetilde{\mathcal{U}} : \{i\} \in \text{span}\left(\{\{j\} \in \widetilde{\mathcal{U}} : \gamma_{\{j\}}^* > 0\} \cup \Delta^{(k-1)}\right)\right\} \end{array} \tag{2.4}$$

then remove from $\mathcal{U}$ and $\widetilde{\mathcal{U}}$ and add to $\Delta^{(k-1)}$ the same set of coalitions (2.4). This way, by repeatedly solving (2.3), we find dual optimal solutions $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$ with larger and larger support $\left\{S \in \Sigma^{(k-1)} : u_S^{(k)} > 0\right\}$ and $\left\{\{i\} \in \Gamma^{(k-1)} : w_{\{i\}}^{(k)} > 0\right\}$.

Eventually we either find $\mathcal{T} = \emptyset$, $\widetilde{\mathcal{T}} = \emptyset$ verifying $\mathcal{T}^{(k)}(x^{(k)}) = \mathcal{T}^{(k)*}$, $\widetilde{\mathcal{T}}^{(k)}(x^{(k)}) = \widetilde{\mathcal{T}}^{(k)*}$, or (2.3) becomes infeasible, resulting in $\mathcal{T}^{(k)*} = \mathcal{T}^{(k)}(x^{(k)}) \setminus \mathcal{T}$, $\widetilde{\mathcal{T}}^{(k)*} = \widetilde{\mathcal{T}}^{(k)}(x^{(k)}) \setminus \widetilde{\mathcal{T}}$. In any case, we find $\mathcal{T}^{(k)*}$, $\widetilde{\mathcal{T}}^{(k)*}$ by solving (2.3) at most

$$\text{rank}\left(\mathcal{T}^{(k)}(x^{(k)}) \cup \widetilde{\mathcal{T}}^{(k)}(x^{(k)}) \cup \Delta^{(k-1)}\right) - \text{rank}\left(\Delta^{(k-1)}\right)$$

times, as the update (2.4) guarantees the rank increases every time we solve (2.3).

In (2.3) the number of variables is usually dominated by the size of the tight set $\mathcal{T}^{(k)}(x^{(k)})$, the number of equations is always $n+1$. As the above argument shows, one has to solve (2.3) at most $2(n-1)$ times over the whole process of calculating the nucleolus: at most $n-1$ rank increasing iterations and at most $n-1$ infeasible runs verifying we found the minimal tight set.

2.3.2 Improving subroutine of Nguyen and Thomas (2016)

Given an optimal solution $x^{(k)} \in \mathcal{P}^{(k)*}$ with tight set $\mathcal{T}^{(k)}(x^{(k)})$, we can exploit the fact that if $\mathcal{T}^{(k)}(x^{(k)})$ is not the minimal tight set, then there must be another optimal solution $y^{(k)} \in \mathcal{P}^{(k)*}$ with smaller tight set $\mathcal{T}^{(k)}(y^{(k)})$. In this manner Nguyen and Thomas (2016) introduces a problem that either yields an optimal solution of ($\mathbf{P^{(k)}}$) with smaller tight set than $\mathcal{T}^{(k)}(x^{(k)})$ or we can conclude that $\mathcal{T}^{(k)}(x^{(k)}) = \mathcal{T}^{(k)*}$.

Let $\sigma = \max\left\{E(S, x^{(k)}) : S \notin \left(\text{span}(\Delta^{(k)}) \cup \mathcal{T}^{(k)}(x^{(k)})\right)\right\}$ be the largest excess value amongst the remaining, unsettled and currently not tight coalitions and let $\delta = (\epsilon^{(k)*} - \sigma)/(n+1) > 0$. Then consider the following LP:

$$
\begin{aligned}
\max_{y^{(k)} \in I(v)} \quad & \sum_{S \in \mathcal{T}^{(k)}(x^{(k)})} y^{(k)}(S) & & & \\
\text{s.t.} \quad & y^{(k)}(S) & \geq x^{(k)}(S) \quad & \forall S \in \mathcal{T}^{(k)}(x^{(k)}) \\
& y^{(k)}(S) & = x^{(k)}(S) \quad & \forall S \in \Delta_r^{(k-1)} \\
& |(y^{(k)})_i - (x^{(k)})_i| & \leq \delta \quad & \forall i \in N
\end{aligned}
\tag{\textbf{FBOS}}
$$

First, we can see that by construction $x^{(k)}$ is a feasible solution of (**FBOS**). In fact, the first set of inequality constraints ensures that any feasible solution $y^{(k)}$ of (**FBOS**) is as good as $x^{(k)}$ for the currently tight unsettled coalitions in $\mathcal{T}^{(k)}(x^{(k)})$, the equality constraints ensure that $y^{(k)}$ satisfies the equality constraints of ($\mathbf{P^{(k)}}$) regarding settled coalitions, and a key element, the remaining inequality constraints restrict the feasible space close enough to $x^{(k)}$ such that the remaining unsettled constraints in ($\mathbf{P^{(k)}}$) are not violated. Therefore any feasible solution $y^{(k)}$ of (**FBOS**) remains feasible (and thus optimal) with $\epsilon^{(k)*}$ in ($\mathbf{P^{(k)}}$).

Further notice that if for any optimal solution $y^{(k)*}$ of (**FBOS**) any constraint of the first set of inequality constraints in (**FBOS**) is satisfied as a strict inequality (inactive), than $x^{(k)}$ is not an optimal solution of (**FBOS**) and we found a solution with smaller tight set. On the other hand if we find that all of those inequalities are active, and consequently

$$
\sum_{S \in \mathcal{T}^{(k)}(x^{(k)})} y^{(k)}(S) = \sum_{S \in \mathcal{T}^{(k)}(x^{(k)})} x^{(k)}(S),
$$

then we can conclude that we found an optimal solution $x^{(k)*}$ with the minimal tight set $\mathcal{T}^{(k)*}$. Although since there can be exponentially large tight sets, worst case we might have to solve (**FBOS**) exponentially many times before reaching this conclusion.

Unfortunately however, this approach can be numerically sensitive in certain cases. With maximising the objective function $\sum_{S\in\mathcal{T}^{(k)}(x^{(k)})} y_k(S)$, (**FBOS**) is tempted to remove any coalition in $\mathcal{T}^{(k)}(x^{(k)})$ from the set of tight constraints, including the ones in $\mathcal{T}^{(k)*}$ belonging to the minimal tight set. The only result prevents it doing so is Lemma 2.3, combined with the fact that every optimiser of (**FBOS**) is in $\mathcal{P}_k^*$. This brings us to the problem of how to correctly identify tight constraints? In practice we can not expect to definitely have $\epsilon^{(k)*} - E(S, x^{(k)}) = 0$, we have to consider calling coalition S tight if $\epsilon^{(k)*} - E(S, x^{(k)}) \approx 0$. That is, coalition S is very close to being tight, possibly within a very small threshold. In this case we are not only obviously limited by our computational precision, but one has to be very careful setting these tolerances for the most peculiar instances, as this can lead to numerical errors in using (**FBOS**) due to the small environment around $x^{(k)}$ we search in.

2.3.3 Interior point methods

The minimal tight set, or optimal partition is strongly connected to strict complementary solutions of LPs. Indeed, if we find a strict complementary pair $(x^{(k)}, \epsilon^{(k)*}) \in \mathcal{P}^{(k)*}, (u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$ then $\mathcal{T}^{(k)*} = \mathcal{T}^{(k)}(x^{(k)}) = \mathcal{NT}^{(k)}(u^{(k)})$. Solving an LP with an interior point method (IPM) will provide a strict complementary solution by construction[3], therefore we have to consider using it to solve ($\mathbf{P^{(k)}}$). Even though Bonnans and André (2008) or Elkind and Pasechnik (2009) acknowledge these facts, we are not aware of any attempts of designing an IPM solving ($\mathbf{P^{(k)}}$). The main reason could be the *traditional* preference of the simplex method, but additionally we highlight some practical difficulties in this subsection below.

In the general primal-dual path-following framework of IPMs (e.g. Gondzio (2012) provides an excellent overview of the topic), we start from a strictly feasible point of the LP and calculate the Newton direction to move along the central path. The Newton system of linear equations we have to solve for w in order to achieve this is in the form of $(ADA^\top)w = d$, where A is the coefficient matrix of the underlying LP, D is a diagonal matrix (see details below), w is the step we are going to take with the dual variable and d can be calculated from the parameters of the LP and the starting strictly feasible point.

Suppose we aim to solve ($\mathbf{P^{(k)}}$) by replacing the inequality constraints $x^{(k)}(S) + \epsilon^{(k)} \geq v(S)$ with subtracting $\mu \log(\epsilon(k) - E(S, x(k)))$ for some positive μ. Then for $k = 1$, A contains almost all possible n-coordinate binary vectors (except the all 0 and all 1)

[3]Without post-processing, or rounding of the solution.

as rows, plus the additional column of all ones. Therefore A has $n+1$ columns and 2^n-2 rows, so $AA^\top$ is a square matrix of dimension 2^n-2. Also, having a full column in A (belonging to the variable $\epsilon^{(1)}$) means that $AA^\top$ is completely dense. Rescaling with the diagonal matrix D having $D_{ii} = \frac{x_i^{(1)}}{\sum_{S\subsetneq N: i\in S} u_S^{(1)}}$ for $i \in N$ and $D_{(n+1)(n+1)} = \frac{\epsilon^{(1)}}{\sum_{S\subsetneq N} u_S^{(1)}}$ does not affect density. More precisely in this case $ADA^\top$ is a symmetric square matrix of dimension 2^n-2 with the element belonging to coalitions $S,T \subsetneq N$: $\epsilon^{(1)} + \sum_{i\in S\cap T} x_i^{(1)}/s_i^{(1)}$, where $s_i^{(1)} = \sum_{S\subsetneq N: i\in S} u_S^{(1)}$ for all $i \in N$. It is prohibitive to solve a Newton system of this size without a favourable sparsity structure. On the other hand, we can turn our attention to formalise the same system from a dual viewpoint.

The resulting different Newton system $(ADA^\top)w = d$ constructed from the dual is much more friendly in size. In this case $B = ADA^\top$ is a symmetric square matrix of dimension $n+1$ having $B_{ii} = \sum_{S\subsetneq N: i\in S} \frac{u_S^{(1)}}{\epsilon^{(1)} - E(S,x^{(1)})}$ for $i \in N$ in the diagonal and $B_{ij} = \sum_{S\subsetneq N: i,j\in S} \frac{u_S^{(1)}}{\epsilon^{(1)} - E(S,x^{(1)})}$ for $i,j \in N$ off the diagonal, with the additional row/column of $\left[(B_{ii})_{i\in N}, \sum_{S\subsetneq N} \frac{u_S^{(1)}}{\epsilon^{(1)} - E(S,x^{(1)})}\right]$, thus it is again completely dense. The additional row/column coincides exactly with the diagonal, and can be calculated from the lower triangle of the remaining part of the matrix. Therefore we only have to calculate $u_S^{(1)}/(\epsilon^{(1)} - E(S,x^{(1)}))$ for all the $S = \{i,j\}$ coalitions ($i = j$ included) requiring $n(n+1)/2$ divisions. Then however, from these values for each player we need to make $2^{n-1}-2$ additions to get the diagonal values of the matrix and finally $n-1$ additions to get the final element of B. This altogether adds up to $n(n+1)/2$ divisions and $n2^{n-1} - n - 1$ additions, so even though the size of the Newton system is favourable, constructing it can be problematic right from the start $k = 1$.

2.3.4 Finding strict complementary solutions

The reason we might consider interior point methods is that (without post-processing) they provide strict complementary solutions, which are in strong connection with the optimal partition of the constraint set: that is, partitioning the constraints into two disjoint sets, where one set includes all the constraints that are active among every optimal solution of the problem. Now we consider how to find a strict complementary solution of an LP without the necessity of using interior point methods. Freund et al. (1985) shows an approach how to deal with a similar problem, namely how to find which inequalities are always active in the set of linear inequalities $Ax \leq b$, no matter what feasible solution x we consider. We are interested in a somewhat more general problem, that is, finding the optimal partition of an LP with equality constraints, e.g. ($\mathbf{P^{(k)}}$). In the following we show how to modify the method of Freund et al. (1985) to achieve this. First suppose that we are able to find an arbitrary $x^{(k)} \in \mathcal{P}^{(k)*}$ optimiser of ($\mathbf{P^{(k)}}$). Then incorporating the equality constraint $\epsilon^{(k)} = \epsilon^{(k)*}$ we proceed with finding the always-active constraints in the constraint set $x^{(k)}(S) + \epsilon^{(k)*} \geq v(S)$ for all $S \in \Sigma^{(k)}$

and $x^{(k)}(S) = v(S) - y^*_S$ for all $S \in \Delta_r^{(k-1)}$. For this purpose, consider the following LP

$$
\begin{aligned}
\max_{x^{(k)},u^{(k)},w^{(k)},z^{(k)},\theta,\tilde{\theta}} \quad & \theta + \tilde{\theta} \\
\text{s.t.} \quad & x^{(k)}(S) = v(S) - y^*_S && \forall S \in \Delta_r^{(k-1)} && (FRT.1)\\
& x^{(k)}(S) \geq v(S) - \epsilon^{(k)*} && \forall S \in \Sigma^{(k-1)} && (FRT.2)\\
& x^{(k)}_i \geq v(\{i\}) && \forall \{i\} \in \Gamma^{(k-1)} && (FRT.3)\\
& \sum_{S\in\Sigma^{(k-1)}} u^{(k)}_S e(S)_i + w_{\{i\}}\tilde{e}(\Gamma^{(k-1)}) + \ldots \\
& \qquad \sum_{S\in\Delta_r^{(k-1)}} z^{(k)}_S e(S)_i = 0 && \forall i \in N && (FRT.4)\\
& \sum_{S\in\Sigma^{(k-1)}} u^{(k)}_S = 1 && && (FRT.5)\\
& u^{(k)}, w^{(k)} \geq 0 && && (FRT.6)\\
& \sum_{S\in\Sigma^{(k-1)}} u^{(k)}_S v(S) + \sum_{\{i\}\in\Gamma^{(k-1)}} w_{\{i\}} v(\{i\}) + \ldots \\
& \qquad \sum_{S\in\Delta_r^{(k-1)}} z^{(k)}_S (v(S) - y^*_S) = \epsilon^{(k)*} && && (FRT.7)\\
& u^{(k)}_S + x^{(k)}(S) - \theta \geq v(S) - \epsilon^{(k)*} && \forall S \in \Sigma^{(k-1)} && (FRT.8)\\
& w^{(k)}_{\{i\}} + x^{(k)}_i - \tilde{\theta} \geq v(\{i\}) && \forall \{i\} \in \Gamma^{(k-1)} && (FRT.9)
\end{aligned}
$$

It is clear, that ($\mathbf{P^{(k)}}$) and ($\mathbf{D^{(k)}}$) are both feasible and therefore problem (*FRT*) is also feasible and evidently bounded. For any feasible tuple $(x^{(k)}, u^{(k)}, w^{(k)}, z^{(k)}, \theta, \tilde{\theta})$ constraints (*FRT*.1)-(*FRT*.3) and (*FRT*.4)-(*FRT*.6) enforce primal optimal and dual feasible pair $x^{(k)}$ and $(u^{(k)}, w^{(k)}, z^{(k)})$, while constraint (*FRT*.7) ensures dual optimality of $(u^{(k)}, w^{(k)}, z^{(k)})$ through strong duality.

The key is constraints (*FRT*.8) and (*FRT*.9) enforcing strict complementarity. It simply maximizes the set $\left\{S : u^{(k)}_S > 0\right\}$ and $\left\{\{i\} : w^{(k)}_{\{i\}} > 0\right\}$ giving us $\mathcal{T}^{(k)*}$ and $\widetilde{\mathcal{T}}^{(k)*}$ through $\mathcal{NT}^{(k)*}$ and $\widetilde{\mathcal{NT}}^{(k)*}$. The biggest disadvantage is of course the size of the problem. In order to find a strict complementary point from any optimal pair $(x^{(k)}, \epsilon^{(k)*})$ and $(u^{(k)}, w^{(k)}, z^{(k)})$ we need to solve an LP with $\mathcal{O}(2^n)$ variables and $\mathcal{O}(2^n)$ constraints. This is obviously impossible when we consider the entire ($\mathbf{P^{(k)}}$), but can be an option to consider combined with a cut generation method.

2.4 Linear span of binary matrices

In this section we focus our attention on how to perform the update of $\Sigma^{(k)}$ in *Step 5* of Algorithm 1. In each iteration we need to find which unsettled coalitions belong to

$\text{span}(\Delta_r^{(k)})$, which is not a trivial task. There is a method presented in Nguyen and Thomas (2016) for the case when we are able to express v as a linear function of the characteristic vectors $e(S)$ of coalitions, e.g. the class of weighted voting games. The corresponding method treats this issue along with the cut generation problem, still, that might involve solving nearly $2(n - |\Delta_r^{(k)}|)$ mixed integer linear programs (MILPs) to identify one coalition outside of $\text{span}(\Delta_r^{(k-1)})$.

The problem leads us to investigating the linear span of a binary matrix. Stating the question with a more traditional notation, we are interested in deciding whether a given binary vector b lies in the linear span of a binary matrix A or not. Throughout this section let us assume that $A \in \{0,1\}^{m \times n}$ and $b \in \{0,1\}^n$, that is A is an m-by-n binary matrix and b is a binary row-vector consisting of n elements. Without loss of generality we can assume that $m < n$ and that $\text{rank}(A) = m$. The representative set of settled coalitions form such a matrix with the characteristic vectors $e(S)$ for $S \in \Delta_r^{(k)}$ being the rows of the matrix. We denote the i-th row of A with a^i, while the j-th coordinate of row vector, also as an element of A with $\left(a^i\right)_j = A_{ij}$. In this context b and a^i represent $e(S)$ for some coalition S in $\Sigma^{(k)}$ and elements of $\Delta_r^{(k)}$ respectively.

2.4.1 Necessary condition from matchings

First we are going to show a necessary condition for a binary vector b not to lie in the linear span of binary matrix A.

We start by showing a straightforward sufficient condition on $b \notin \text{span}(A)$. Since any linear combination of zeros can never yield a 1, if there is an $i \in \{1, \ldots, n\}$ coordinate such that $b_i = 1$, but $\left(a^j\right)_i = 0$ for all $j = 1, \ldots, m$, then b can not be in the linear span of A. Since $b_i \leq 1$ for all i and every coordinate of the sum of binary vectors is non-negative, we can state an easily verifiable condition as below.

$$\max_{1 \leq i \leq n} \left(b_i - \sum_{j=1}^{m} \left(a^j\right)_i \right) = 1 \Longrightarrow b \notin \text{span}(A)$$

However, since in our method of calculating the nucleolus we build up matrix A starting with the vector of all ones ($\Delta_r^{(0)} = N$), this condition is unfortunately meaningless in our case. That is, every vector b we are about to consider, will fail to meet the sufficient condition in our setting. Now we turn to show a necessary condition for $b \notin \text{span}(A)$ more meaningful in our case.

Definition 2.8. In a binary matrix A, let us call a pair of elements $A_{ij} = 1$ and $A_{kl} = 1$ *independent 1 elements* if $i \neq k$ and $j \neq l$, that is, if they are 1 elements of the matrix in different rows and columns.

We denote with $\text{ind}(A)$ the largest number of independent 1 elements that we can choose from A. Evidently $\text{ind}(A) \leq m$.

From any binary matrix $A \in \{0,1\}^{m\times n}$ we can construct an undirected bipartite graph $G = (U, V, E)$, such that vertex set $U = \{u_1, \ldots, u_m\}$ represents the rows while $V = \{v_1, \ldots, v_n\}$ represents the columns of A, and $(u_i, v_j) \in E$ is and edge in G if and only if $A_{ij} = 1$. A *matching* in a bipartite graph is a set of *independent edges*, that is any two edges of that set has different endpoints both in U and V. We know the following theorem about maximal matchings from Hall (1935).

Theorem 2.9 (Hall's theorem)**.** *In a bipartite graph $G = (U, V, E)$ there exists a matching entirely covering U if and only if for every subset $W \subseteq U : |W| \leq |N(W)|$, where $N(W) \subseteq V$ denotes the neighbouring vertices of W.*

Every matching in the induced bipartite graph corresponds to a set of independent 1 elements in the underlying matrix A. Using this we can show a connection between the rank and the maximal number of independent 1 elements.

Proposition 2.10. *If a binary matrix $A \in \{0,1\}^{m\times n}$ has rank of m, then* $\text{ind}(A) = m$.

Proof. The proof goes by induction. For $m = 1$ it is trivially true, since $\text{rank}(A) = 1$. Suppose that in an $(m-1)$ row binary matrix with rank $(m-1)$ we are able to find $(m-1)$ independent 1 elements. We are going to show indirectly that then it is possible to choose m independent 1 elements from A. We suppose the contrary in terms of the graph, that is, there exists $M \subseteq \{1, \ldots, m-1\}$ such that the number of non-zero elements in the vector $\sum_{j\in M} a^j$ is at least $|M|$, but the non-zeros of $\sum_{j\in M} a^j + a^m$ is less than $|M|+1$. The only way that this could happen if the number of non-zeros in $\sum_{j\in M} a^j$ is exactly $|M|$ and the corresponding coordinates cover the 1 coordinates of a^m. Let $\mathcal{A} = \{a^j : j \in M\}$ the collection of row-vectors and denote with $\mathcal{A}|_M$ the restriction of these vectors to M, that is, we simply delete the coordinates not in M. Since there are $|M|$ linearly independent binary vectors in $\mathcal{A}|_M$, it forms a basis of $\mathbb{R}^M$, and so they linearly span $a^m|_M$ with coefficients $(\alpha_j)_{j\in M}$, which vector we get from applying the same restriction to a^m. However, in coordinates outside of M, both a^m and every vector in $\mathcal{A}$ have only zeros, therefore vectors in $\mathcal{A}$ span a^m with the same coefficients having arbitrary values outside of M. This is contradicting the m rank of A. □

That means, in order for an additional binary row-vector $b \in \{0,1\}^n$ not to lie in the linear span of matrix A, adding that vector to the matrix has to increase the maximal number of independent 1 elements in the resulting matrix, implying a necessary condition for $b \notin \text{span}(A)$.

Figure 2.1 illustrates the intuition behind Proposition 2.10. Matrix A is represented as the bipartite graph consisting of the black nodes and edges, the addition of vector b as the red dashed node. If this addition not produces a red dotted edge in the graph, increasing the neighbourhood of the rows vertices U, then by Proposition 2.10 and Hall's Theorem 2.9, adding b to A cannot increase its rank, therefore $b \in \text{span}(A)$.

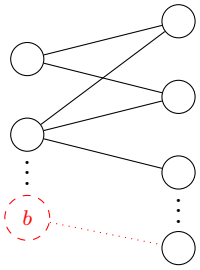


Figure 2.1: Bipartite graph illustrating Proposition 2.10

This is particularly useful for our setting in light of Theorem 2.9. When we evaluate this necessary condition, we do not have to check every subset of the rows of A by far, we just have to focus on subsets which are on the boundary of violating the condition in Hall's theorem by themselves, that is for subsets $M_l \subseteq \{1, \dots, m\}$ for which $|M_l| = |N(M_l)|$. In terms of the binary matrix, throughout the algorithm we can keep updated a list of $M_l \subseteq \{1, \dots, m\}$ subsets of the rows of A such that

$$|M_l| = |N(M_l)| := \sum_{1 \le i \le n} \min \left(\left(\sum_{j \in M_l} a^j \right)_i, 1 \right), \tag{2.6}$$

and we just have to check whether is there any such M_l that adding b to it would increase the right hand side of (2.6) or not. If not, we can conclude that $b \in \text{span}(A)$. Unfortunately if b does indeed increase the right hand side, we can not draw any conclusion from that.

Algorithm 2: Pseudocode for finding b not increasing rank(A) using Equation 2.6

1. Initialisation: Find $M \subsetneq 2^m : |M_l| = |N(M_l)|, \forall M_l \in M$ and set $\mathcal{B} \leftarrow \emptyset$;
for $i = 1, \dots, |M|$ **do**
  2. Find $\mathcal{J}_i \subseteq \{1, \dots, k\} \setminus \mathcal{B} : N(b^j) = N(M_i), \forall j \in \mathcal{J}_i$;
  3. Set $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{J}_i$;
end
4. Output: $\mathcal{B}$;

Algorithm 2 shows how we can make use of this necessary condition in our framework. It takes the input of $A \in \{0,1\}^{m \times n}$ and $B \in \{0,1\}^{h \times n}$, where the latter contains (as its rows) the $(b^j)_{j=1}^h$ vectors we would like to check with the necessary condition. The output $\mathcal{B} \subseteq \{1, \dots, k\}$ the algorithm produces contains row indices in B which for the constraint corresponding to the row can be eliminated from the problem, because it is

already implied by the equality constraints of A. Finding M in line 1 and subsequently every $\mathcal{J}_i$ in line 4 simply requires one matrix multiplication each.

2.4.2 Sufficient condition from the Reduced Row Echelon Form (*RREF*)

In this subsection we establish a sufficient condition for $b \notin \text{span}(A)$ from a different representation of $\text{span}(A)$. It is clear that we can perform certain row operations on A such that the resulting matrix $\tilde{A}$ yields the same row-space, that is $\text{span}(A) = \text{span}(\tilde{A})$. We are interested in two typical representations, in order to define them, let us call the first non-zero element of a vector *pivot element*.

Definition 2.11. A matrix $A \in \mathbb{R}^{m \times n}$ is in *row echelon form* if for every row a^i which has a pivot element

- if $i > 1$: the pivot element of a^i is to the right of the pivot element of a^{i-1},
- if $i < n$: the pivot element of a^i is to the left of the pivot element of a^{i+1},

and all rows without any pivot elements (if there is any) are in the bottom of the matrix.

Additionally, the matrix is in *reduced row echelon form* if

- it is in row echelon form,
- all of the pivot elements are ones and
- above the pivot elements (in the same column) there are only zeros.

Given A, it is possible to construct a matrix $\tilde{A}$ which is in reduced row echelon form and $\text{span}(A) = \text{span}(\tilde{A})$. This can be done by applying a series of row operations on A. Moreover, the resulting $\tilde{A}$ is unique. Definition 2.11 follows the traditional notions of these forms in the literature, but because of our special setting we are mainly interested in a form that is lying between the two introduced form. From A we are going to produce another matrix A_{RREF} which is close to being in reduced row echelon form, in the sense that it can only fail to meet the third criterion of having only zero elements above the pivots. Since that is not essential in our setting, we are going to relax that criterion.

Algorithm 3 shows how can we produce A_{RREF} from the binary matrix A. Recall that we assume $m < n, \text{rank}(A) = m$ and we take advantage of our setting yielding $a^1 = (1, \ldots, 1) \in \mathbb{R}^n$. Therefore by definition, the first row of A already meets our expectations. In light of this *Step 1* handles the first column under special consideration. After this we can start the loop for the rest of the matrix in *Step 2*. We repeat the process until all the rows or all the columns were handled, which means we transformed the entire A into A_{RREF}. Until then, we proceed column by column and row by row.

Algorithm 3: Pseudocode for calculating A_{RREF}

1. **for** $i = 2, \ldots, m$ **do**
  if $(a^i)_1 = 1$ **then**
    $a^i \leftarrow a^1 - a^i$;
  end
end
$i \leftarrow 2, j \leftarrow 2, J \leftarrow \emptyset$;
2. **if** $i > m$ ***or*** $j > n$ **then**
  Go To *Step 3*;
else
  $P \leftarrow \{p_1, \ldots, p_k\} \subseteq \{i, \ldots, m\} : (a^{p_l+i-1})_j \neq 0, \forall l \in \{1, \ldots, k\}$;
  if $P = \emptyset$ **then**
    $J \leftarrow J \cup \{j\}, j \leftarrow j + 1$, **Go To** *Step 2*;
  else
    $Q \leftarrow \{q_1, \ldots, q_h\} \subseteq \{1, \ldots, k\} : (a^{p_l+i-1})_j = 1, \forall l \in Q$;
    if $Q = \emptyset$ **then**
      $a^{p_1+i-1} \leftarrow \frac{1}{(a^{p_1+i-1})_j} a^{p_1+i-1}, Q \leftarrow \{1\}$;
    end
    if $p_{q_1} \neq 1$ **then**
      switch rows a^i and $a^{p_{q_1}+i-1}$;
      if $p_1 \neq 1$ **then**
        $P \leftarrow \{1\} \cup P, q_l \leftarrow q_l + 1, \forall l \in Q$;
      end
      $Q \leftarrow \{1\} \cup Q$;
      if $a^{p_{q_2}+i-1} = 0$ **then**
        $P \leftarrow P \setminus \{p_{q_2}\}$;
      end
      $Q \leftarrow Q \setminus \{q_2\}$;
    end
    if $|P| > 1$ **then**
      $a^{p_{q_l}+i-1} = a^{p_{q_l}+i-1} - (a^{p_{q_l}+i-1})_j a^i, \forall p_l \in P : l \neq q_1$;
    end
    $i \leftarrow i + 1, j \leftarrow j + 1$;
  end
end
3. Output: $A_{RREF} \leftarrow A, J \leftarrow J \cup (\{1, \ldots, n\} \setminus \{1, \ldots, m, m + 1, \ldots, m + |J|\})$;

First we identify the non-zero elements of the j-th column which are in the row i or below, we store this information in set P. If the set is empty, then we do not find a pivot element in this column and we can move to the next column. These columns without pivot elements have great significance, as we will see later on.

If we have non-zero elements to consider as pivots, we identify which of these are already 1 elements, and store those indices in Q. By doing so we minimise the number of multiplications we make over the process. If there is a pivot element of 1, we can exchange its row vector with the i-th row. Then we are finished with that row and what

is left is to clear the non-zero elements below the pivot. For that purpose we update sets P and Q accordingly. If there is no 1 element in P, we can make one at the A_{ij} position with a single division, and then the rest of the column is handled in the same manner.

Finally when all the columns or all the rows are treated, we can set the results in *Step 3*. Which brings us back to the importance of columns without pivot elements and why we consider storing those column indices in set J. It is straightforward to see that the row operations we use in Algorithm 3 does not change the row-space of matrix A and hence $\text{span}(A) = \text{span}(A_{RREF})$. Also, if $b \in \{0,1\}^n$ is such that the position of its first 1 element, that is, the pivot element of b is in J, then $b \notin \text{span}(A_{RREF})$, again for the same reason of linear combination of zeros can not yield one, and consequently $b \notin \text{span}(A)$.

2.4.3 Rank calculations using A_{RREF}

Having preserved the row-space of matrix A (based on $\Delta_r^{(k)}$) in our method of calculating the nucleolus, A_{RREF} not only proves useful tackling the representation problem through index set J. Some unsettled characteristic vectors are handled by either the necessary or the sufficient condition, for the rest of the vectors A_{RREF} can also help if we consider 'brute-force' rank calculations.

Algorithm 4: Pseudocode for deciding whether b increases rank(A)

1. Initialisation: $Q \leftarrow \{q_1, \ldots, q_{n-h}\} = \{1, \ldots, n\} \setminus J, j \leftarrow 0$;
while $j < n + 1$ **do**
  2. $P \leftarrow \{p_1, \ldots, p_k\} \subsetneq \{1, \ldots, n\} : b_{p_l} \neq 0, \forall l \in \{1, \ldots, k\}$;
  if $P = \emptyset$ **then**
    return Output: NO
  else
    if $p_1 \in J$ **then**
      return Output: YES
    else
      3. $b \leftarrow \frac{b_{p_1}}{(a^l)_j} a^l : q_l = p_1$;
      $j \leftarrow j + 1$;
    end
  end
end

Algorithm 4 works as follows. First we identify the possible pivot columns, where we might find a pivot element in b. Then we go into a loop until we checked every column. In the loop first we identify the non-zero elements of b. If it leads to an empty set, we finished and the conclusion is that adding b to A does not increase its rank. Otherwise we check whether the current pivot element of b is in J, and if it is, we can conclude that b would increase rank(A). Otherwise we change the pivot element of b by eliminating it using some row of A_{RREF} and increase the counter for the checked columns.

Having established a necessary and a sufficient condition, and it's expanded use in the rank calculations, in Section 5.5 we revisit the treatment of the constraint set $S \notin \text{span}(\Delta_r^{(k)})$ in a conceptually different, new approach.

Chapter 3

A new algorithm computing the nucleolus

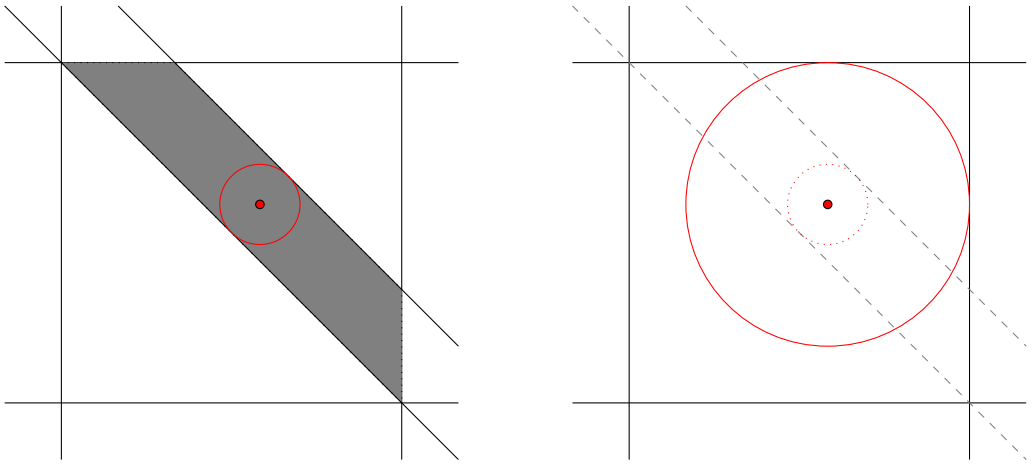
After addressing *Steps 3* and *4-5* of Algorithm 1 in Sections 2.3 and 2.4 respectively, we turn to the biggest challenge of *Step 2*. In this chapter we present our main results, a new method for calculating the nucleolus and prove its correctness. Ever since Kohlberg (1971) we know that *balancedness* is an important concept in terms of the nucleolus. Likewise, Solymosi (1993) considers certain kind of balancedness of some set of tight constraints. We base our method on balancedness and duality, and find that we are able to solve ($\mathbf{P^{(k)}}$) without explicitly formulating the large LP, rather focusing on small linear systems and LPs. In terms of general cooperative games, the only result we are aware of in this aspect is the unpublished paper of Derks and Kuipers (1997). As opposed to their purely dual approach, the primal-dual method we propose requires performing minimal number of iterations as well, a novelty in the literature and yields efficiency both in terms of CPU time and required memory.

3.1 Simplex-like algorithm

3.1.1 Geometric view of the nucleolus

We start by demonstrating the geometric view of the nucleolus on a simple three-player cooperative game.

Example 3.1. *Consider the 3-player game v with coalition values $v(\{1\}) = 1$, $v(\{2\}) = 2$, $v(\{3\}) = 5$, $v(\{1,2\}) = 6$, $v(\{1,3\}) = 7$, $v(\{2,3\}) = 8$, and $v(N) = 12$. The set of all imputations is $I(v) = \{(x_1, x_2, x_3) \ : \ x_1 + x_2 + x_3 = 12, \ x_1 \geq 1, \ x_2 \geq 2, \ x_3 \geq 5\}$, while*



(a) Starting level (b) Subsequent level

Figure 3.1: Geometric view of the nucleolus

the core of the game is

$$C(v) = \{(x_1, x_2, x_3) : \quad x_1 + x_2 + x_3 = 12,\ x_1 \geq 1,\ x_2 \geq 2,\ x_3 \geq 5,$$
$$x_1 + x_2 \geq 6,\ x_1 + x_3 \geq 7,\ x_2 + x_3 \geq 8\}.$$

The least core is the line segment connecting $x = (2, 4.5, 5.5)$ *and* $y = (3.5, 3, 5.5)$. *The nucleolus is* $\nu = (2.75, 3.75, 5.5)$.

Following the footsteps of Maschler et al. (1979), one can formulate the following geometric property of the nucleolus. For the purpose of convenient visualisation, let us consider the game in Example 3.1, which has a non-empty core. From any point x within the core of the game, the distances to the hyperplanes forming the core are the levels of satisfactions the corresponding coalitions have over the proposed solution x. The least core are those points in the core that minimise the maximum level of dissatisfaction, or equivalently, maximize the minimum level of satisfaction. The least core, therefore, contains the centre of the largest balls that we can fit into the core. The corresponding largest radius determines the minimum level of satisfaction.

This geometrical property is captured on Figure 3.1. In Figure 3.1a, the grey trapezoid is the core, and we find that we cannot increase the ball fitted into the core around the point of the nucleolus (or any other solution in the least core) because of the two parallel tight constraints (corresponding to coalitions S and $N \setminus S$). The tight constraints that avoid the ball from getting larger are known as a *balanced* collection of coalitions, a concept to be formally defined in the following Subsection 3.1.2.

Among the least core solutions, we want to minimise the second level of maximal dissatisfaction. This is equivalent to finding the largest ball within the new (and larger) polytope that is derived from the core after having removed the tight constraints found

in the previous step. If we repeat this process, we will finally arrive at the unique point that lexicographically minimises the dissatisfactions among all the coalitions, that is, the nucleolus.

In the example, if we relax the tight core constraints that are limiting the size of the ball in the first iteration, then we arrive at a new polytope as shown in Figure 3.1b. At this point, if we consider only the center points of the largest balls that fit into the least core, the center of the largest ball in the new, enlarged polytope is the nucleolus. The problem of finding the largest ball within a polytope can be formulated as a linear program ($\mathbf{P^{(k)}}$).

3.1.2 Balancedness and dual feasibility

We turn to introducing some further necessary notions. A collection of coalitions $\mathcal{B} \subseteq 2^N$ is called $\mathcal{B}_0$-*balanced* if

$$\begin{array}{llll} \sum\limits_{S\in\mathcal{B}} \lambda_S e(S) + \sum\limits_{T\in\mathcal{B}_0} \gamma_T e(T) & = e(N) & & \\ \lambda_S & > 0 & \forall S \in \mathcal{B} & \\ \gamma_T & \geq 0 & \forall T \in \mathcal{B}_0 & \end{array} \tag{3.1}$$

is feasible. We can assume $\mathcal{B} \cap \mathcal{B}_0 = \emptyset$ without loss of generality. That is, if λ and γ satisfies (3.1) with $S \in \mathcal{B} \cap \mathcal{B}_0$, then let $\tilde{\lambda}_S = \lambda_S + \gamma_S > 0$, $\tilde{\lambda}_T = \lambda_T$ for all $S \neq T \in \mathcal{B}$ and $\tilde{\gamma}_T = \gamma_T$ for all $S \neq T \in \mathcal{B}_0$ showing that $\mathcal{B}$ is $(\mathcal{B}_0 \setminus S)$-balanced with $\tilde{\lambda}$ and $\tilde{\gamma}$. Furthermore, $\mathcal{B}$ is $(\mathcal{B}_0 \cup S)$-balanced for all $S \in \mathcal{B}$ (cf. Remark 3.1 below).

The collection $\mathcal{B}$ being $\mathcal{B}_0$-balanced essentially means that coalitions in $\mathcal{B} \cup \mathcal{B}_0$ linearly span the grand coalition N with non-negative weights that are strictly positive for $\mathcal{B}$. Coefficients λ and γ in (3.1) are called *balancing weights*. This is a classical notion in cooperative game theory and appears in relation to the nucleolus in Kohlberg (1971) already.

We say that $\mathcal{B}$ contains a $\mathcal{B}_0$-balanced subcollection $\mathcal{A} \subsetneq \mathcal{B}$ if there exists $\lambda, \gamma \geq 0$ *almost* satisfying (3.1): non-negative balancing weights satisfying the equation in (3.1) with $\lambda_S > 0$ if $S \in \mathcal{A}$ and $\lambda_S = 0$ if $S \in \mathcal{B} \setminus \mathcal{A}$. We further introduce that collection of coalitions $\mathcal{B}$ is $\mathcal{B}_0$-*balanced with respect to* collection of coalitions $\mathcal{E}$ if the following system is feasible:

$$\begin{array}{llll} \sum\limits_{S\in\mathcal{B}} \lambda_S e(S) + \sum\limits_{T\in\mathcal{B}_0} \gamma_T e(T) + \sum\limits_{P\in\mathcal{E}} \mu_P e(P) & = e(N) & & \\ \lambda_S & > 0 & \forall S \in \mathcal{B} & \\ \gamma_T & \geq 0 & \forall T \in \mathcal{B}_0 & \end{array} \tag{3.2}$$

Notice that in this definition we only require strict positivity of balancing weights for coalitions in $\mathcal{B}$; balancing weights of coalitions in $\mathcal{E}$ can be arbitrary, might as well be negative.

Remark 3.1. Obviously if $\mathcal{B}$ is $\mathcal{B}_0$-balanced (with respect to $\mathcal{E}$), than it is also $\overline{\mathcal{B}}_0$-balanced (with respect to $\overline{\mathcal{E}}$) for any $\mathcal{B}_0 \subseteq \overline{\mathcal{B}}_0$ (and $\mathcal{E} \subseteq \overline{\mathcal{E}}$).

For the sake of brevity, whenever it is clear from context, we drop the $\mathcal{B}_0$ in front of balancedness. In the following Lemma 3.2 we argue that, in certain situations, we *could* drop the with respect to $\mathcal{E}$ part as well.

Lemma 3.2. *The following statements hold:*

(a) The collection $\mathcal{B}$ is $\mathcal{B}_0$-balanced if and only if there exists $\lambda \in \mathbb{R}^{|\mathcal{B}|}_{>0}, \gamma \in \mathbb{R}^{|\mathcal{B}_0|}_{\geq 0}, \mu \in \mathbb{R}$ such that

$$\sum_{S\in\mathcal{B}} \lambda_S e(S) + \sum_{T\in\mathcal{B}_0} \gamma_T e(T) + \mu e(N) = e(N). \tag{3.3}$$

(b) Suppose $\mathcal{B}$ contains a $\mathcal{B}_0$-balanced subcollection $\mathcal{A}$. Then $\mathcal{B}$ is $\mathcal{B}_0$-balanced if and only if there exists $\lambda \in \mathbb{R}^{|\mathcal{B}\setminus\mathcal{A}|}_{>0}, \gamma \in \mathbb{R}^{|\mathcal{B}_0|}_{\geq 0}, \mu \in \mathbb{R}^{|\mathcal{A}|}$ such that

$$\sum_{S\in\mathcal{B}\setminus\mathcal{A}} \lambda_S e(S) + \sum_{T\in\mathcal{B}_0} \gamma_T e(T) + \sum_{P\in\mathcal{A}} \mu_P e(P) = e(N), \tag{3.4}$$

that is, $\mathcal{B}\setminus\mathcal{A}$ is $\mathcal{B}_0$-balanced with respect to $\mathcal{A}$.

Proof. Part (a): if $\mathcal{B}$ is $\mathcal{B}_0$-balanced then obviously (3.3) is feasible with $\mu = 0$. Now suppose that (3.3) is satisfied with some weights $\lambda \in \mathbb{R}^{|\mathcal{B}|}_{>0}, \gamma \in \mathbb{R}^{|\mathcal{B}_0|}_{\geq 0}$, and $\mu \in \mathbb{R}$. First observe that $\mu < 1$ since $\lambda > 0$ and $\gamma \geq 0$. Thus,

$$\sum_{S\in\mathcal{B}} \frac{\lambda_S}{1-\mu} e(S) + \sum_{T\in\mathcal{B}_0} \frac{\gamma_T}{1-\mu} e(T) = e(N),$$

and hence $\mathcal{B}$ is $\mathcal{B}_0$-balanced.
Part (b): if $\mathcal{B}$ is $\mathcal{B}_0$-balanced then obviously (3.4) is feasible with $\mu > 0$. Now suppose $\exists \lambda \in \mathbb{R}^{|\mathcal{B}\setminus\mathcal{A}|}_{>0}, \gamma \in \mathbb{R}^{|\mathcal{B}_0|}_{\geq 0}$, and $\mu \in \mathbb{R}^{|\mathcal{A}|}$ satisfying (3.4). Since $\mathcal{A}$ is $\mathcal{B}_0$-balanced,

$$\sum_{S\in\mathcal{A}} \beta_S e(S) + \sum_{T\in\mathcal{B}_0} \alpha_T e(T) = e(N). \tag{3.5}$$

with $\beta > 0$ and $\alpha \geq 0$. Choose $\delta > 0$ large enough such that $\mu_S + \delta\beta_S > 0$ for all $S \in \mathcal{A}$. Then (3.4) and (3.5) lead to

$$\sum_{T\in\mathcal{B}_0} \frac{\gamma_T + \delta\alpha_T}{\delta+1} e(T) + \sum_{S\in\mathcal{B}\setminus\mathcal{A}} \frac{\lambda_S}{\delta+1} e(S) + \sum_{P\in\mathcal{A}} \frac{\mu_P + \delta\beta_P}{\delta+1} e(P) = e(N)$$

showing that $\mathcal{B}$ is indeed $\mathcal{B}_0$-balanced. □

Remark 3.3. Note that by Lemma 3.2, for every balanced collection $\mathcal{E}$, $\mathcal{B}$ is $\mathcal{B}_0$-balanced if and only if there exists $\lambda \in \mathbb{R}^{|\mathcal{B}|}_{>0}, \gamma \in \mathbb{R}^{|\mathcal{B}_0|}_{\geq 0}, \mu \in \mathbb{R}^{|\mathcal{E}|}$ such that

$$\sum_{S\in\mathcal{B}} \lambda_S e(S) + \sum_{T\in\mathcal{B}_0} \gamma_T e(T) + \sum_{P\in\mathcal{E}} \mu_P e(P) = 0,$$

that is, we can remove $e(N)$ from the right hand side.

For the purpose of the new balancedness notion recall the set of dual constraints in ($\mathbf{D^{(k)}}$).

$$\begin{aligned}
&\sum_{S\in\Sigma^{(k-1)}} u_S^{(k)} e(S)_i + \quad w_{\{i\}}^{(k)} \widetilde{e}(\Gamma^{(k-1)})_i + \sum_{S\in\Delta_r^{(k-1)}} z_S^{(k)} e(S)_i \quad = 0 \quad \forall i \in N \\
&\sum_{S\in\Sigma^{(k-1)}} u_S^{(k)} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad = 1 \\
&\qquad\qquad u^{(k)}, \qquad\qquad w^{(k)} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \geq 0
\end{aligned}$$

Note that for any feasible $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)}$, $\mathcal{NT}^{(k)}(u^{(k)})$ is $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)})$-balanced with respect to $\Delta_r^{(k-1)}$, since $N \in \Delta_r^{(k-1)}$ for all $k \geq 1$.

In the primal sequence of Solymosi (1993), (2.3) is introduced to find the minimal tight set $\mathcal{T}^{(k)*}$ within some tight set $\mathcal{T}^{(k)}(x^{(k)})$. However, in order to find a tight set containing the minimal tight set, finding solution $x^{(k)} \in \mathcal{P}^{(k)*}$ or finding $\mathcal{T}^{(k)}(x^{(k)})$ for some $x^{(k)} \in \mathcal{P}^{(k)*}$ is the challenging part. Instead we provide efficient means to solve large LPs like ($\mathbf{P^{(k)}}$).

Take any $x^{(k)} \in I(v)$ satisfying the current equality constraints, that is $x^{(k)}(S) = v(S) - y^*_S$ for all $S \in \Delta_r^{(k-1)}$. This is considerably easier, as for all such $x^{(k)}$ there exists an $\varepsilon^{(k)}$ such that the pair satisfies all unsettled inequalities. For that purpose find the current largest excess among the unsettled coalitions at this point

$$\varepsilon^{(k)}(x^{(k)}) = \max_{S\in\Sigma^{(k-1)}} E(S, x^{(k)})$$

and note that $(x^{(k)}, \varepsilon^{(k)}(x^{(k)})) \in \mathcal{P}^{(k)}$ is a feasible point in the k-th primal LP ($\mathbf{P^{(k)}}$). Hence we can find its tight set $\mathcal{T}^{(k)}(x^{(k)}, \varepsilon^{(k)}(x^{(k)}))$, $\widetilde{\mathcal{T}}^{(k)}(x^{(k)})$. For brevity, in the remaining part of this chapter, whenever the feasible point $(x^{(k)}, \varepsilon^{(k)}(x^{(k)}))$ is clear from context, we denote the tight set and imputation-tight set with $\mathcal{T}^{(k)}$ and $\widetilde{\mathcal{T}}^{(k)}$ respectively. Similarly, we shorten the notation of the representative set of settled coalitions to Δ_r from $\Delta_r^{(k-1)}$, whenever iteration k is fixed.

Now consider the relaxation of the dual constraints, involving only those unsettled dual variables u_S and $w_{\{i\}}$ that belong to tight coalitions in $\mathcal{T}^{(k)}$ and tight players (as singleton

coalitions) in $\widetilde{\mathcal{T}}^{(k)}$, along with all the settled dual variables μ for Δ_r.

$$\begin{array}{lll} \sum_{S\in\mathcal{T}^{(k)}} u_S^{(k)} e(S)_i + & w_{\{i\}}^{(k)} \widetilde{e}(\widetilde{\mathcal{T}}^{(k)})_i + \sum_{S\in\Delta_r} z_S^{(k)} e(S)_i & = 0 \quad \forall i \in N \\ \sum_{S\in\mathcal{T}^{(k)}} u_S^{(k)} & & = 1 \\ u^{(k)}, & w^{(k)} & \geq 0 \end{array}$$

Tight sets can grow very large (exponential size) in certain extreme cases, but in most instances they only include a manageable number of coalitions, thus the relaxation yields a tractable linear system of $n+1$ constraints and not too much variables. Therefore it is easy to solve and, supposing that the relaxation is feasible, we obtain a feasible point in the dual ($\mathbf{D^{(k)}}$) by setting $u_S = 0$ for all $S \in \Sigma^{(k-1)} \setminus \mathcal{T}^{(k)}$ and $w_{\{i\}} = 0$ for all $\{i\} \in \Gamma^{(k-1)} \setminus \widetilde{\mathcal{T}}^{(k)}$.

Recall that we are actually more interested to find $\mathcal{T}^{(k)*}$ and $\widetilde{\mathcal{T}}^{(k)*}$, rather than the possibly harder task of finding $(u^{(k)*}, w^{(k)*}, z^{(k)*}) \in \mathcal{D}^{(k)*}$ yielding $\mathcal{NT}^{(k)}(u^{(k)*}) = \mathcal{T}^{(k)*}$ and $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)*}) = \widetilde{\mathcal{T}}^{(k)*}$. Therefore, given a feasible dual relaxation on $\mathcal{T}^{(k)} \cup \widetilde{\mathcal{T}}^{(k)}$, it is of particular interest what is the largest non-tight set we can found within the primal tight sets $\mathcal{T}^{(k)} \cup \widetilde{\mathcal{T}}^{(k)}$ belonging to a primal feasible point.

Dual feasible point $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)}$ yields $\mathcal{NT}^{(k)}(u^{(k)})$ and $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)})$, but as a consequence of Lemma 6.1.2 from Peleg and Sudhölter (2007), we know that every (primal) tight coalition in the linear span of the (dual) non-tight coalitions $\mathcal{NT}^{(k)}$ is (dual) non-tight for some feasible point in the dual. Now we expand this to our notions of balancedness.

Lemma 3.4. *If collection of coalitions $\mathcal{B}$ is $\mathcal{B}_0$-balanced with respect to collection of coalitions $\mathcal{E}$, then* $\operatorname{span}(\mathcal{B}) \cap 2^N$ *is also $\mathcal{B}_0$-balanced with respect to $\mathcal{E}$.*

Proof. Since $\mathcal{B}$ is $\mathcal{B}_0$-balanced with respect to $\mathcal{E}$, there exists unrestricted μ, strict positive λ and non-negative γ such that

$$\sum_{S\in\mathcal{B}} \lambda_S e(S) + \sum_{T\in\mathcal{B}_0} \gamma_T e(T) + \sum_{P\in\mathcal{E}} \mu_P e(P) = e(N)$$

For any $S_0 \in \operatorname{span}(\mathcal{B}) \cap 2^N$ we have $e(S_0) = \sum_{S\in\mathcal{B}} \alpha_S e(S)$, thus for any δ

$$\begin{aligned} e(N) &= \sum_{S\in\mathcal{B}} \lambda_S e(S) + \sum_{T\in\mathcal{B}_0} \gamma_T e(T) + \sum_{P\in\mathcal{E}} \mu_P e(P) \\ &= \sum_{S\in\mathcal{B}} \lambda_S e(S) + \sum_{T\in\mathcal{B}_0} \gamma_T e(T) + \sum_{P\in\mathcal{E}} \mu_P e(P) + \delta \left(e(S_0) - \sum_{S\in\mathcal{B}} \alpha_S e(S) \right) \\ &= \sum_{S\in\mathcal{B}} (\lambda_S - \delta\alpha_S) e(S) + \sum_{T\in\mathcal{B}_0} \gamma_T e(T) + \sum_{P\in\mathcal{E}} \mu_P e(P) + \delta\epsilon(S_0). \end{aligned}$$

Hence there exists $\delta > 0$ such that $\lambda_S - \delta\alpha_S > 0$, showing that $\mathcal{B} \cup S_0$ is $\mathcal{B}_0$-balanced with respect to $\mathcal{E}$. Repeating the same finitely many times for every element in $(\text{span}(\mathcal{B}) \cap 2^N) \setminus \mathcal{B}$ shows that $\text{span}(\mathcal{B}) \cap 2^N$ is indeed $\mathcal{B}_0$-balanced with respect to $\mathcal{E}$. □

The set $\text{span}(\mathcal{NT}^{(k)}(u^{(k)}) \cap \mathcal{T}^{(k)}$ $(\text{span}(\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) \cap \widetilde{\mathcal{T}}^{(k)})$ is the largest (imputation-) non-tight set we can find using the dual feasible point $u^{(k)}$ $(w^{(k)})$ we currently have at our disposal. From this point we can proceed by transferring coalitions $\text{span}(\mathcal{NT}^{(k)}(\lambda)) \cap \mathcal{T}^{(k)}$ and $\text{span}(\widetilde{\mathcal{NT}}^{(k)}(\gamma)) \cap \widetilde{\mathcal{T}}^{(k)}$ from $\mathcal{T}^{(k)}$ and $\widetilde{\mathcal{T}}^{(k)}$ to Δ_r, and if there is any remaining, we can try to solve the relaxed dual system again. If feasible, it gives us a set of new dual non-tight coalitions, which then can be extended by its linear span due to Lemma 3.4. We can proceed with this until we found $\exists(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)} : \mathcal{NT}^{(k)}(u^{(k)}) = \mathcal{T}^{(k)}$, $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) = \widetilde{\mathcal{T}}^{(k)}$ or the relaxation becomes infeasible. In essence we are iteratively solving dual relaxations yielding feasible points in the dual with larger and larger non-tight (support) sets.

If the underlying (primal) feasible point $(x^{(k)}, \varepsilon^{(k)}(x^{(k)}))$ is optimal, this would not only mean we found the minimal tight set $\mathcal{T}^{(k)*}$, but basically get back Solymosi (1993)'s subroutine (2.3). Notice that this happens precisely when the dual relaxation restricted to the primal tight set is feasible, as in that case we have a primal-dual feasible pair $(x^{(k)}, \varepsilon^{(k)}(x^{(k)})) \in \mathcal{P}^{(k)}$ and $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)}$, hence both being not only feasible but optimal as well.

In the followings we show an improved version of the subroutine, tailored for the LP sequence ($\mathbf{P^{(k)}}$) we work with and one where we can use infeasible dual relaxations to actually find a (primal) optimal point from any feasible one, and thus subsequently the minimal tight set. Consider Algorithm 5 decomposing tight sets into $\mathcal{T}^{(k)} = \mathcal{T} \cup \mathcal{U}$ and $\widetilde{\mathcal{T}}^{(k)} = \widetilde{\mathcal{T}} \cup \widetilde{\mathcal{U}}$ essentially finding the largest $\widetilde{\mathcal{T}}$-balanced subcollection in $\mathcal{T}^{(k)}$ with respect to Δ_r: $\mathcal{T}$.

Lemma 3.5. *In every iteration of Algorithm 5 $\mathcal{T}$ is $\widetilde{\mathcal{T}}$-balanced with respect to $\Delta_r^{(k-1)}$.*

Proof. The proof goes by induction, that is, we show that $\mathcal{T}$ is $\widetilde{\mathcal{T}}$-balanced with respect to $\Delta_r^{(k-1)}$ in the beginning of every iteration: starting with the first, then assuming for the l-th and proving for the $(l+1)$-th iteration, provided the algorithm does not terminate in iteration l. First, obviously $\mathcal{T} = \emptyset$ is ($\widetilde{\mathcal{T}}$-)balanced with respect to any set containing N, such as $\Delta_r^{(k-1)}$. Thus let us suppose we start iteration l with $\mathcal{T}$ being $\widetilde{\mathcal{T}}$-balanced with respect to $\Delta_r^{(k-1)}$.

Suppose at the start of the $(l+1)$-th iteration we have $\mathcal{U} = \emptyset$. Then the algorithm terminates immediately and our claim holds by assumption. Therefore suppose $\mathcal{U} \neq \emptyset$. Furthermore, let us suppose that in iteration $(l+1)$ (3.6) is infeasible. Similarly, the algorithm terminates without changing $\mathcal{T}$ or $\widetilde{\mathcal{T}}$, and the claim holds true by assumption. Hence let us suppose that (3.6) is feasible.

Algorithm 5: Finding largest $\widetilde{\mathcal{T}}^{(k)}$-balanced subcollection in $\mathcal{T}^{(k)}$

1. Initialisation: $\mathcal{T}, \widetilde{\mathcal{T}} = \emptyset$, $\mathcal{U} = \mathcal{T}^{(k)}(x^{(k)})$, $\widetilde{\mathcal{U}} = \widetilde{\mathcal{T}}^{(k)}(x^{(k)})$, $\Delta = \Delta_r^{(k-1)}$;
while $|\mathcal{T}| < |\mathcal{T}^{(k)}(x^{(k)})|$ **do**
2. Find $(\lambda^*, \gamma^*, \mu^*)$ solving

$$\begin{array}{lllll} \max\limits_{\lambda,\gamma,\mu} & \sum\limits_{S\in\mathcal{U}} \lambda_S & + \sum\limits_{\{i\}\in\widetilde{\mathcal{U}}} \gamma_{\{i\}} & & \\ \text{s.t.} & \sum\limits_{S\in\mathcal{U}} \lambda_S e(S) & + \sum\limits_{\{i\}\in\widetilde{\mathcal{U}}} \gamma_{\{i\}} e(\{i\}) + \sum\limits_{S\in\Delta} \mu_S e(S) & = 0 & \\ & \sum\limits_{S\in\mathcal{U}} \lambda_S & & = 1 & \\ & \lambda, & \gamma & \geq 0 & \end{array} \tag{3.6}$$

if (3.6) *is infeasible* **then**
3. Stop and output: $\mathcal{T}, \mathcal{U}, \widetilde{\mathcal{T}}, \widetilde{\mathcal{U}}, \Delta$;
else
4. Set $\mathcal{T} = \text{span}\left(\{T \in \mathcal{U} : \lambda_T^* > 0\} \cup \Delta\right) \cap \mathcal{T}^{(k)}(x^{(k)})$, $\mathcal{U} = \mathcal{T}^{(k)}(x^{(k)}) \setminus \mathcal{T}$,
$\widetilde{\mathcal{T}} = \text{span}\left(\left\{\{i\} \in \widetilde{\mathcal{U}} : \gamma_{\{i\}}^* > 0\right\} \cup \Delta\right) \cap \widetilde{\mathcal{T}}^{(k)}(x^{(k)})$, $\widetilde{\mathcal{U}} = \widetilde{\mathcal{T}}^{(k)}(x^{(k)}) \setminus \widetilde{\mathcal{T}}$,
and expand Δ : $\text{span}\left(\Delta\right) = \text{span}\left(\Delta_r^{(k-1)} \cup \mathcal{T} \cup \widetilde{\mathcal{T}}\right)$, $|\Delta| = \text{rank}\left(\Delta\right)$;
end
end
5. Output: $\mathcal{T}^{(k)*} = \mathcal{T}$, $\mathcal{U} = \emptyset$, $\widetilde{\mathcal{T}}^{(k)*} = \widetilde{\mathcal{T}}$, $\widetilde{\mathcal{U}}$, Δ;

The LP (3.6) is evidently bounded as well, since $\sum_{S\in\mathcal{U}} \lambda_S = 1$ is fixed, and $\widetilde{\mathcal{U}} \cap \text{span}\left(\Delta\right) = \emptyset$ by construction. Hence we find optimal solution $(\lambda^*, \gamma^*, \mu^*)$ of (3.6). Let us define the following subsets of $\mathcal{U}$ and $\widetilde{\mathcal{U}}$ respectively:

$$\begin{aligned} \mathcal{L} &= \left\{S \in \mathcal{U} : \lambda_S^* > 0\right\}, \\ \mathcal{K} &= \left\{S \in \mathcal{U} \setminus \mathcal{L} : S \in \text{span}\left(\mathcal{L} \cup \Delta\right)\right\}, \\ \widetilde{\mathcal{L}} &= \left\{\{i\} \in \widetilde{\mathcal{U}} : \gamma_{\{i\}}^* > 0\right\}, \\ \widetilde{\mathcal{K}} &= \left\{\{i\} \in \widetilde{\mathcal{U}} \setminus \widetilde{\mathcal{L}} : \{i\} \in \text{span}\left(\widetilde{\mathcal{L}} \cup \Delta\right)\right\}. \end{aligned}$$

Then for all $P \in \mathcal{K}$ there exist ω^P and β^P such that

$$e(P) = \sum_{S\in\mathcal{L}} \omega_S^P e(S) + \sum_{S\in\Delta} \beta_S^P e(S).$$

As a result, $\delta\left(e(P) - \sum_{S\in\mathcal{L}} \omega_S^P e(S) + \sum_{S\in\Delta} \beta_S^P e(S)\right) = 0$ for any δ, while there exists $\delta^P > 0$ such that $\lambda_S^* - \delta^P \omega_S^P > 0$ for all $S \in \mathcal{L}$. Hence if we define $\overline{\lambda}$ and $\overline{\mu}$ to have $\overline{\lambda}_S = \lambda_S^* - \delta^P \omega_S^P$ for all $S \in \mathcal{L}$, while $\overline{\lambda}_P = \delta^P$ and $\overline{\mu}_S = \mu_S^* - \delta^P \beta_S^P$ for all $S \in \Delta$, we get

$$\sum_{S\in\mathcal{L}\cup\{P\}} \overline{\lambda}_S e(S) + \sum_{\{i\}\in\widetilde{\mathcal{L}}} \gamma_{\{i\}}^* e(\{i\}) + \sum_{S\in\Delta} \overline{\mu}_S e(S) = 0$$

with $\overline{\lambda}, \gamma^* > 0$, that is, $\mathcal{L} \cup \{P\}$ is $\widetilde{\mathcal{L}}$-balanced with respect to Δ. Let us repeat this process for every element of $\mathcal{K}$ with $\overline{\lambda}$ and $\overline{\mu}$ replacing λ^* and μ^*, until we get that the whole $\mathcal{L} \cup \mathcal{K}$ is $\widetilde{\mathcal{L}}$-balanced with respect to Δ.

As we noted in Remark 3.1, obviously $\mathcal{L} \cup \mathcal{K}$ is $(\widetilde{\mathcal{L}} \cup \widetilde{\mathcal{K}})$-balanced as well with respect to Δ, but repeating the previous process with the elements of $\widetilde{\mathcal{K}}$ instead of $\mathcal{K}$, we can even get $(\widetilde{\mathcal{L}} \cup \widetilde{\mathcal{K}})$-balancedness with $\overline{\gamma}_{\{i\}} > 0$ for all $\{i\} \in \widetilde{\mathcal{L}} \cup \widetilde{\mathcal{K}}$. Thus altogether we have $\overline{\lambda}, \overline{\gamma} > 0$ and $\overline{\mu}$ such that

$$\sum_{S \in \mathcal{L} \cup \mathcal{K}} \overline{\lambda}_S e(S) + \sum_{\{i\} \in \widetilde{\mathcal{L}} \cup \widetilde{\mathcal{K}}} \gamma^*_{\{i\}} e(\{i\}) + \sum_{S \in \Delta} \overline{\mu}_S e(S) = 0. \tag{3.7}$$

Additionally, by assumption we have $\lambda > 0$, $\gamma \geq 0$ and μ such that

$$\sum_{S \in \mathcal{T}} \lambda_S e(S) + \sum_{\{i\} \in \widetilde{\mathcal{T}}} \gamma_{\{i\}} e(\{i\}) + \sum_{S \in \Delta_r^{(k-1)}} \mu_S e(S) = 0 \tag{3.8}$$

However, note that by the construction of $\widetilde{\mathcal{T}}$ we can assume $\gamma > 0$ as well. Since Δ is essentially the representative set of $\Delta_r^{(k-1)} \cup \mathcal{T} \cup \widetilde{\mathcal{T}}$, all that is left when combining (3.7) and (3.8) to take care of the sign of balancing weights corresponding to coalitions in $\mathcal{T} \cap \Delta$ and $\widetilde{\mathcal{T}} \cap \Delta$. Note however, that there exists $\delta > 0$ such that $\lambda_S + \delta\overline{\mu}_S > 0$ and $\gamma_{\{i\}} + \delta\overline{\mu}_{\{i\}} > 0$ for all $S \in \mathcal{T} \cap \Delta$ and $\{i\} \in \widetilde{\mathcal{T}} \cap \Delta$.

Thus let us define $u_S = \lambda_S$ for all $S \in \mathcal{T} \setminus \Delta$, $u_S = \lambda_S + \delta\overline{\mu}_S$ for all $S \in \mathcal{T} \cap \Delta$, $u_S = \overline{\lambda}_S$ for all $S \in \mathcal{L} \cup \mathcal{K}$, $w_{\{i\}} = \gamma_{\{i\}}$ for all $\{i\} \in \widetilde{\mathcal{T}} \setminus \Delta$, $w_{\{i\}} = \gamma_{\{i\}} + \delta\overline{\mu}_{\{i\}}$ for all $\{i\} \in \widetilde{\mathcal{T}} \cap \Delta$, $w_{\{i\}} = \overline{\gamma}_{\{i\}}$ for all $\{i\} \in \widetilde{\mathcal{L}} \cup \widetilde{\mathcal{K}}$ and $z_S = \mu_S + \delta\overline{\mu}_S$ for all $S \in \Delta_r^{(k-1)}$, yielding

$$\sum_{S \in \mathcal{T} \cup \mathcal{L} \cup \mathcal{K}} u_S e(S) + \sum_{\{i\} \in \widetilde{\mathcal{T}} \cup \widetilde{\mathcal{L}} \cup \widetilde{\mathcal{K}}} w_{\{i\}} e(\{i\}) + \sum_{S \in \Delta_r^{(k-1)}} z_S e(S) = 0$$

with $u, w > 0$, and the proof is complete[1]. □

There are two possibilities when Algorithm 5 terminates: either $\mathcal{T} = \mathcal{T}^{(k)}$ (occurring if and only if $\mathcal{U} = \emptyset$), or otherwise (3.6) is infeasible and $\mathcal{U} \neq \emptyset$. In the former case $\mathcal{T}^{(k)}$ is balanced with respect to $\Delta_r^{(k-1)}$, while in the latter we not only know that it is not, the algorithm also provides a (sub)collection $\mathcal{U} \subseteq \mathcal{T}^{(k)}$ that causes the lack of balancedness.

3.1.3 Improving directions

In the followings we show how the decomposition of the tight set is related to solutions of the underlying LPs and the minimal tight set. Suppose Algorithm 5 terminates in

[1]Note that we proved the stronger claim of $\mathcal{T} \cup \widetilde{\mathcal{T}}$ is $\emptyset$-balanced with respect to $\Delta_r^{(k-1)}$ in every iteration of Algorithm 5, however, we believe Lemma 3.5 phrased in this form fits into the setting more.

Step 3., that is (3.6) is infeasible with $\mathcal{U} \neq \emptyset$. That happens precisely when

$$\begin{aligned} \sum_{S\in\mathcal{U}} \lambda_S e(S) \quad + \sum_{\{i\}\in\widetilde{\mathcal{U}}} \gamma_{\{i\}} e(\{i\}) + \sum_{S\in\Delta} \mu_S e(S) \quad &= 0 \\ \lambda, \qquad\qquad \gamma \qquad\qquad &\geq 0 \\ \lambda_S \qquad\qquad\qquad\qquad &> 0 \end{aligned}$$

is infeasible for all $S \in \mathcal{U}$. Then by Farkas' lemma there exists $d^S \in \mathbb{R}^n : d^S(S) > 0$, $d^S_i \geq 0$ for all $\{i\} \in \widetilde{\mathcal{U}}$ and $d^S(S) = 0$ for all $S \in \Delta$. It follows that there is a d, e.g. the sum or the average of the d^S vectors for $S \in \mathcal{U}$, satisfying all the constraints for all $S \in \mathcal{U}$. After normalisation we arrive at our definition of *improving directions*.

Definition 3.6. Given an output of Algorithm 5 $\mathcal{T}$, $\mathcal{U} \neq \emptyset$, $\widetilde{\mathcal{T}}$, $\widetilde{\mathcal{U}}$, Δ define the set of improving directions, denoted by D as

$$D := \{d \in \mathbb{R}^n : d(S) \geq 1 \ \forall S \in \mathcal{U}, d_i \geq 0 \ \forall \{i\} \in \widetilde{\mathcal{U}}, d(R) = 0 \ \forall R \in \Delta\}. \tag{3.9}$$

Note that an improving direction $d \in D$ is a payoff reallocation ($d(N) = 0$ as, $N \in \Delta$ for all k) that increases the payoff, and thereby the satisfaction of all coalitions having currently lowest satisfaction among coalitions not *supported by* a dual feasible solution, while making sure players on the boundary of individual rationality are not paying for this increase, and maintaining the payoff of coalitions already *supported by* a dual feasible solution.

Lemma 3.7. *If for some k the decomposition of tight set $\mathcal{T}^{(k)}$ into $\mathcal{T} \cup \mathcal{U}$ with Algorithm 5 yields $\mathcal{U} \neq \emptyset$ and $\mathcal{T} = \emptyset$, then $x^{(k)} \notin \mathcal{P}^{(k)*}$.*

Proof. Since $\mathcal{U} \neq \emptyset$ (3.6) is infeasible. Then by Farkas' lemma D is non-empty. It follows that

$$E(S, x^{(k)} + \alpha d) = E(S, x^{(k)}) - \alpha d(S) \leq \varepsilon^{(k)*} - \alpha$$

for all $S \in \mathcal{U}, d \in D$ and $\alpha \geq 0$. Since $\mathcal{T} = \emptyset$ we have $\mathcal{U} = \mathcal{T}^{(k)}$ and thus $\exists \alpha > 0$ small enough such that $E(S, x^{(k)} + \alpha d) \leq \varepsilon^{(k)*} - \alpha$ for all unsettled coalitions $S \in \Sigma^{(k-1)}$ too. Since satisfactions of settled coalitions are clearly maintained along d with $d(S) = 0 \ \forall S \in \Delta = \Delta_r^{(k-1)}$, we have $(x^{(k)} + \alpha d, \varepsilon^{(k)}(x^{(k)}) - \alpha) \in \mathcal{P}^{(k)}$, thus

$$\varepsilon^{(k)}(x^{(k)}) > \varepsilon^{(k)}(x^{(k)}) - \alpha \geq \varepsilon^{(k)}(x^{(k)} + \alpha d) \geq \max_{y\in\mathcal{P}^{(k+1)*}} \varepsilon^{(k)}(y) = \varepsilon^{(k)*},$$

with the latter equality coming from $\mathcal{P}^{(k+1)*} \subsetneq \mathcal{P}^{(k)*}$. □

The converse holds true in general, regardless of $\mathcal{T}$.

Lemma 3.8. *If $x^{(k)} \in \mathcal{P}^{(k)} \setminus \mathcal{P}^{(k)*}$ then the decomposition of tight set $\mathcal{T}^{(k)}$ into $\mathcal{T} \cup \mathcal{U}$ with Algorithm 5 yields $\mathcal{U} \neq \emptyset$.*

Proof. Since $x^{(k)} \notin \mathcal{P}^{(k)*}$ we have $\varepsilon^{(k)}(x^{(k)}) > \varepsilon^{(k)*}$. As noted above $\varepsilon^{(k)}(y) = \varepsilon^{(k)*}$ for any $y \in \mathcal{P}^{(k+1)*} \neq \emptyset$, and therefore $y \neq x^{(k)}$. Let

$$d = \frac{y - x^{(k)}}{\varepsilon^{(k)}(x^{(k)}) - \varepsilon^{(k)*}}.$$

It is easy to verify that d is in D. By Farkas' lemma it follows that (3.6) is infeasible and thus $\mathcal{U} \neq \emptyset$. □

Before drawing the conclusion from Lemmas 3.7 and 3.8 regarding the nucleolus, we introduce the *truncated ordered excess vectors* focusing on only certain coordinates of $\Theta(\overline{E}(x))$. Let $\overline{E}\big|_{\mathcal{M}}(x)$ be a restriction of $\overline{E}(x)$, listing the excess of coalitions $E(S,x)$ only for $S \in \mathcal{M} \subseteq \mathcal{N}$, and $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x)\right)$ remains the non-increasingly ordered vector of those excess values: $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x)\right)_1 \geq \Theta\left(\overline{E}\big|_{\mathcal{M}}(x)\right)_2 \cdots \geq \Theta\left(\overline{E}\big|_{\mathcal{M}}(x)\right)_{|\mathcal{M}|}$.

Corollary 3.9. *If for some k the decomposition of tight set $\mathcal{T}^{(k)}$ into $\mathcal{T} \cup \mathcal{U}$ with Algorithm 5 yields $\mathcal{U} \neq \emptyset$, then $x^{(k)} \neq \nu$.*

Proof. For $\mathcal{T} = \emptyset$ it is proven since $x^{(k)} \notin \mathcal{P}^{(k)*}$. Suppose $\mathcal{T} \neq \emptyset$. Since $\mathcal{U} \neq \emptyset$ the set of improving directions D defined in (3.9) is non-empty. Let $d \in D$ be arbitrary and $\mathcal{M} = \mathcal{N} \setminus \left(\text{span}(\Delta) \setminus \cup_{i=1}^{k-1}\mathcal{T}^{(i)*}\right)$, that is we disregard coalitions that are in span(Δ) but not (necessarily) have excess of at least $\epsilon^{(k-1)*}$. Then the first $|\cup_{i=1}^{k-1}\mathcal{T}^{(i)*}|$ coordinates in $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)})\right)$ belong to coalitions in $\cup_{i=1}^{k-1}\mathcal{T}^{(i)*}$ and the following $|\mathcal{T}^{(k)}|$ coordinates belong to $\mathcal{T}^{(k)}$. For sufficiently small $\alpha > 0$ the same applies to $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)} + \alpha d)\right)$. By the definition of d it is clear that $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)})\right)$ equals $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)} + \alpha d)\right)$ in the first $\cup_{i=1}^{k-1}\mathcal{T}^{(i)*}$ coordinates, while as we increase α from 0, $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)} + \alpha d)\right)$ is decreasing somewhere in the following $|\mathcal{T}^{(k)}|$ coordinates. Thus $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)} + \alpha d)\right) <_L \Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)})\right)$, and since $E(S, x^{(k)}) = E(S, x^{(k)} + \alpha d)$ for all the disregarded coalitions $S \in \text{span}(\Delta) \setminus \cup_{i=1}^{k-1}\mathcal{T}^{(i)*}$ by the definition of d, we have $\Theta(\overline{E}(x^{(k)} + \alpha d)) <_L \Theta(\overline{E}(x^{(k)}))$. As a result, while $x^{(k)} + \alpha d$ can be the nucleolus, $x^{(k)}$ certainly can not. □

Theorem 3.10. *Tight set $\mathcal{T}^{(k)}$ is the minimal tight set $\mathcal{T}^{(k)*}$ for some k if and only if its decomposition into $\mathcal{T} \cup \mathcal{U}$ with Algorithm 5 yields $\mathcal{U} = \emptyset$.*

Proof. Suppose that $\mathcal{T}^{(k)} = \mathcal{T}^{(k)*}$. Solymosi (1993) shows (in Theorem 2.10) that $\cup_{i=1}^{k}(\mathcal{T}^{(i)*} \cup \widetilde{\mathcal{T}}^{(i)*})$ form a balanced collections of coalitions, suppose that with balancing weights $\omega > 0$:

$$\sum_{S \in \cup_{i=1}^{k}\left(\mathcal{T}^{(i)*} \cup \widetilde{\mathcal{T}}^{(i)*}\right)} \omega_S e(S) = e(N).$$

Denote by $\mathcal{G}$ and $\widetilde{\mathcal{G}}$ the sets $\cup_{i=1}^{k-1}\mathcal{T}^{(i)*} \setminus \Delta_r^{(k-1)}$ and $\cup_{i=1}^{k-1}\widetilde{\mathcal{T}}^{(i)*} \setminus \Delta_r^{(k-1)}$. Then exists β^S for all $S \in \mathcal{G}$ and $\kappa^{\{i\}}$ for all $\{i\} \in \widetilde{\mathcal{G}}$ such that

$$e(S) = \sum_{T \in \Delta_r^{(k-1)}} \beta_T^S e(T)$$
$$e(\{i\}) = \sum_{T \in \Delta_r^{(k-1)}} \kappa_T^{\{i\}} e(T)$$

Thus define

$$\begin{aligned}
&\lambda_S^* = \tfrac{\omega_S}{\sum_{R \in \mathcal{T}^{(k)*}} \omega_R} && \forall S \in \mathcal{T}^{(k)*} \\
&\gamma_{\{i\}}^* = \tfrac{\omega_{\{i\}}}{\sum_{R \in \mathcal{T}^{(k)*}} \omega_R} && \forall \{i\} \in \widetilde{\mathcal{T}}^{(k)*} \\
&\gamma_{\{i\}}^* = 0 && \forall \{i\} \in \widetilde{\mathcal{U}} \setminus \widetilde{\mathcal{T}}^{(k)*} \\
&\mu_T^* = \tfrac{\omega_T + \sum_{S \in \mathcal{G}} \omega_S \beta_T^S + \sum_{\{i\} \in \widetilde{\mathcal{G}}} \omega_{\{i\}} \kappa_T^{\{i\}}}{\sum_{R \in \mathcal{T}^{(k)*}} \omega_R} && \forall T \in \Delta_r^{(k-1)} \setminus \{N\} \\
&\mu_N^* = \tfrac{\omega_N + \sum_{S \in \mathcal{G}} \omega_S \beta_N^S + \sum_{\{i\} \in \widetilde{\mathcal{G}}} \omega_{\{i\}} \kappa_N^{\{i\}} - 1}{\sum_{R \in \mathcal{T}^{(k)*}} \omega_R}
\end{aligned}$$

Then at the start of Algorithm 5 $(\lambda^*, \gamma^*, \mu^*)$ satisfies (3.6) and in the subsequent update $\mathcal{U}$ becomes empty.
Conversely, suppose that $\mathcal{U} = \emptyset$. Lemma 3.8 shows that if $x^{(k)} \notin \mathcal{P}^{(k)*}$ then $\mathcal{U} \neq \emptyset$. Therefore by assumption $x^{(k)} \in \mathcal{P}^{(k)*}$ and thus $\mathcal{T}^{(k)*} \subseteq \mathcal{T}^{(k)}$ by Lemma 2.3. Denote by $x^{(k)*} \in \mathcal{P}^{(k)*}$ the solution such that $\mathcal{T}^{(k)}(x^{(k)*}) = \mathcal{T}^{(k)*}$. If $\mathcal{T}^{(k)} \setminus \mathcal{T}^{(k)*}$ is empty, we arrive to the desired conclusion $\mathcal{T}^{(k)} = \mathcal{T}^{(k)*}$. Supposing that $\mathcal{T}^{(k)*} \subsetneq \mathcal{T}^{(k)}$, we have

$$\begin{aligned}
E(S, x^{(k)*}) &< E(S, x^{(k)}) \quad \forall S \in \mathcal{T}^{(k)} \setminus \mathcal{T}^{(k)*} \neq \emptyset \\
E(S, x^{(k)*}) &= E(S, x^{(k)}) \quad \forall S \in \mathcal{T}^{(k)*} \cup \Delta_r^{(k-1)}
\end{aligned} \tag{3.10}$$

Since $\mathcal{U} = \emptyset$ we know that $\mathcal{T}^{(k)}$ is $\widetilde{\mathcal{T}}^{(k)}$-balanced with respect to $\Delta_r^{(k-1)}$. Then by Farkas' lemma $\nexists d \in \mathbb{R}$ such that $d(S) > 0$ for any $S \in \mathcal{T}^{(k)}$ while $d(S) \geq 0$ for all $S \in \mathcal{T}^{(k)} \cup \Delta_r^{(k-1)}$. However, this together with $d = x^{(k)*} - x^{(k)}$ contradicts (3.10), thus $\mathcal{T}^{(k)*} = \mathcal{T}^{(k)}$ □

Remark 3.11. Is is worth to note, that if $\mathcal{T} = \emptyset$ and $\mathcal{U} \neq \emptyset$, then $\varepsilon^{(k)}(x^{(k)} + \alpha d) < \varepsilon^{(k)}(x^{(k)})$ for any $d \in D$ and any $\alpha > 0$. Also, if $\mathcal{T}$,$\mathcal{U} \neq \emptyset$ then it can happen that $\varepsilon^{(k)}(x^{(k)} + \alpha d) = \varepsilon^{(k)}(x^{(k)})$, nevertheless $\Theta(E(x^{(k)} + \alpha d)) <_L \Theta(E(x^{(k)}))$ because satisfaction increases for all coalitions in $\mathcal{U} \subsetneq \mathcal{T}^{(k)}$.

As a result, by iteratively checking the balancedness of tight sets and taking positive steps along improving directions there is no circulation in the payoff space. If $\mathcal{U} \neq \emptyset$ at a current payoff point $x^{(k)}$, than the worst satisfaction belonging to some unsettled coalitions can be increased by taking a positive step in improving direction $d \in D$. In the following subsections we find suitable step sizes matching the improving directions,

guaranteeing no circulation in the tight set space either. Thereby we establish an algorithm that at some point reaches $\mathcal{U} = \emptyset$, thus not only solving ($\mathbf{P^{(k)}}$), but also finding the minimal tight set $\mathcal{T}^{(k)*}$ in the process.

3.1.4 Optimal step size

Suppose that for some k we find that $\mathcal{U} \neq \emptyset$ and we choose an improving direction $d \in D$. It is relatively straightforward to choose an optimal step size α for direction d, completing a pivot step in a simplex-like algorithm. Naturally we want to have large enough α to avoid small steps without changes in the tight set, since we found that $\mathcal{T}^{(k)}$ is not $\widetilde{\mathcal{T}}^{(k)}$-balanced with respect to $\Delta_r^{(k-1)}$. On the other hand as we increase α and experience a change in the tight set, we might find that the new tight set is balanced and we can proceed with the next iteration.

For all coalitions S, the change of excess as we move in direction d with step size α is

$$E(S, x^{(k)} + \alpha d) - E(S, x^{(k)}) = x^{(k)}(S) + \alpha d(S) - v(S) - x^{(k)}(S) + v(S) = \alpha d(S).$$

Currently we have $\varepsilon^{(k)}(x^{(k)}) = E(S, x^{(k)})$ for any $S \in \mathcal{T}^{(k)}$. Thus for sufficiently small $\alpha > 0$ the new minimal satisfaction is $\varepsilon^{(k)}(x^{(k)} + \alpha d) = E(S, x^{(k)} + \alpha d)$ for some (possibly more than one) $S \in \mathcal{T}^{(k)}$. Fix one such coalition as $\tilde{S}$ for the moment, then the change in the minimal satisfactions is $\varepsilon^{(k)}(x^{(k)}) - \varepsilon^{(k)}(x^{(k)} + \alpha d) = \alpha d(\tilde{S})$.

We are essentially interested in the *tightness* of coalitions measured as the difference of their excess from the maximal dissatisfaction, even more especially in the change of their tightness.

$$\begin{aligned} E(S, x^{(k)} + \alpha d) - \varepsilon^{(k)}(x^{(k)} + \alpha d) - E(S, x^{(k)}) + \varepsilon^{(k)}(x^{(k)}) &= \alpha d(\tilde{S}) - \alpha d(S) \\ &\geq \alpha(1 - d(S)), \end{aligned} \tag{3.11}$$

with the last inequality coming from $d(\tilde{S}) \geq 1$.

This leads us back to the practical question of how to choose improving directions from the half-space determined by (3.9). Since every feasible point of D is an improving direction we can use, we have the freedom to choose an objective function to optimise over this set. In order to control $\min_{S \in \mathcal{U}} d(S)$ as well as to make the bound we used in (3.11) sharp, we choose to minimise $\sum_{S \in \mathcal{U}} d(S)$, leading to the following LP for choosing improving direction d:

$$\begin{aligned} \min_{d} \quad & \sum_{S \in \mathcal{U}} d(S) & & \\ s.t. \quad & d(S) & \geq 1 \quad & \forall S \in \mathcal{U} \\ & d_i & \geq 0 \quad & \forall \{i\} \in \widetilde{\mathcal{U}} \\ & d(S) & = 0 \quad & \forall S \in \Delta \end{aligned} \tag{3.12}$$

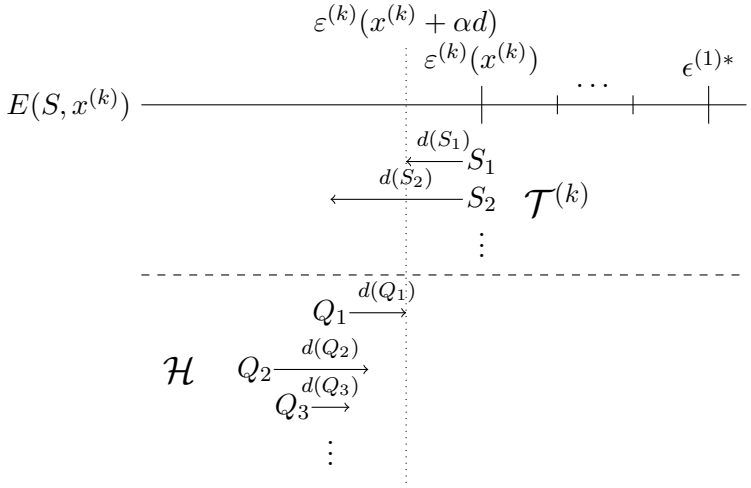


Figure 3.2: Optimal step size

.

As a result, for every optimal solution d of (3.12) we have $d(\tilde{S}) = 1$. Given $\alpha > 0$, from (3.11) we conclude that the tightness of coalition S decreases if $d(S) > 1$, it does not change if $d(S) = 1$ and it increases if $d(S) < 1$ as we increase α from 0. By increased tightness we mean that it gets closer to be tight, that is when the difference $E(S, x^{(k)} + \alpha d) - \varepsilon^{(k)}(x^{(k)} + \alpha d)$ decreases. Denote the collection of such coalitions with $\mathcal{H} = \{S \in \Sigma^{(k-1)} \setminus \mathcal{T}^{(k)} : d(S) < 1\}$.

We know that $E(S, x^{(k)}) < \varepsilon^{(k)}(x^{(k)})$ for all coalitions $S \in \Sigma^{(k-1)} \setminus \mathcal{T}^{(k)}$. Hence there exists $\alpha > 0$ sufficiently small that

$$E(S, x^{(k)}) + \alpha d(S) = E(S, x^{(k)} + \alpha d) \leq \varepsilon^{(k)}(x^{(k)} + \alpha d) \leq \varepsilon^{(k)}(x^{(k)}) + \alpha$$

Rearranging we get $E(S, x^{(k)}) + \alpha(d(S) - 1) \leq \varepsilon^{(k)}(x^{(k)})$. Candidates of coalitions satisfying the latter inequality with equality for a large enough α are collected in $\mathcal{H}$. However, α also need to satisfy additional bounds to guarantee that we stay in the imputation set. Taking both constraints into account, and introducing $\widetilde{\mathcal{H}} = \{j \in \Gamma^{(k-1)} \setminus \widetilde{\mathcal{T}}^{(k)} : d_j < 0\}$, the optimal step size is thus

$$\alpha = \min\left(\left\{\frac{\varepsilon^{(k)}(x^{(k)}) - E(S, x^{(k)})}{1 - d(S)} : S \in \mathcal{H}\right\} \cup \left\{\frac{x_j^{(k)} - v(\{j\})}{-d_j} : j \in \widetilde{\mathcal{H}}\right\}\right), \quad (3.13)$$

the smallest step size for which we experience either $\mathcal{T}^{(k)}(x^{(k)}) \neq \mathcal{T}^{(k)}(x^{(k)} + \alpha d)$ or $\widetilde{\mathcal{T}}^{(k)}(x^{(k)}) \neq \widetilde{\mathcal{T}}^{(k)}(x^{(k)} + \alpha d)$.

Figure 3.2 captures how the tight set changes as we move from $x^{(k)}$ to $x^{(k)} + \alpha d$. At $x^{(k)}$, the largest dissatisfaction outside of the already settled $\text{span}(\Delta_r^{k-1})$ belongs to coalitions in $\mathcal{T}^{(k)}$. Their dissatisfactions decrease with varying rates, depending on d, but with a

rate no smaller than 1. The new largest dissatisfaction $\varepsilon^{(k)}(x^{(k)} + \alpha d)$ is determined by coalitions in $\operatorname{argmin}_{S\in\mathcal{U}} d(S)$, controlled by the objective function of (3.12).

In Figure 3.2, coalitions in $\mathcal{H}$ increase their dissatisfaction (relative to the moving target of $\varepsilon^{(k)}(x^{(k)} + \alpha d)$), again with varying speed depending on d. The coalition(s) first meeting $\operatorname{argmin}_{S\in\mathcal{U}} d(S)$ enters the tight set[2].

3.1.4.1 Step size calculation

Since an overwhelming part of the computational burden of (3.13) falls to finding the minimum of the first term, with possibly exponential many unsettled coalitions involved belonging to $\mathcal{H}$, we focus on that when evaluating computational performance. Checking whether that minimum value is limited by the second term is considered negligible for this evaluation.

In a straightforward calculation of α, one can start with calculating $d(S)$, which requires the most work when $\Sigma^{(0)} = \mathcal{N}$ not counting $\mathcal{T}^{(1)}$. The scalar product $d(S) = e(S)d$ requires $2|S| - 1$ elementary operations worst case, that is assuming $d_i \neq 0\ \forall i \in N$, thus in total we need

$$\sum_{k=1}^{n-1} \binom{n}{k} (2k-1) = 2^n(n-2)/2 - n + 2 \tag{3.14}$$

elementary operations. This is unfortunately already $\mathcal{O}(n2^n)$, but it is easy to outperform this naive approach by exploiting sparsity and direction d.

We are primarily interested in $[e(S)_{S\in\Omega}]d$, where $\Omega = \Sigma^{(k-1)} \setminus \mathcal{T}^{(k)}$. We can further exploit sparsity of characteristic vectors through an important property of direction d, that it is a payoff reallocation since $N \in \Delta_r^{(k-1)}$ for all k, i.e. $\sum_i d_i = 0$. Consequently $d(S) = -d(N \setminus S)$.

Whenever $S \in \Omega$, this allows us to calculate $d(S)$ using the sparser characteristic vector of $e(S)$ and $e(N \setminus S)$. If $N \setminus S \in \Omega$ also, another multiplication comes with a single operation. Assuming full x and d vectors and neglecting the tight set $\mathcal{T}^{(1)}$, the whole calculation requires

$$2\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} k\binom{n}{k} + \left(\left\lceil \frac{n+1}{2} \right\rceil - \left\lceil \frac{n}{2} \right\rceil \right) \frac{n}{4} \binom{n}{\frac{n}{2}} \tag{3.15}$$

elementary operations. This is obviously less then half of (3.14) since we perform exactly half number of $d(S)$ multiplications, and get the denser part with a single sign change each. Unfortunately in the limit we can not expect much better bound though, as the dominant $\binom{n}{\frac{n}{2}}$ appears in both expressions.

[2] If there are multiple, all of them enter the tight set.

Once we found $d(S)$ in the above described manner for all $S \in \Omega$, we only calculate $x^{(k)}(S)$ if $d(S) < 1$ (or $d(S) > -1$ and $N \setminus S \in \Omega$), and even then we again can use the sparser characteristic vector because $x^{(k)}(N \setminus S) = v(N) - x^{(k)}(S)$. Algorithm 6 summarises the approach.

Algorithm 6: Calculating optimal step size α

Input: $v, N, x^{(k)}, d, \Sigma^{(k-1)}, \varepsilon^{(k)}(x^{(k)}), \mathcal{T}^{(k)}$; **Output:** α;
Initialisation: $\Omega = \Sigma^{(k-1)} \setminus \mathcal{T}^{(k)}, \alpha = \infty$;
while $\Omega \neq \emptyset$ **do**
  Choose $S \in \Omega$ arbitrary;
  if $|S| > \lfloor \frac{n}{2} \rfloor$ **then**
    $S = N \setminus S$;
  end
  $ed = e(S)d$;
  if $ed < 1$ *and* ($|S| \leq \lfloor \frac{n}{2} \rfloor$ *or* $S \in \Omega$) **then**
    $ex = e(S)x^{(k)}, \alpha = \min\left(\alpha, \frac{ex - v(S) - \varepsilon^{(k)}(x^{(k)})}{1 - ed}\right)$;
    if $ed > -1$ *and* $N \setminus S \in \Omega$ **then**
      $\alpha = \min\left(\alpha, \frac{v(N) - ex - v(N \setminus S) - \varepsilon^{(k)}(x^{(k)})}{1 + ed}\right)$;
    end
  else
    if $|S| \leq \lfloor \frac{n}{2} \rfloor$ *or* $S \in \Omega$ **then**
      $ex = e(S)x^{(k)}, \alpha = \min\left(\alpha, \frac{v(N) - ex - v(N \setminus S) - \varepsilon^{(k)}(x^{(k)})}{1 + ed}\right)$;
    end
  end
  $\Omega = \Omega \setminus \{S, N \setminus S\}$;
end
$\alpha = \min\left(\alpha, \min\left\{\frac{x_j^{(k)} - v(\{j\})}{-d_j} : j \in \widetilde{\mathcal{H}}\right\}\right)$;

3.1.5 The new algorithm and convergence

Now that every element of the proposed method is covered, we formulate Algorithm 7.

Algorithm 7 starts with an arbitrarily chosen imputation. If, at the current point, the tight set $\mathcal{T}^{(k)}$ fails to pass a balancedness requirement, we generate an improving direction and a step size, thereby performing a descent-like algorithm.

On the other hand, Algorithm 7 shares similarities with the simplex method for linear programming, as finding an improving direction d and a suitable step size α in Step 4 of the algorithm is essentially the same as a pivot step in the simplex algorithm. Inside the **while** loop the algorithm keeps pivoting until $\widetilde{\mathcal{T}}^{(k)}$-balancedness is achieved, while the iterations of the loop correspond to solving LPs in the sequential LP formulation of the nucleolus (cf. Chapter 2). The algorithm overall can also be interpreted as a column generation approach, because checking the balancedness of a collection of (primal) tight

Algorithm 7: Algorithm computing the nucleolus ν of cooperative game v with $I(v) \neq \emptyset$

Input: Game (N, v) with $I(v) \neq \emptyset$;
Output: ν nucleolus of game (N, v);

1. Initialisation: Set $x \in I(v)$ arbitrary, $\Delta_r^{(0)} = \{N\}$ and $k = 1$;
while $|\Delta_r^{(k-1)}| < n$ **do**
  2. Find

$$\mathcal{T}^{(k)}(x^{(k)}) = \underset{S \notin \operatorname{span}(\Delta_r^{(k-1)})}{\operatorname{argmax}} E(S, x^{(k)}),$$
$$\widetilde{\mathcal{T}}^{(k)}(x^{(k)}) = \left\{ \{i\} \in \Gamma^{(k-1)} : x_i^{(k)} = v(\{i\}) \right\}$$

  3. Generate output $\mathcal{T}$, $\mathcal{U}$, $\widetilde{\mathcal{T}}$, $\widetilde{\mathcal{U}}$, $\Delta =: \Delta_r^{(k)}$ with Algorithm 5;
  if $\mathcal{U} \neq \emptyset$ **then**
    4. Find d solving (3.12) and α using (3.13).
    Update $x^{(k)} = x^{(k)} + \alpha d$ and go to Step **2**.;
  else
    5. Set $\Gamma^{(k)} = N \setminus \operatorname{span}(\Delta_r^{(k)})$, $x^{(k+1)} = x^{(k)}$ and $k = k + 1$;
  end
end
6. $x^{(k)} = \nu$ is the nucleolus.

coalitions is nothing else than solving relaxed dual programs in the aforementioned LP sequence, sharing similarities with Derks and Kuipers (1997).

However, it is probably most suitable to be interpreted as an active-set method, as those dual relaxations are precisely determined by (the whole) active sets at a primal feasible point, and pivot steps, improving directions and step sizes are determined exactly to have suitable changes at the appropriate parts of the active sets in case of local non-optimality. The method again shares similar features with Derks and Kuipers (1997) in this sense as well, however there is a crucial difference, that we come back to in Section 3.2.

Example 3.2. *Let us continue with Example 3.1 to demonstrate how Algorithm 7 works, as well as the underlying geometrical intuition behind the method. For the sake of readability, in this example we use superscripts of payoff vectors to distinguish between different imputations, while superscripts of maximum dissatisfaction levels $\varepsilon^{(k)}$ and tight sets $\mathcal{T}^{(k)}$ are still keeping track of iterations. Figure 3.3 summarises the game and the solution approach in the (x_1, x_2) preimputation space, where $x_3 = v(\mathcal{N}) - x_1 - x_2$. The dotted lines show the coalitional rationality constraints and the gray trapezoid is the projection of the core $C(v)$ of the game onto the (x_1, x_2) space. We are using Algorithm 7 to find the nucleolus of ν from a starting imputation $x^0 = [1, 4, 7] \notin C(v)$.*

The underlying geometrical intuition can be easily followed through by considering Figure 3.4, which represents a magnified part of Figure 3.3. First, we find that our distance to the boundary of the core *is 1, hence $\varepsilon^{(1)}(x^0) = 1$. We also find that currently the largest infeasibility among core inequalities belongs to constraint $x_1 + x_2 \geq 6$. Therefore,*

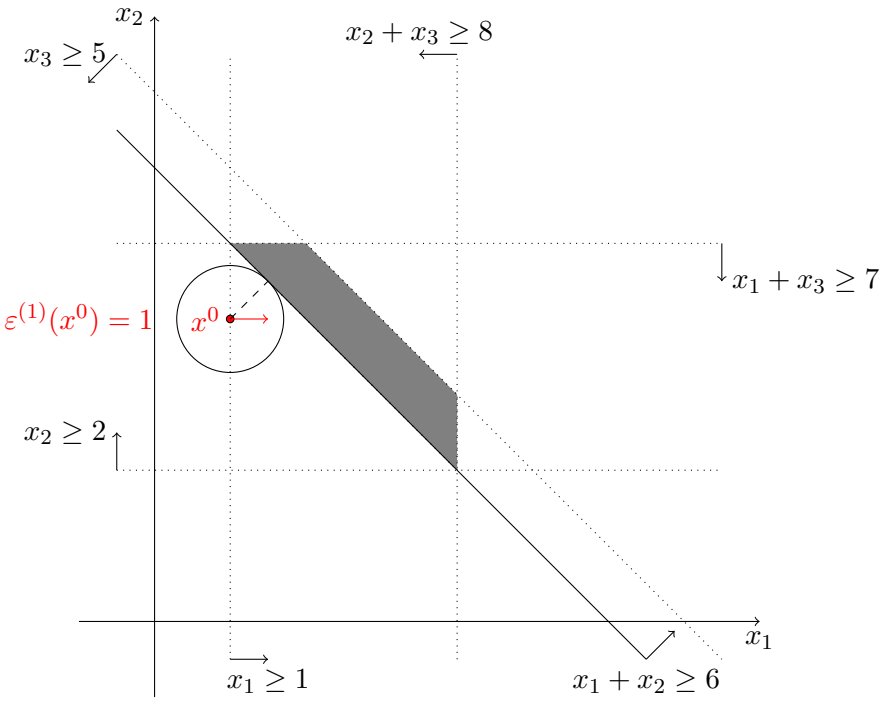


Figure 3.3: Hyperplane of preimputations: $x_1 + x_2 + x_3 = v(N) = 12$

$\mathcal{T}^1(x^0) = \{\{1,2\}\}$, *while* $\widetilde{\mathcal{T}}^1(x^0) = \{\{1\}\}$ *because* $x^0_1 = v(\{1\})$. *It is easy to see that the current tight set* $\mathcal{T}^1(x^0)$ *is not* $\widetilde{\mathcal{T}}^1(x^0)$*-balanced. In geometrical terms this means there are nearby points having smaller distance to the boundary of the core. In the algorithm we run into an infeasible system when checking the balancedness, so we can find improving directions by solving* (3.12). *As an example,* $d = [1, 0, -1]$ *is an improving direction depicted with an arrow from the current point* x^0. *Moving along this direction we can decrease the radius of the Euclidean ball touching the boundary of the core.*

Notice that here we measure the distance from the boundary with a special signed *distance; in the interior of the core the distance from the boundary is understood to be negative. Thus, we can move even further along the direction after reaching the core, until eventually we can not* decrease *the radius of the ball touching the boundary of the core any more, as Figure 3.4 illustrates. This happens precisely when the distances from both constraints* $(x_3 \geq 5)$ *and* $(x_1 + x_2 \geq 6)$ *are* -0.5*, that is, when* $x^1 = [2.5, 4, 5.5]$, $\varepsilon^{(1)}(x^1) = -0.5$, $\mathcal{T}^{(1)}(x^1) = \{\{1,2\},\{3\}\}$ *and* $\widetilde{\mathcal{T}}^1(x^1) = \emptyset$. *This corresponds to a pivot step in a simplex-like algorithm, where coalition* $\{3\}$ *enters the basis. In accordance with* (3.13)*, the step size chosen in direction* d *is* $\alpha = 1.5$. *After this step, we find that* $\mathcal{T}^{(1)}(x^1)$ *is* ($\widetilde{\mathcal{T}}^1(x^1)$*-)balanced (with respect to* $\Delta_r^{(0)} = N$*), therefore we expand* $\Delta_r^{(0)}$ *with an arbitrary element of* $\mathcal{T}^{(1)}(x^1)$ *as* $\operatorname{rank}\left(\Delta_r^{(0)} \cup \mathcal{T}^{(1)}(x^1)\right) = 2$. *By doing so we lift those inequality constraints that made it impossible to decrease the radius of the ball further; that is, throughout the remaining execution of the algorithm those constraints remain satisfied with equality at the current largest excess level of* $\varepsilon^{(1)}(x^1) = -0.5$.

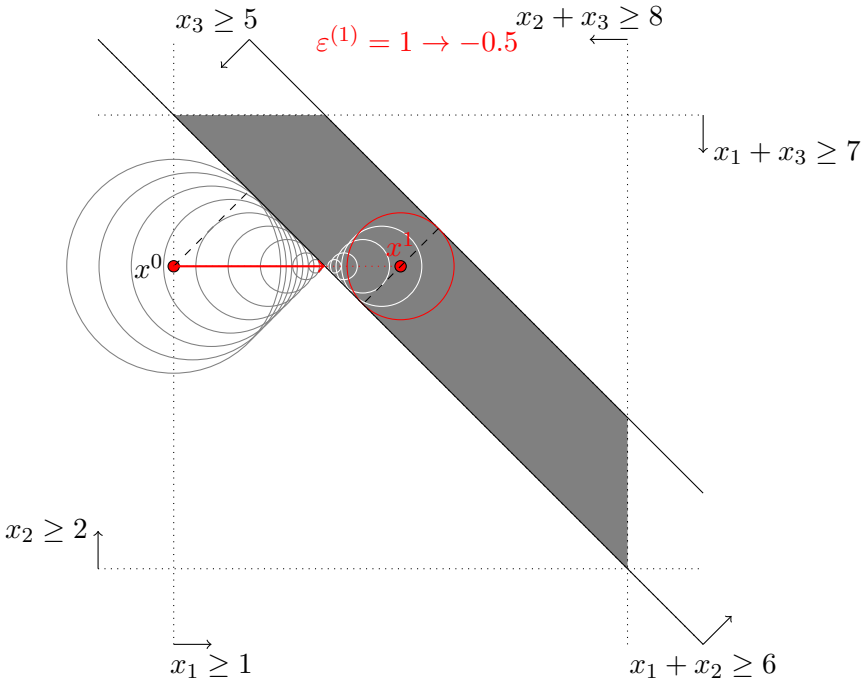


Figure 3.4: Pivot step

The solutions having the coalitions in $\mathcal{T}^{(1)}(x^1)$ *being tight at* $\varepsilon^{(1)}(x^1) = \epsilon^{(1)*} = -0.5$ *actually form the least core* $LC(v)$ *of the game, depicted by the thick black line in Figure 3.5. The dashed line shows that, at the current point* x^1*, among constraints not in* $\text{span}(\Delta_r^{(1)})$*, we are closest to violating* $x_1 + x_3 \geq 7$ *with* $\varepsilon^{(2)}(x^1) = -1$*. Further, we have that* $\widetilde{\mathcal{T}}^{(2)}$ *remains empty at* x^1 *and the tight set* $\mathcal{T}^{(2)}(x^1) = \{\{1,3\}\}$ *is not* $\widetilde{\mathcal{T}}^{(2)}$*-balanced with respect to* $\Delta_r^{(1)}$*, so we find an improving direction parallel to the set of least core solutions. That is, the unique solution of (3.12) is* $d = [1,-1,0]$*, and we can take a step size of* $\alpha = 1/4$ *until coalition* $\{2,3\}$ *becomes tight as well. The resulting point is* $x^2 = [2.75, 3.75, 5.5]^\top$ *with largest excess* $\varepsilon^{(2)}(x^2) = -1.25$*,* $\widetilde{\mathcal{T}}^{(2)}(x^2) = \emptyset$ *and tight set* $\mathcal{T}^{(2)}(x^2) = \{\{1,3\},\{2,3\}\}$*. Since* $\mathcal{T}^{(2)}(x^2)$ *is balanced with respect to* $\Delta_r^{(1)}$ *and* $\text{rank}(\Delta_r^{(1)} \cup \mathcal{T}^{(2)}(x^2)) = n$ *we found the nucleolus* $\nu = x^2$*.*

In the followings we establish results regarding the convergence of the algorithm.

Lemma 3.12. *Suppose that at iteration* k*, Algorithm 7 goes through to Step 4 and updates* $x^{(k)}$ *to* $x^{(k)} + \alpha d$*. Then we have* $\Theta(\overline{E}(x^{(k)} + \alpha d)) <_L \Theta(\overline{E}(x^{(k)}))$*. Furthermore,*

(a) if $\mathcal{T} = \emptyset$*, then* $\varepsilon^{(k)}(x^{(k)} + \alpha d) < \varepsilon^{(k)}(x^{(k)})$*,*

(b) if $\mathcal{T} \neq \emptyset$*, then* $\varepsilon^{(k)}(x^{(k)} + \alpha d) = \varepsilon^{(k)}(x^{(k)}) = \varepsilon^{(k)}(\nu)$ *and* $\mathcal{T}^{(k)}(x^{(k)} + \alpha d) = \mathcal{T}^{(k)}(\nu)$*.*

Proof. Let us start by noting that by the definition of d solving 3.12 and α in (3.13), the new point $(x^{(k)} + \alpha d) \in I(v)$ provided that $x^{(k)} \in I(v)$. Additionally $x^{(k)}(S) = (x^{(k)} + \alpha d)(S)$ for all $S \in \text{span}(\Delta_r^{(k-1)})$, also due to d solving 3.12. As a result, both $\Theta(\overline{E}(x^{(k)}))$

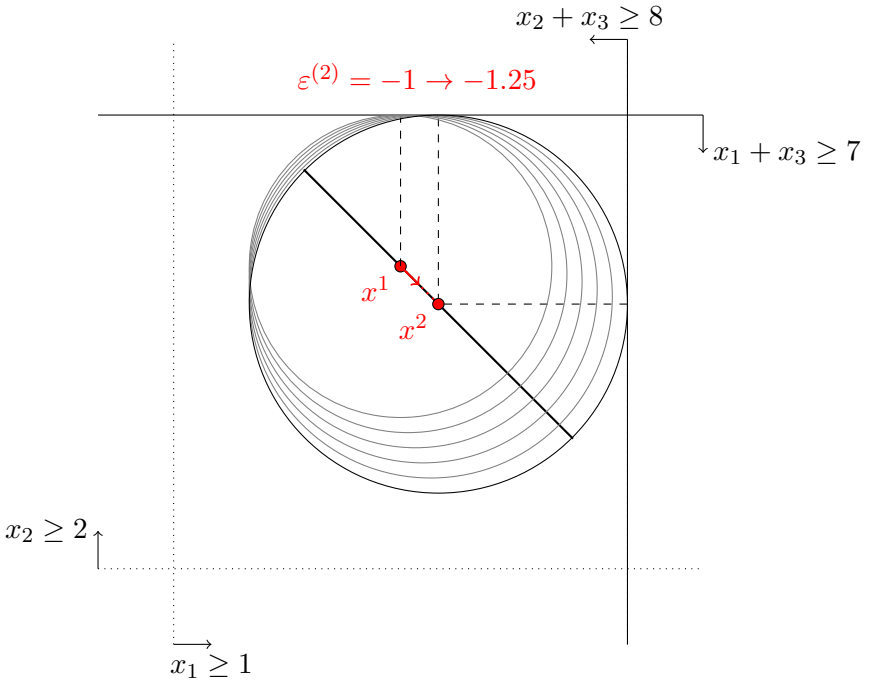


Figure 3.5: Pivot step

and $\Theta(\overline{E}(x^{(k)} + \alpha d))$ contain (and start with) the same excess values for coalitions S with excess $E(S, \nu) \geq \varepsilon^{(k-1)}(\nu)$. Therefore, in order to make the lexicographical comparison, it is sufficient to focus on the truncated ordered excess vectors over the set $\mathcal{M} := \mathcal{N} \setminus \text{span}(\Delta_r^{(k-1)})$, i.e., between $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)})\right)$ and $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)} + \alpha d)\right)$.

(a) The first component of both truncated ordered excess vectors is a value corresponding to a tight coalition $S \in \mathcal{T}^{(k)}(x^{(k)})$. Since $d(S) \geq 1$ for all $S \in \mathcal{T}^{(k)}(x^{(k)})$ and $\alpha > 0$, we have that $\varepsilon^{(k)}(x^{(k)} + \alpha d) < \varepsilon^{(k)}(x^{(k)})$ (as Figure 3.2 demonstrates) and therefore $\Theta(x^{(k)} + \alpha d) <_L \Theta(x^{(k)})$ in this case.

(b) However, if $\mathcal{T} \neq \emptyset$, then we have $d(S) = 0$ for all $S \in \mathcal{T}$ since d is a solution of 3.12, hence $\varepsilon^{(k)}(x^{(k)}) = \varepsilon^{(k)}(x^{(k)} + \alpha d) = E(S, x^{(k)})$ for $S \in \mathcal{T}$. Note that at Step 4, we have already assumed $\mathcal{U} \neq \emptyset$. Since $d(S) \geq 1$ for all $S \in \mathcal{U}$, we have $\mathcal{T} = \mathcal{T}^{(k)}(x^{(k)} + \alpha d)$, as Figure 3.6 demonstrates. Consequently, $\Theta(x^{(k)} + \alpha d) <_L \Theta(x^{(k)})$ because of $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)} + \alpha d)\right)$ starts with less excess values of $\varepsilon^{(k)}(x^{(k)})$ than $\Theta\left(\overline{E}\big|_{\mathcal{M}}(x^{(k)})\right)$; that is $|\mathcal{T}^{(k)}(x^{(k)} + \alpha d)| < |\mathcal{T}^{(k)}(x^{(k)})|$. On the other hand, since $\mathcal{T}^{(k)}(x^{(k)} + \alpha d)$ is $\widetilde{\mathcal{T}}^{(k)}$-balanced, we have $\varepsilon^{(k)}(x^{(k)} + \alpha d) = \varepsilon^{(k)}(\nu)$ and $\mathcal{T}^{(k)}(x^{(k)} + \alpha d) = \mathcal{T}^{(k)}(\nu)$.

□

Theorem 3.13. *Algorithm 7 stops after a finite number of steps. The* ***while*** *loop is executed at most* $(n-1)$ *iterations before finding the nucleolus* ν.

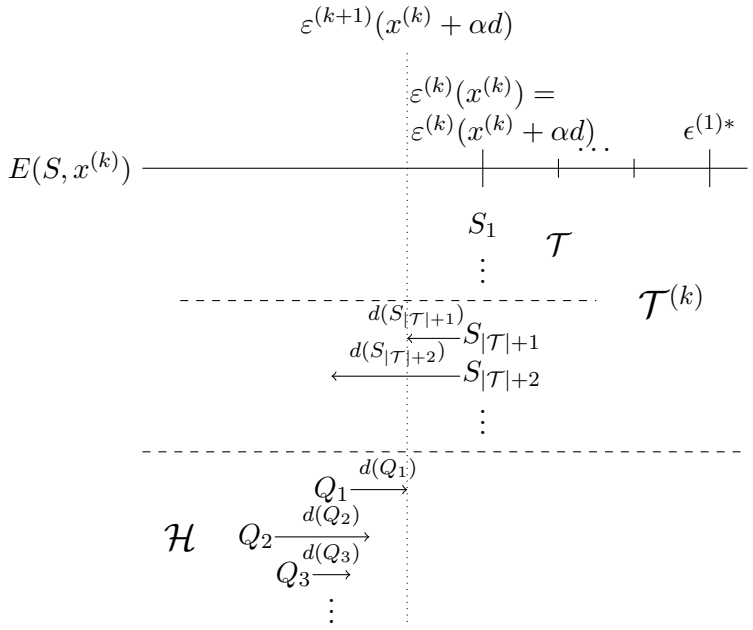


Figure 3.6: Changes of tight coalitions at a pivot step when $\mathcal{T} \neq \emptyset$.

Proof. We start by showing the iteration limit of the **while** loop. Note that $\operatorname{rank}(\Delta_r^{(k)})$ increases in every iteration as $\emptyset \neq \mathcal{T}^{(k)} \subseteq \Sigma^{(k-1)}$, and since $\operatorname{rank}(\Delta_r^{(0)}) = 1$ the algorithm terminates in at most $(n-1)$ iterations for the **while** loop.

Thus, for finite convergence to ν, we only need to show, that within a single iteration a finite number of pivot steps from $x^{(k)}$ to $x^{(k)} + \alpha d$ is sufficient to reach a tight set $\mathcal{T}^{(k)}$ that is $\widetilde{\mathcal{T}}^{(k)}$-balanced with respect to $\Delta_r^{(k-1)}$, which will get the algorithm out of the current iteration. For that purpose let us fix that iteration to be k, and for the remainder of the proof we omit the iteration superscripts (k); that is $\mathcal{T}(x)$ is used in place of $\mathcal{T}^{(k)}(x^{(k)})$ or $\mathcal{T}^{(k)}$, and so on.

According to Lemma 3.12, as soon as $\mathcal{T} \neq \emptyset$, we reached a $\widetilde{\mathcal{T}}(x)$-balanced tight set, $\mathcal{T}$ itself. Therefore we suppose that $\mathcal{T} = \emptyset$.

Since $\mathcal{T}(x) \subseteq \mathcal{N} \setminus \operatorname{span}(\Delta_r)$, the possible tight sets we can encounter is finite, among which there exists balanced as well, for instance $\mathcal{T}(\nu)$. Thus, the only way not to reach a balanced tight set is to encounter an infinite series of tight sets $T = \mathcal{T}^1, \mathcal{T}^2, \ldots, \mathcal{T}^m, \mathcal{T}^1, \mathcal{T}^{m+1} \ldots$ and $\widetilde{T} = \widetilde{\mathcal{T}}^1, \widetilde{\mathcal{T}}^2, \ldots, \widetilde{\mathcal{T}}^m, \mathcal{T}^1, \widetilde{\mathcal{T}}^{m+1} \ldots$, while among the former none is $\widetilde{\mathcal{T}}^i$-balanced. Here, we use the superscripts to denote the different pivot steps that we perform under the same iteration k. Again, because of the finitely many tight sets, we guaranteed to revisit at least one infinitely many times, w.l.o.g. let that be $\mathcal{T}^1$ and suppose that we first revisit it after taking m steps. Let us denote the corresponding improving directions, step sizes and maximal dissatisfactions we encounter at each tight set as $(d^1, d^2, \ldots)$, $(\alpha_1, \alpha_2, \ldots)$ and $(\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m, \varepsilon_{m+1}, \ldots)$, respectively.

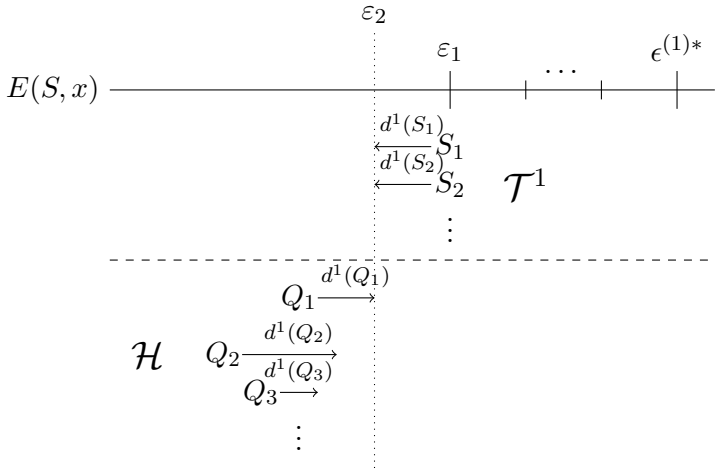


Figure 3.7: Changes of tight coalitions at a pivot step when $d^1(S) = 1$ for all $S \in \mathcal{T}^1$.

Let us suppose that the starting tight set $\mathcal{T}^1$ correspond to the imputation $x^1 = x$. By Lemma 3.12 we have $\varepsilon_1 > \varepsilon_m > \varepsilon_{m+1}$ and $x + \sum_{j=1}^m \alpha_j d^j \in I(v)$. Thus, we generate improving direction d^1 solving the LP

$$\begin{aligned} \min_d \quad & \sum_{Q \in \mathcal{T}^1} d(Q) & & & \\ s.t. \quad & d(Q) & \geq 1, \quad & \forall Q \in \mathcal{T}^1, \\ & d_i & \geq 0, \quad & \forall \{i\} \in \widetilde{\mathcal{T}}^1, \\ & d(S) & = 0, \quad & \forall S \in \Delta_r. \end{aligned} \tag{$ID(\mathcal{T}^1)$}$$

Since $\sum_{j=1}^m \alpha_j d^j(S) = \varepsilon_1 - \varepsilon_{m+1}$ is constant through $S \in \mathcal{T}^1$, for large enough $\beta > 0$ we have that $\beta \sum_{j=1}^m \alpha_j d^j$ is feasible in $ID(\mathcal{T}^1)$, and $\frac{\sum_{j=1}^m \alpha_j d^j}{\varepsilon_1 - \varepsilon_{m+1}}$ is optimal in $ID(\mathcal{T}^1)$. As a result, $d^1(S) = 1$ for all $S \in \mathcal{T}^1$ as d^1 is also an optimal solution of $ID(\mathcal{T}^1)$. Consequently all coalitions remain tight, while based on the definition of the step size, some coalition must join (as Figure 3.7 demonstrates), yielding us $\mathcal{T}^1 \subsetneq \mathcal{T}^2$.

Note that it is not necessarily always the case that $\mathcal{T}^1$ is followed by the same $\mathcal{T}^2$. Even though upon the re-occurrence of $\mathcal{T}^1$, the LP ($ID(\mathcal{T}^1)$) we need to solve to generate improving direction d^{m+1} is the same as before, the solution of the LP is not necessarily unique. Therefore $\mathcal{T}^{m+1} = \mathcal{T}^2$ is not guaranteed, only $\mathcal{T}^1 \subsetneq \mathcal{T}^{m+1}$, using the same argument as above. However, T is an infinite series of tight sets, with all of it's elements coming from a subset of the finite set $\mathcal{N} \setminus \text{span}(\Delta_r)$, the non-$\widetilde{\mathcal{T}}^i$-balanced tight sets.

Therefore, w.l.o.g. we can truncate the beginning of the series, throwing away a portion that includes all occurrence of all elements that appear only finitely many times in the series. Let us truncate the series up to the first $\mathcal{T}^1$ after the last occurrence of such an element appearing only finitely many times. Furthermore, we can choose m such a way, that in the truncated series $\mathcal{T}^1$ is followed by $\mathcal{T}^2$ and $\mathcal{T}^m$ is followed by $\mathcal{T}^1$ infinitely many times, hence w.l.o.g. we can assume $\mathcal{T}^2 = \mathcal{T}^{m+1}$.

In fact, we can assume that after throwing away only finitely many elements from the beginning of the infinite series T, we get an infinite series of non-balanced tight sets where *exactly* $\mathcal{T}^1, \mathcal{T}^2, \ldots, \mathcal{T}^m$ is repeated infinitely many times. However, as $\mathcal{T}^1 \subsetneq \mathcal{T}^2$, applying the same argument with $\mathcal{T}^2$ and $\mathcal{T}^3$ taking the roles of $\mathcal{T}^1$ and $\mathcal{T}^2$, and so on so forth, after $(m-1)$ shifts in the series we conclude that $\mathcal{T}^1 \subsetneq \mathcal{T}^1$, an obvious contradiction. Thus there exists no tight set we can encounter infinitely many times throughout Algorithm 7, hence the algorithm converges in finitely many steps. □

Remark 3.14. Note that requiring a finite number of pivots is enough to achieve the result claimed above. Numerical experiments indicate that Algorithm 7 requires a small number of pivots (cf. Section 5.4).

Finally, note that not only the step size calculation using Algorithm 6, but the whole Algorithm 7 can be straightforwardly adapted to the case when one does have a *characterisation set* $\mathcal{F} \subsetneq \mathcal{N}$ available, a subset of coalitions enough to determine the nucleolus, by simply initialising the set of unsettled coalitions as $\Sigma^{(0)} = \mathcal{F}$.

3.2 Derks and Kuipers (1997)'s method

In their unpublished work Derks and Kuipers (1997) present a method to calculate the prenucleolus which in essence solve the sequence of dual LPs ($\mathbf{D^{(k)}}$) with a specific column generation, or even more precisely, an active-set framework. This section is devoted to present their method adapted to the nucleolus and to highlight the key differences compared to our approach.

The authors find $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)}$ by repeatedly solving the following relaxed system:

$$\begin{aligned} \sum_{S \in \sigma^{(k-1)}} u_S^{(k)} e(S)_i + \quad w_{\{i\}}^{(k)} \tilde{e}(\tilde{\sigma}^{(k-1)})_i + \sum_{S \in \Delta_r^{(k-1)}} z_S^{(k)} e(S)_i \quad &= 0 \quad \forall i \in N \\ \sum_{S \in \sigma^{(k-1)}} u_S^{(k)} \quad &= 1 \\ w_{\{i\}}^{(k)} \quad &\geq 0 \quad \forall \{i\} \in \tilde{\sigma}^{(k-1)} \end{aligned} \tag{3.16}$$

where collections of coalitions $\sigma^{(k-1)} \subsetneq \Sigma^{(k-1)}$ and $\tilde{\sigma}^{(k-1)} \subsetneq \Gamma^{(k-1)}$ is built up element by element. First find

$$\varepsilon^{(k)}(x^{(k)}) = \max_{S \in \Sigma^{(k-1)}} E(S, x^{(k)}) \tag{3.17}$$

for an arbitrarily chosen $x^{(1)} \in I(v)$ in the first iteration. Naturally $(x^{(k)}, \varepsilon^{(k)}(x^{(k)})) \in \mathcal{P}^{(k)}$, and so far the process is the same, introduced above. Then proceed by picking any coalition $S \in \Sigma^{(k-1)}$ with $E(S, x^{(k)}) = \varepsilon^{(k)}(x^{(k)})$ to initialise $\sigma^{(k-1)} = S$ and $\tilde{\sigma}^{(k-1)} = \emptyset$. Hence the system (3.16) is a primal-dual relaxation of the underlying LPs in the sense

of including only some of the dual variables with no sign restrictions on $u^{(k)}$. Notice that in this setting the matrix

$$\begin{bmatrix} e(S)_{S\in\Delta_r^{(k-1)}} & 0 \\ e(\{i\})_{\{i\}\in\tilde{\sigma}^{(k-1)}} & 1 \\ e(S)_{S\in\sigma^{(k-1)}} & 1 \end{bmatrix}$$

has full row-rank, and therefore the system (3.16) has at most one solution $(u^{(k)}, w^{(k)}, z^{(k)})$. At this point the authors distinguish between 3 cases.

1. There is a (unique) solution $(u^{(k)}, w^{(k)}, z^{(k)})$ with $u^{(k)} \geq 0$. Then it follows that $(x^{(k)}, \varepsilon^{(k)}(x^{(k)})) \in \mathcal{P}^{(k)}$ and $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)}$ are a primal-dual feasible pair with the same objective function value and thus both are optimal.

2. There is a (unique) solution $(u^{(k)}, w^{(k)}, z^{(k)})$ with $u_S^{(k)} < 0$ for some $S \in \sigma^{(k-1)}$. Suppose S_i has the smallest i among these coalitions. Removing S_i from $\sigma^{(k-1)}$ yields an infeasible (3.16) and hence by Farkas' lemma

$$\begin{array}{lll} d(S) & = 1 & \forall S \in \sigma^{(k-1)} \\ d_i & \geq 0 & \forall \{i\} \in \tilde{\sigma}^{(k-1)} \\ d(S) & = 0 & \forall S \in \Delta_r^{(k-1)} \end{array} \tag{3.18}$$

   is feasible. Solutions $d \in \mathbb{R}^n$ satisfying (3.18) are called *improving directions* by Solymosi (1993), but notice that these are different from the one defined in (3.9).

3. The system (3.16) has no solution. Similarly by Farkas' lemma (3.18) is feasible (with unchanged $\sigma^{(k-1)}$).

In the first case we solved the k-th level LPs, $\varepsilon^{(k)}(x^{(k)}) = \epsilon^{(k)*}$, $x^{(k)} \in \mathcal{P}^{(k)*}$, $\Delta_r^{(k-1)}$ can be expanded from $\mathcal{NT}^{(k)}(u^{(k)}) \subseteq \mathcal{T}^{(k)*}$ and $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) \subseteq \widetilde{\mathcal{T}}^{(k)*}$ to get $\Delta_r^{(k)}$, $\Sigma^{(k)}$ and $\Gamma^{(k)}$ can be updated to $\mathcal{N} \setminus \operatorname{span}(\Delta_r^{(k)})$ and $N \setminus \operatorname{span}(\Delta_r^{(k)})$ we can proceed with next iteration (if $\operatorname{rank}(\Delta_r^{(k)}) < n$).

In the other 2 cases we make a *pivot step*. Choose an arbitrary improving direction d satisfying (3.18) and the largest step size α such that $(x^{(k)} + \alpha d, \varepsilon^{(k)}(x^{(k)}) - \alpha) \in \mathcal{P}^{(k)}$, that is

$$\alpha = \min\left(\left\{\frac{\varepsilon^{(k)}(x^{(k)}) - E(S, x^{(k)})}{1 - d(S)} : S \in \mathcal{J}\right\} \cup \left\{\frac{x_j^{(k)} - v(\{j\})}{-d_j} : \{j\} \in \widetilde{\mathcal{J}}\right\}\right), \tag{3.19}$$

where $\mathcal{J} := \{S \in \Sigma^{(k-1)} \setminus \sigma^{(k-1)}, d(S) < 1\}$ and $\widetilde{\mathcal{J}} := \left\{\{j\} \in \Gamma^{(k-1)} \setminus \tilde{\sigma}^{(k-1)} : d_j < 0\right\}$. Then update $x^{(k)}$ to $x^{(k)} + \alpha d$, $\varepsilon^{(k)}(x^{(k)})$ becomes $\varepsilon^{(k)}(x^{(k)}) - \alpha$ and expand $\sigma^{(k-1)}$ with S_i, where i is the lowest index among coalitions in $\left\{S \in \Sigma^{(k-1)} : d(S) < 1\right\}$ that satisfy $\alpha = \left(\varepsilon^{(k)}(x^{(k)}) - E(S, x^{(k)})\right) / (1 - d(S))$, or expand $\tilde{\sigma}^{(k-1)}$ with $\{j\}$, where j is the lowest

index among players in $\left\{\{j\} \in \Gamma^{(k-1)} : d_j < 0\right\}$ that satisfy $\alpha = \left(x_j^{(k)} - v(\{j\})\right) / (-d_j)$. If $\alpha > 0$, remove all $\{j\}$ from $\tilde{\sigma}^{(k-1)}$ such that $d_j > 0$. The resulting system (3.16) again has full row-rank, hence the process may be repeated.

Notice that in every pivoting step $|\sigma^{(k-1)}|$ either increases by 1, decreases by 1 or stays the same. Thus Derks and Kuipers (1997)'s method can also be viewed as column generation, as a simplex-like algorithm, or as an active set method. The process altogether, with the step size α as (3.19) is very similar to our method. However, the approach of expanding $\sigma^{(k-1)}$ shows a key difference between the methods, namely working with only a *partial* active set. As long as the system (3.16) is in Case 2 or 3, that includes building up $\mathcal{T}^{(k)}$ element by element, and $\{S \in \Sigma^{(k-1)} \setminus \sigma^{(k-1)} : d(S) < 1\}$ is non-empty, $\sigma^{(k-1)}$ keeps on being expanded, in a pivot step with $\alpha = 0$. Even if the tight set is the smallest, that is $\mathcal{T}^{(k)}(x^{(k)}) = \mathcal{T}^{(k)*}$, $\sigma^{(k-1)}$ needs to be built one by one. However, note that tight sets $\mathcal{T}^{(k)}$ can grow very large, up to be of exponential size. In the active set method formulated in Algorithm 7, $\alpha > 0$ is guaranteed in every single pivot step.

But a difference with even more serious consequences is that even in Case 1, we can end up paying a high price in computation because of using only a partial active set (the dual non-tight set), updating $\Delta_r^{(k-1)}$ with $\mathcal{NT}^{(k)}(u^{(k)}) \subseteq \mathcal{T}^{(k)*}$ and $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) \subseteq \widetilde{\mathcal{T}}^{(k)*}$, not guaranteeing maximal rank increase that leads to minimal number of iterations, as Remark 2.7 and the following small example demonstrates.

Example 3.3. *Take the example of the following 3-player, 0-normalized game v for any $\beta > 0$:*

$$v(\{i\}) = 0 \ \forall i \in N; v(\{1,2\}) = \frac{\beta}{3}; v(\{1,3\}), v(\{2,3\}) < \frac{\beta}{3}; v(N) = \beta.$$

Suppose we start from a straightforward imputation of equal surplus division (ESD) $x_i^{(1)} = v(N)/n = 1, \forall i = 1,2,3$. *Then $\varepsilon^{(k)}(x^{(1)}) = -\beta/3$ and the tight set $\mathcal{T}^{(1)}(x^{(1)}, \varepsilon^{(k)}(x^{(1)}))$ ($\widetilde{\mathcal{T}}^{(1)}(x^{(1)}) = \emptyset$) is actually the minimal tight set $\mathcal{T}^{(1)*}$ ($\widetilde{\mathcal{T}}^{(1)*} = \emptyset$) consisting of $\{1,2\}$ along with the singleton coalitions. Naturally* $\operatorname{rank}(\mathcal{T}^{(1)*} \cup N) = n$, *hence we could finish in the first iteration.*

Suppose instead we follow Derks and Kuipers (1997). We are left free to choose any coalition from $\mathcal{T}^{(1)}(x^{(1)}, \varepsilon^{(k)}(x^{(1)}))$, say $\sigma^{(0)} = \{1,2\}$. Since (3.16) is infeasible we choose a direction $d = [\gamma, 1-\gamma, -1]^\top$ solving (3.18). For any γ, $\alpha = 0$ and $d(\{3\}) = -1 < 1$, with additionally $d(\{1\}) < 1$ and/or $d(\{2\}) < 1$, depending on γ. We are left with the indexing of coalitions to choose which one enters $\sigma^{(0)}$. Following the order that Derks and Kuipers (1997) uses to calculate α (cf. Subsection 3.2.1) we choose $\sigma^{(0)} = \{\{1,2\},\{3\}\}$ thus we get a feasible (3.16). Now we found a dual feasible $u^{(1)}$ with $\mathcal{NT}^{(1)}(u^{(1)}) = \sigma^{(0)}$, so we can expand $\Delta_r^{(0)}$ to have a rank of 2. Unfortunately however, this leads us to perform another iteration.

The above small example evidently exploits that the starting imputation $x^{(1)}$ is the nucleolus. However, often we experience extra unnecessary pivots ($\alpha = 0$) from Derks and Kuipers (1997)'s approach not only when the current solution is the nucleolus, and not even only when $\mathcal{T}^{(k)} = \mathcal{T}^{(k)*}$ with $\operatorname{rank}(\Delta_r^{(k-1)}) < n-1$. Furthermore, the example highlights the phenomena that their method can take more iterations whenever the minimal tight set contains a subcollection of coalitions $\mathcal{B} \subsetneq \mathcal{T}^{(k)*}$ that is $\widetilde{\mathcal{T}}^{(k)}$-balanced with respect to $\Delta_r^{(k-1)}$. The impact of these *weaknesses* of the method can be clearly seen in the results and findings of Section 5.4.

3.2.1 Step size calculation

Derks and Kuipers (1997) also offers a way to compute optimal step size α with $\mathcal{O}(2^n)$ complexity. They exploit how characteristic vectors of coalitions $e(S)$ are related to each other. Similarly to Subsection 3.1.4.1, we ignore the second term of the minimisation, as it has no influence on computational complexity.

Let us suppose that the fixed indices of coalitions in $\mathcal{N}$ follow a lexicographically increasing order in the characteristic vectors, that is $e(S_1) = (0, \dots, 0, 1)$, $e(S_2) = (0, \dots, 0, 1, 0)$ and $e(S_{2^n-2}) = (1, \dots, 1, 0)$. Furthermore, for $j = 1, \dots, n$ let us denote with $h^j \in \mathbb{R}^n$ the vector that contains zeros in the first $j-1$ coordinates, then a 1 in the j-th coordinate, followed by -1 in the remaining $n-j$ coordinates. Observe that if j is the rightmost position of a zero in $e(S_i)$, then $e(S_{i+1}) = e(S_i) + h^j$. Algorithm 8 shows how Derks and Kuipers (1997) calculates α given the current feasible point $(x^{(k)}, \varepsilon^{(k)}(x^{(k)}))$, improving direction d and unsettled coalitions $\Sigma^{(k-1)}$.

Algorithm 8 exploits that the first $j-1$ elements of h^j are zero, hence the number of elementary operations in an iteration of the for loop is bounded by a constant multiple of $n-j+1$. Since the number of n-dimension binary vectors with the rightmost position of a zero at j is 2^{j-1}, the total complexity of their approach is

$$\mathcal{O}\left(\sum_{j=1}^{n}(n-j+1)2^{j-1}\right) = \mathcal{O}(2^{n+1} - n - 2) = \mathcal{O}(2^n)$$

Apart from the obvious significant improvement Algorithm 8 introduces, observe one main disadvantage, that is, the for loop requires $2^n - 3$ iterations even when the size of $\Sigma^{(k-1)}$ is much less (although $|\Sigma^{(k-1)}| \geq 2^{n-1} - 2$ throughout the whole process). Furthermore, if in fact $d(S) < 1$ for just a single coalition $S \in \Sigma^{(k-1)}$, then calculating all the *actually necessary* $x(S)$ requires only $2|S| - 1$ operations, while in the setting of Algorithm 8 *ex* still requires $2^n - 2$ updates.

Comparing to our approach, even though our step size calculation is still of order $\mathcal{O}(n2^n)$, it has certain benefits. The required number of elementary operations in the main part of

Algorithm 8: Derks and Kuipers (1997)'s method of computing α

Input: $v, n, x^{(k)}, \varepsilon^{(k)}(x^{(k)}), d, \Sigma^{(k-1)}, (h^j)_{j=1}^n$; **Output:** α;
Initialisation: $e = e(S_1), ex = x_n, ed = d_n$;
if $ed < 1$ *and* $S_1 \in \Sigma^{(k-1)}$ **then**
  $\alpha = \frac{\varepsilon^{(k)}(x^{(k)}) + ex - v(S_1)}{1 - ed}$;
else
  $\alpha = \infty$;
end
for $i = 2, \dots 2^n - 2$ **do**
  $j = n$;
  while $j > 0$ *and* $e_j = 1$ **do**
    $j = j - 1$;
  end
  $e = e + h^j$;
  $ex = ex + h^j x$;
  $ed = ed + h^j d$;
  if $ed < 1$ *and* $S_i \in \Sigma^{(k-1)}$ **then**
    $\alpha = \min\left(\alpha, \frac{\varepsilon^{(k)}(x^{(k)}) + ex - v(S_i)}{1 - ed}\right)$;
  end
end
$\alpha = \min\left(\alpha, \min\left\{\frac{x_j^{(k)} - v(\{j\})}{-d_j} : j \in \widetilde{\mathcal{J}}\right\}\right)$;

the computation, (3.15) decreases as k increases, that is as $\Sigma^{(k-1)}$ contains less and less coalitions. Whereas Algorithm 8 depending on a specific order of the coalitions always have to do exactly $2^n - 3$ iterations calculating all the $d(S)$ and $x^{(k)}(S)$ for $S \in \mathcal{N}$.

Furthermore, an important consequence along the shrinking $\Sigma^{(k-1)}$ is the handling of characterisation sets. While our step size calculation is capable to adapt to this situation with the initialisation $\Sigma^{(0)} = \mathcal{F}$, Derks and Kuipers (1997)'s method still requires performing the whole exponential loop, due to their special treatment on the order of coalitions.

Chapter 4

Algorithm variants

This chapter is devoted to develop, using the results of Chapters 2 and 3, variants of existing algorithms, outperforming their classical counterparts. In Section 4.1 we focus on theoretical results related to the verification of the nucleolus by the Kohlberg criterion, while in Section 4.2 we introduce variants of classical sequential LP methods dissolving a trade-off between primal and dual methods.

4.1 Verification by the Kohlberg criterion

This section is based on joint work (Benedek et al. (2018)) with Tri-Dung Nguyen and Jörg Fliege

Kohlberg (1971) introduced necessary and sufficient conditions for an imputation to be the nucleolus based on the so-called *coalitional arrays*.

Definition 4.1. Collections of coalitions $\mathcal{T}^0, \mathcal{T}^1, \ldots, \mathcal{T}^p \subseteq \mathcal{N}$ form a coalitional array belonging to imputation $x \in I(v)$ if

$$
\begin{aligned}
&\mathcal{T}^0 = \{i \in N : x_i = v(\{i\})\} \\
&\mathcal{T}^1 = \operatorname*{argmax}_{S \in \mathcal{N}} E(S,x), && \epsilon^1 = \max_{S \in \mathcal{N}} E(S,x) \\
&\mathcal{T}^2 = \operatorname*{argmax}_{S \in \mathcal{N} \setminus \mathcal{T}^1} E(S,x), && \epsilon^2 = \max_{S \in \mathcal{N} \setminus \mathcal{T}^1} E(S,x) \\
&\qquad\vdots && \qquad\vdots \\
&\mathcal{T}^k = \operatorname*{argmax}_{S \in \mathcal{N} \setminus \cup_{i=1}^{k-1} \mathcal{T}^i} E(S,x), && \epsilon^k = \max_{S \in \mathcal{N} \setminus \cup_{i=1}^{k-1} \mathcal{T}^i} E(S,x) \quad \forall 1 \leq k \leq p
\end{aligned}
$$

such that $\cup_{i=1}^{p} \mathcal{T}^i = \mathcal{N}$.

That is, if $(\mathcal{T}^0, \mathcal{T}^1, \ldots, \mathcal{T}^p)$ is the coalitional array belonging to imputation $x \in I(v)$, then $\mathcal{T}^0 = \widetilde{\mathcal{T}}^{(1)}(x)$, $\epsilon^1 = \varepsilon^{(1)}(x)$ and $\mathcal{T}^1 = \mathcal{T}^{(1)}(x)$. We highlight the difference in approach by omitting imputation x as an argument for the coalitional array, as when it comes to verifying whether $x = \nu$ or not, the underlying imputation is fixed. Moreover, typically $\mathcal{T}^2 \neq \mathcal{T}^{(2)}(x)$. Note that for all $S \in \mathcal{N}$ there is a unique $\mathcal{T}^k$ containing coalition S, and $\epsilon^1 > \epsilon^2 > \cdots > \epsilon^p$.

	$S \subseteq N$	$E(x,S)$
$\mathcal{T}^1$	S_1	
	$\vdots$	ε_1
	S_{m_1}	
$\mathcal{T}^2$	S_{m_1+1}	
	$\vdots$	ε_2
$\vdots$	$\vdots$	$\vdots$
$\mathcal{T}^p$	$\vdots$	
	S_{2^n-2}	ε_p

Table 4.1: $(\mathcal{T}^0, \ldots, \mathcal{T}^p)$ coalitional array

Kohlberg defines two properties of coalitional arrays that characterise the nucleolus.

Definition 4.2. Coalitional array $(\mathcal{T}^0, \mathcal{T}^1, \ldots, \mathcal{T}^p)$

- has *property I* (P1) if for all $y \in \mathbb{R}^n$ and $k = 1, \ldots, p$:

$$\left.\begin{array}{lll} y(S) \geq 0 & \forall S \in \cup_{j=1}^{k} \mathcal{T}^j \\ y_j \geq 0 & \forall j \in \mathcal{T}^0 \\ y(N) = 0 \end{array}\right\} \Rightarrow y(S) = 0 \quad \forall S \in \cup_{j=1}^{k} \mathcal{T}^j$$

- has *property II* (P2) if for all $k = 1, \ldots, p : \cup_{i=1}^{k} \mathcal{T}^i$ is $\mathcal{T}^0$-balanced, that is $\exists \omega \geq 0$:

$$\begin{array}{lll} \sum\limits_{S \in \cup_{j=0}^{k} \mathcal{T}^j} \omega_S e(S) & = e(N) & \\ \omega_S & > 0 & \forall S \in \cup_{j=1}^{k} \mathcal{T}^j. \end{array}$$

Using these properties we can state Kohlberg's conditions.

Theorem 4.3 (Kohlberg criteria)**.** *Given imputation $x \in I(v)$ the followings are equivalent:*

- *x is the nucleolus of v*
- *the coalitional array belonging to x has P1*

- *the coalitional array belonging to x has P2*

In light of Theorem 4.3, Property I states that for any dissatisfaction level, if we find a payoff reallocation ($y(N) = 0$) that does not take payoff away from coalitions having one of the k-th worst dissatisfaction level and from players who are on the boundary of individual rationality, then that payoff reallocation actually gives nothing to coalitions with the k-th worst dissatisfaction level. This is somewhat of a local optimality condition, and it translates to not having improving directions ($D = \emptyset$) in our setting.

Property II provides a condition that is easier to check directly in the form of a feasibility problem. Accordingly we formulate Algorithm 9.

Algorithm 9: Kohlberg algorithm verifying $x = \nu$ or not.

Input: N, v (with $I(v) \neq \emptyset$), $x \in I(v)$;
Output: Conclude if $x = \nu$ or not;
1. Initialisation: Set $\Delta^0 = N$, $\mathcal{T}^0 = \{\{j\} : x_j = v(\{j\}), j \in N\}$ and $k = 1$;
while $\Delta^{k-1} \neq \mathcal{N}$ **do**
  2. Find $\mathcal{T}^k = \underset{S \notin \Delta^{k-1}}{\operatorname{argmax}} E(S, x)$;
  if $\Delta^{k-1} \cup \mathcal{T}^k$ *is* $\mathcal{T}^0$*-balanced* **then**
    3. Set $\Delta^k = \Delta^{k-1} \cup \mathcal{T}^k$ and $k = k + 1$;
  else
    4. STOP and conclude that $x \neq \nu$;
  end
end
5. Conclude that $x = \nu$;

Even though balancedness is originally stated as a strict feasibility problem, Lemmas 3.2, 3.4 and 3.5 allow us to check the condition using Algorithm 5. However, the obvious downside of Algorithm 9 is that it takes p iterations, as many as distinguished dissatisfaction levels there are. Worst case, that could be exponential. Furthermore, for every game when verifying the nucleolus, in the last iteration p, $|\Delta^{p-1} \cup \mathcal{T}^p| = 2^n - 2$, making checking balancedness intractable.

We can overcome these difficulties using Lemma 3.4, allowing us to formulate an improved version of the Kohlberg algorithm. Differences between the original Algorithm 9 and the improved version Algorithm 10 are highlighted in red. The restricted search space in Step 2 guarantees that the rank of the collection Δ^{k-1} increases in every iteration, while it starts from initially having rank of 1. Lemma 3.4 guarantees that the restricted search space does not effect the outcome of the verification, and it allows us to modify the stopping criterion accordingly. Thus the number of iterations performed by the improved Kohlberg algorithm is immediately reduced to at most $n - 1$. However, we face the same issues with the size of the collection of coalitions.

Algorithm 10: Improved Kohlberg algorithm verifying $x = \nu$ or not.

Input: N, v (with $I(v) \neq \emptyset$), $x \in I(v)$;
Output: Conclude if $x = \nu$ or not;
1. Initialisation: Set $\Delta^0 = N$, $\mathcal{T}^0 = \{\{j\} : x_j = v(\{j\}), j \in N\}$ and $k = 1$;
while $\text{rank}(\Delta^{k-1}) < n$ **do**
  2. Find $\mathcal{T}^k = \underset{S \notin \text{span}(\Delta^{k-1})}{\text{argmax}} \; E(S, x)$;
  if $\Delta^{k-1} \cup \mathcal{T}^k$ *is* $\mathcal{T}^0$*-balanced* **then**
    3. Set $\Delta^k = \Delta^{k-1} \cup \mathcal{T}^k$ and $k = k + 1$;
  else
    4. STOP and conclude that $x \neq \nu$;
  end
end
5. Conclude that $x = \nu$;

Lemma 3.2 allows us to introduce a representative set Δ_r of union of tight sets Δ, to take care of compactly representing extremely large (union of) tight sets. Differences between

Algorithm 11: Improved Kohlberg algorithm with compact representation verifying $x = \nu$ or not.

Input: N, v (with $I(v) \neq \emptyset$), $x \in I(v)$;
Output: Conclude if $x = \nu$ or not;
1. Initialisation: Set $\Delta_r^0 = N$, $\mathcal{T}^0 = \{\{j\} : x_j = v(\{j\}), j \in N\}$ and $k = 1$;
while $|\Delta_r^{k-1}| < n$ **do**
  2. Find $\mathcal{T}^k = \underset{S \notin \text{span}(\Delta_r^{k-1})}{\text{argmax}} \; E(S, x)$;
  if $\Delta_r^{k-1} \cup \mathcal{T}^k$ *is* $\mathcal{T}^0$*-balanced* **then**
    3. Update $\Delta_r^k : \text{span}(\Delta_r^k) = \text{span}(\Delta_r^{k-1} \cup \mathcal{T}^k)$, $|\Delta_r^k| = \text{rank}(\Delta_r^k)$ and $k = k + 1$;
  else
    4. STOP and conclude that $x \neq \nu$;
  end
end
5. Conclude that $x = \nu$;

the improved Kohlberg Algorithm 10 and the new version (Algorithm 11) with compact representation are again highlighted in red. The update in Step 3 guarantees that the size of the representative set remains at most n, while spanning the same subspace of coalitions as the union of tight sets. Thus Lemmas 3.2 and 3.4 together guarantee that the outcome of the verification remains unchanged, while it also allows us to somewhat simplify the stopping condition.

4.2 Sequential LP model variants

We expect that the main driving force of computation time among the various types of sequential LP frameworks finding the nucleolus is the type of LP we need to solve,

and the number of iterations needed, Remark 2.7 connecting the two. In terms of the former, the dual sequence is preferred compared to the primal, based on the structure of the LPs: having small number of constraints and large number of variables lead to bases of compact size, that makes the problem more tractable for LP solvers. While in terms of the latter, the number of iterations needed, the primal sequence is preferred, as noted in Remark 2.7. This creates a classical trade-off between primal and dual sequential LP methods, that Solymosi (1993) recognise, and generally sides with preferring the dual sequence, despite the additional iterations possibly required. Based on the numerical results in the upcoming Subsection 5.3 one finds that in some cases solving the dual LPs is worthwhile even at the price of additional iterations, however, for certain games the difference in iterations grow so large that this turns around. Overall, we see a mixed picture, however, in this section our aim is to introduce variants of sequential LP methods that dissolve this trade-off completely.

Let us start by introducing an enhanced version of Algorithm 1 based on the primal sequence. Once we solved the LP ($\mathbf{P^{(k)}}$), we can not only form the tight set $\mathcal{T}^{(k)}(x^{(k)})$, $\widetilde{\mathcal{T}}^{(k)}(x^{(k)})$, it is safe to assume that we have access to a dual solution $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$, and thus due to Corollary 2.6 a part of the minimal tight set in the form of a dual non-tight set $\mathcal{NT}^{(k)}(u^{(k)}) \subseteq \mathcal{T}^{(k)*}$ and $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) \subseteq \widetilde{\mathcal{T}}^{(k)*}$. Therefore just by looking at the support of the dual solution, we already identified a part of the primal tight set that belongs to the minimal tight set.

Naturally we can exploit this to use Algorithm 5 only on $\mathcal{T}^{(k)}(x^{(k)}) \setminus \mathcal{NT}^{(k)}(u^{(k)})$, $\widetilde{\mathcal{T}}^{(k)}(x^{(k)}) \setminus \widetilde{\mathcal{NT}}^{(k)}(w^{(k)})$, requiring smaller auxiliary LPs (3.6) and less iterations in the dual subroutine. However, if we find that $\mathcal{NT}^{(k)}(u^{(k)}) = \mathcal{T}^{(k)}(x^{(k)})$, $\widetilde{\mathcal{NT}}^{(k)}(w^{(k)}) = \widetilde{\mathcal{T}}^{(k)}(x^{(k)})$, then by Proposition 2.5 we have found the minimal tight set and we do not have to run the subroutine at all. Furthermore, even when the latter does not hold, we can update $\Delta_r^{(k-1)}$ from $\mathcal{NT}^{(k)}(u^{(k)}) \cup \widetilde{\mathcal{NT}}^{(k)}(w^{(k)})$, and if we find $\operatorname{rank}(\Delta_r^{(k-1)} \cup \mathcal{NT}^{(k)}(u^{(k)}) \cup \widetilde{\mathcal{NT}}^{(k)}(w^{(k)})) = n$, then again we do not even have to run the subroutine, as the current point $x^{(k)}$ satisfying the stopping (rank) condition is the nucleolus. Algorithm 12 summarises the above introduced primal-dual algorithm.

Since the primal sequence already benefited from minimal number of iterations, and the primal-dual still requires to solve the less favourable ($\mathbf{P^{(k)}}$), we expect that Algorithm 12 is only going to be a minor improvement to the original primal sequence, especially in less number of subroutine iterations.

However, exploiting the same for the dual sequence, namely having access to a primal solution $x^{(k)} \in \mathcal{P}^{(k)*}$ once we solved the dual ($\mathbf{D^{(k)}}$), allows us to find the whole minimal tight set in every iteration of the dual algorithm. Once we solved ($\mathbf{D^{(k)}}$), we expand $\Delta_r^{(k-1)}$ from the support set of $u^{(k)}$ and $w^{(k)}$. Afterwards, using $x^{(k)}$ we form the tight set $\mathcal{T}^{(k)}(x^{(k)})$, but obviously it is enough to restrict our attention to active constraints

Algorithm 12: Primal-dual algorithm finding the nucleolus ν of cooperative game v with $I(v) \neq \emptyset$

1. Initialisation: $k = 1, \Delta_r^{(0)} = N, \Sigma^{(0)} = \mathcal{N}, \Gamma^{(0)} = \{\{i\} : i \in N\}, y_N^* = 0$;
while $\operatorname{rank}(\Delta_r^{(k-1)}) < n$ **do**
  2. Solve ($\mathbf{P^{(k)}}$), finding $(x^{(k)}, \epsilon^{(k)*}) \in \mathcal{P}^{(k)*}$ and $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$;
  3. Update $\Delta_r^{(k-1)}$: $\operatorname{span}\left(\Delta_r^{(k-1)}\right) = \operatorname{span}\left(\Delta_r^{(k-1)} \cup \mathcal{NT}^{(k)}(u^{(k)}) \cup \widetilde{\mathcal{NT}}^{(k)}(w^{(k)})\right)$,
  $\operatorname{rank}\left(\Delta_r^{(k-1)}\right) = |\Delta_r^{(k-1)}|$;
  if $\operatorname{rank}\left(\Delta_r^{(k-1)}\right) = n$ *or* $\mathcal{T}^{(k)}(x^{(k)}) \cup \widetilde{\mathcal{T}}^{(k)}(x^{(k)}) \subset \operatorname{span}\left(\Delta_r^{(k-1)}\right)$ **then**
    4. $\Delta_r^{(k)} = \Delta_r^{(k-1)}$, $k = k + 1$;
  else
    5. Find the minimal tight set $\mathcal{T}^{(k)*}$, $\widetilde{\mathcal{T}}^{(k)*}$ using Algorithm 5;
    6. Update $\Delta_r^{(k-1)}$: $\operatorname{span}\left(\Delta_r^{(k)}\right) = \operatorname{span}\left(\Delta_r^{(k-1)} \cup \mathcal{T}^{(k)*} \cup \widetilde{\mathcal{T}}^{(k)*}\right)$,
    $\operatorname{rank}\left(\Delta_r^{(k)}\right) = |\Delta_r^{(k)}|$, $k = k + 1$;
  end
  7. Find all $S \in \Sigma^{(k-1)} \setminus \operatorname{span}(\Delta_r^{(k-1)})$, $\{i\} \in \Gamma^{(k-1)} \setminus \operatorname{span}(\Delta_r^{(k-1)})$ to formulate ($\mathbf{P^{(k)}}$);
end
8. Output: ν is the unique solution of $x(S) = v(S) - y_S^* \; \forall S \in \Delta_r^{(k-1)}$;

in the primal outside of the new, expanded $\operatorname{span}(\Delta_r^{(k-1)})$. Then we can use Algorithm 5 to find the rest of the minimal tight set in $\mathcal{T}^{(k)}(x^{(k)}) \setminus \operatorname{span}(\Delta_r^{(k)})$.

Naturally, if the latter set is empty, or the rank condition is satisfied, we again do not even have to call the subroutine. This way we get a *dual-primal* approach (Algorithm 13), that is guaranteed to have maximal rank increase of $\Delta_r^{(k-1)}$ in every iteration by finding the whole minimal tight set, while still solving the computationally easier to handle dual LPs.

In the upcoming Section 5.3 we assess the performance of the above introduced algorithms against the classical sequential LP methods, and observe the enhancing effect of reducing the number of iterations for the dual sequence exploiting the primal solution. Then in the following Section 5.4 we expand this comparison to the active-set methods of Chapter 3. Moreover, in Subsection 5.5 we introduce variants of the aforementioned algorithms with increased number of iterations, while still possibly decreasing computation time.

Algorithm 13: Dual-primal algorithm finding the nucleolus ν of cooperative game v with $I(v) \neq \emptyset$

1. Initialisation: $k = 1, \Delta_r^{(0)} = N, \Sigma^{(0)} = \mathcal{N}, \Gamma^{(0)} = \{\{i\} : i \in N\}, y_N^* = 0$;

while $\text{rank}(\Delta_r^{(k-1)}) < n$ **do**

2. Solve ($\mathbf{D^{(k)}}$), finding $(u^{(k)}, w^{(k)}, z^{(k)}) \in \mathcal{D}^{(k)*}$ and $(x^{(k)}, \epsilon^{(k)*}) \in \mathcal{P}^{(k)*}$;

3. Update $\Delta_r^{(k-1)}$: $\text{span}\left(\Delta_r^{(k-1)}\right) = \text{span}\left(\Delta_r^{(k-1)} \cup \mathcal{NT}^{(k)}(u^{(k)}) \cup \widetilde{\mathcal{NT}}^{(k)}(w^{(k)})\right)$, $\text{rank}\left(\Delta_r^{(k-1)}\right) = |\Delta_r^{(k-1)}|$;

if $\text{rank}\left(\Delta_r^{(k-1)}\right) = n$ *or* $\mathcal{T}^{(k)}(x^{(k)}) \cup \widetilde{\mathcal{T}}^{(k)}(x^{(k)}) \subset \text{span}\left(\Delta_r^{(k-1)}\right)$ **then**

4. $\Delta_r^{(k)} = \Delta_r^{(k-1)}$, $k = k + 1$;

else

5. Find the minimal tight set $\mathcal{T}^{(k)*}$, $\widetilde{\mathcal{T}}^{(k)*}$ using Algorithm 5;

6. Update $\Delta_r^{(k-1)}$: $\text{span}\left(\Delta_r^{(k)}\right) = \text{span}\left(\Delta_r^{(k-1)} \cup \mathcal{T}^{(k)*} \cup \widetilde{\mathcal{T}}^{(k)*}\right)$, $\text{rank}\left(\Delta_r^{(k)}\right) = |\Delta_r^{(k)}|$, $k = k + 1$;

end

7. Find all $S \in \Sigma^{(k-1)} \setminus \text{span}(\Delta_r^{(k-1)})$, $\{i\} \in \Gamma^{(k-1)} \setminus \text{span}(\Delta_r^{(k-1)})$ to formulate ($\mathbf{D^{(k)}}$);

end

8. Output: ν is the unique solution of $x(S) = v(S) - y_S^* \ \forall S \in \Delta_r^{(k-1)}$;

Chapter 5

Numerical results

In this chapter we present extensive amount of numerical results, where we thoroughly test the methods we introduce in Chapters 2, 3 and 4. Therefore we devote the upcoming Section 5.1 to introduce the computational environment we use to conduct these tests, followed by numerical results related to the verification of the nucleolus by the Kohlberg criterion. In Section 5.3 we compare the sequential LP variants introduced in Sections 2.1, 2.2 and 4.2, while in Section 5.4 we expand the numerical comparison of these to the state-of-the-art active set methods introduced in Section 3.1.5, and covered in Section 3.2. Finally we introduce a new conceptual treatment to handling *coalition-subspaces* in Section 5.5, eventually unfolding a new aspect of the classical primal-dual trade-off.

5.1 Implementation environment

We present computational results comparing Solymosi (1993)'s primal (SP) and dual sequences (SD), and introduce two further variants of these in Section 4.2. Three algorithms presented in Section 4.1, tested in Section 5.2 are focusing on verifying whether a given imputation is the nucleolus solution or not. Finally we show results comparing Algorithm 7 introduced in Section 3.1 to the sequential LP methods as well as Derks and Kuipers (1997)'s state-of-the-art active set method.

Thus we work with 9 different algorithms computing or verifying the nucleolus of general cooperative games. Furthermore, in Section 5.5 we introduce a conceptually new setting to these algorithms, hence we conduct the same computational experiments for the 6 constructive algorithms under this new approach, providing further analysis and a broader comparison.

We test these algorithms on 5 different types of games, with player set sizes varying from $n = 5$ to $n = 30$. For smaller (larger) games, that is $n \leq 25$ ($n > 25$) we generate 50 (10) instances for each size and type (except for type V), and report average

computational times, number of iterations, number of pivots and number of subroutine iterations required.

All algorithms were implemented in C++ and computations were carried out on a desktop PC with Intel Core i5-2500 3.30 GHz CPU and 16 Gb RAM[1]. All the LPs involved are solved with CPLEX 12.7.1's primal simplex method (with default settings[2]). Time limitations were set to 12 hours for $n \leq 25$, 15 hours for $n \leq 28$, and 18 hours for $n > 28$ for each game. All of the codes (along with the test instances) used to produce these results are available for free, open-source access at the GitHub repository Benedek (2018).

5.1.1 Testgames

5.1.1.1 Type I and II games

Type I and II games both appear in Potters et al. (1996) and Reijnierse (1995). The characteristic function for type I is given by

$$v(S) = \begin{cases} 0, & \text{if } |S| = 1, \\ \text{random integer between 1 and } 100|S|, & \text{if } 2 \leq |S| < n, \\ \text{random integer between } 100(n-2) \text{ and } 100n, & \text{if } S = N. \end{cases}$$

while type II games are generated as

$$v(S) = \begin{cases} 0, & \text{if } |S| = 1, \\ \text{random integer between 1 and } 50n, & \text{otherwise.} \end{cases}$$

The overall features of these games are requiring relatively low number of iterations, especially type I games, and having very modest distinguishing power between primal and dual methods in terms of number of iterations required.

5.1.1.2 Type III games

Derks and Kuipers (1997) were interested in games where the number of iterations grows more or less linearly with the number of players, and so they introduced type III games

[1] In this configuration, the time-efficient implementation of the algorithms run out of memory at $n = 29$ while processing initialisation, therefore we used a memory-efficient implementation instead for $n > 28$.

[2] In the case of *SP*, *SD*, and the other sequential LP methods, the average number of pivots reflect the values of CPLEX parameter *iterations* reported in the output. In order to obtain realistic pivot numbers, preprocessing was turned off (which did not change the overall computation time significantly).

as

$$v(S) = \begin{cases} 0, & \text{if } |S| < n-2, \\ 1 \text{ with probability } 0.9, & \text{if } n-2 \leq |S| < n, \\ 1, & \text{if } S = N. \end{cases}$$

The effort of the authors turned out to be successful, but only in terms of the dual methods. While dual methods require significant work, for primal methods these games turned out to be trivial. Hence these games generally over-estimate the effect of Remark 2.7, Derks and Kuipers (1997) offering a sort of worst-case scenario for their own purely dual method.

5.1.1.3 Type IV games

In order to test the methods on games, which distinguish between the number of iterations required by primal and dual methods more realistically, that is, games that are *somewhere between types I-II and III*, we introduce type IV games as

$$v(\mathcal{S}) = \begin{cases} 0, & \text{if } |S| = 1, \\ \text{random integer between 1 and } n, & \text{otherwise.} \end{cases}$$

Note that while the only difference between type II and type IV games is the 50 multiplier, it is far from $v = 50w$ with v being type II and w being type IV, as otherwise their nucleoli would coincide, or at least due to the randomness, asymptotically behave the same. Instead, the main difference is that the smaller spread in the integer coalitional values changes the structure of the tight sets in a way that produces a moderate difference between the number of iterations required by primal and dual methods.

5.1.1.4 Type V: UN Security Council game

We ourselves found it worthwhile to study the bottleneck of our algorithm, hence attempted to find games that our proposed method struggles with. The efficiency of Algorithm 5, the balancedness subroutine of Algorithm 7 depends on the size of the tight set, so we now look for games with extremely large tight sets.

For that purpose we adopt the United Nations (UN) Security Council voting mechanism into weighted voting games (WVGs) with arbitrary size, where there are 5 *big* (veto) players and the rest (originally 10) are *small*. Weights are calibrated such that all 5 veto players' agreement needed to pass a vote, while still approx. half of the small players also needed. The weights are for $n \geq 7$: $w_j = \lfloor \frac{n-3}{2} \rfloor$ for $j = 1, \ldots, 5$ and $w_j = 1$ otherwise,

while the quota is $q = 4w_1 + n - 4$ and the game is given by

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

It should be noted that finding the nucleoli of these games, them being *balanced* WVGs (i.e. with non-empty core) is trivial, that is, one can easily find analytically that the 5 veto players share the total payoff of 1 amongst themselves in an egalitarian way (0.2 for each veto player), while all the small players get 0 (cf. Elkind et al. (2007)). Therefore anyone interested in finding the nucleolus of such a game would never turn to any of the aforementioned algorithms. However, since these games are of a very peculiar nature from an algorithmic perspective, they carry *theoretical interest* from a computational point of view.

As these WVGs are deterministic, type V games are exceptions for the average reported values of 50 (10) randomly generated games per player set size, instead we provide the exact values of single runs for each size.

We close this section by noting that all games through types I to V have non-empty imputation sets, therefore they are suitable to study for nucleolus computations. The starting imputation point for all 6 constructive approaches is the *equal surplus division (ESD)* point, specified as

$$x_i = v(\{i\}) + \frac{v(N) - \sum_{j \in N} v(\{j\})}{n} \quad \forall i \in N,$$

that is when each player i receives it's own value $v(\{i\})$ plus an equal share of the remainder of the grand coalition value $v(N)$. Since the Kohlberg algorithms of the following Section 4.1 are concerned with verification, rather than computing the nucleolus of a game, their starting points, the imputation in question, is part of the input, as specified in Subsection 5.2 (and in Appendix A).

5.2 Numerical results of the Kohlberg algorithms

In a thorough test of a verification algorithm, one would like to make sure that, first of all, in each and every case the nucleolus is accepted while all the other imputations are rejected. Moreover, we would naturally expect to reject imputations having absolutely no relation to the nucleolus quicker, than those which are *closer* to it, in terms of sharing certain attributes. Accordingly, for the sake of completeness, the various form of Kohlberg algorithms are tested with 4 solution points, including the nucleolus, a random imputation, a point in the least core and a point in the *least-least core* (an element of the least core with $\mathcal{T}^0$-balanced ($\mathcal{T}^1 \cup \mathcal{T}^2$)).

Table 5.1: Original and improved Kohlberg algorithms on type I games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.014	0.0016	**0.0014**	25.6	2.6	2.6	43.3	3.3	2.6	2.6
10	2.5	**0.0034**	0.0042	964.1	2.3	2.3	1279.8	3	2.3	2.4
15	OoT	**0.068**	0.068	OoT	1.9	1.9	OoT	2.5	1.9	1.9
20	-	3.3	**3.2**	-	1.7	1.7	-	1.9	1.7	1.8
25	-	161	**159**	-	1.9	1.9	-	2.4	1.9	1.9
26	-	257	**250**	-	1.6	1.6	-	1.7	1.6	1.6
27	-	650	**631**	-	1.9	1.9	-	2.4	1.9	1.9
28	-	2827	**2591**	-	1.9	1.9	-	2.2	1.9	1.9
29	-	**2087**	2145	-	1.7	1.7	-	2.0	1.7	1.7
30	-	**3248**	3323	-	1.4	1.4	-	1.7	1.4	72842.4

Table 5.2: Original and improved Kohlberg algorithms on type II games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.014	**0.0016**	**0.0016**	26.1	2.8	2.6	48.3	3.9	2.6	1.6
10	6.9	**0.0033**	0.0036	951	2.6	2.5	2023.5	3.6	2.6	2.1
15	OoT	0.11	**0.11**	OoT	2.5	2.5	OoT	3.6	2.6	2.3
20	-	5.3	**5.2**	-	2.5	2.5	-	3.4	2.5	2.5
25	-	295	**289**	-	3.6	3.6	-	5.8	3.6	3.6
26	-	576	**558**	-	3.1	3.1	-	3.8	3.1	3.1
27	-	1061	**1041**	-	3	3	-	4.2	3.1	3.1
28	-	5672	**5271**	-	4.8	4.8	-	7.9	4.8	4.9
29	-	**7253**	7547	-	4.4	4.4	-	7.3	4.4	4.4
30	-	**12766**	13325	-	3.8	3.8	-	5	3.8	3.9

For brevity, here we only present results for the solution being the nucleolus. Results for the other three solutions are presented in Appendix A. The original and the improved Kohlberg algorithms 9 and 10 are denoted with *Kohlberg* and *IKA* respectively, while Algorithm 11 including the compact representation is denoted with *IKAcr*. Among the aforementioned features, since number of pivots are not applicable for these algorithms, instead we report an additional number of coalitions saved from storage by the compact representation in *IKAcr*.

In terms of the verifying Kohlberg algorithms we can start with the general observation that the classical Kohlberg algorithm is only usable up to a certain size. From Tables 5.1-5.4 we find that games of size $n = 15$ provide already a challenge that Algorithm 9 can not tackle, as it runs out of time.

For the remaining two algorithms *IKA* and *IKAcr*, the differences in performance do not seem to be significant. The advantage of having a compact representation only affects a small number of coalitions most of the time, with the notable exception of one type I game with 30 players. This is a random game with quite substantially larger tight set $\mathcal{T}^1(x)$ than the other games considered. However, as both *IKA* and *IKAcr* terminate after performing 1 iteration, the advantage of a compact representation is

Table 5.3: Original and improved Kohlberg algorithms on type III games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0021	**0.0006**	0.0011	5.2	1	1	6.1	2	2	5
10	0.43	**0.0012**	0.0028	10.7	1	1	12.7	2.6	2.9	30.2
15	OoT	0.026	**0.026**	OoT	1	1	OoT	2.2	2.5	80.9
20	-	**1.0**	1.0	-	1	1	-	2	2.7	152.2
25	-	**38.7**	39.7	-	1	1	-	2.1	2.9	245.2
26	-	81.8	**81.6**	-	1	1	-	2.2	2.6	270.3
27	-	**168**	170	-	1	1	-	2.3	2.7	291.5
28	-	1092	**1076**	-	1	1	-	2.2	2.6	315.7
29	-	**217**	226	-	1	1	-	2.2	2.7	336
30	-	**447**	449	-	1	1	-	2.2	2.8	358.5

Table 5.4: Original and improved Kohlberg algorithms on type IV games and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.010	**0.0009**	0.0009	14.2	1.6	1.6	24.5	2.6	1.8	2.8
10	2.2	**0.0027**	0.0028	285	2	2	608	3.4	2.4	3.7
15	OoT	**0.079**	0.079	OoT	2	2	OoT	3.6	2.8	9.4
20	-	3.7	**3.6**	-	2.1	2.1	-	4.3	4	18.6
25	-	**220**	225	-	2.9	2.9	-	6.1	3.9	15.6
26	-	590	**573**	-	3.4	3.4	-	6.2	5	16.1
27	-	1305	**1269**	-	3.6	3.6	-	7.1	4.7	18.1
28	-	4576	**4344**	-	3.7	3.7	-	7.4	5.1	18.2
29	-	**5464**	5658	-	3.5	3.5	-	7.2	5.4	25.2
30	-	**9647**	9842	-	2.8	2.8	-	5.9	4.4	29.1

not realised during any subsequent iterations. On the contrary, the small additional workload necessary for finding this compact representation appears to provide a slight disadvantage in computation time.

Both *IKA* and *IKAcr* solve type III games extremely efficiently, making them hard to distinguish from each other, while solution for type IV games show a similar behaviour as games of types I and II. The non-monotonicity in computation time occurring in between 28- and 29-player type I, and type III games is due to the two different implementations, a so-called time-efficient and memory-efficient version. It is even more apparent for type III games, because of the trivial nature of these games for these algorithms, while the same phenomena, arguably with much less effect, occurs for type I games. That is, the time-efficient implementation is *actually not efficient in computational time*, since it *wastes* a lot of time at initialisation compared to the memory-efficient version. Nevertheless, for the sake of consistency, we stick to switching implementation only when it is definitely needed from the available memory point of view, at $n = 29$.

As we noted in the introduction of the UN Security Council game in Subsection 5.1.1.4, since these games produce extremely large tight sets, where a compact representation is vital, we expect them to carry sufficient distinguishing power between *IKA* and *IKAcr*.

Table 5.5: Original and improved Kohlberg algorithms on UN Security Council game and nucleolus solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
7	0.003	**0.001**	**0.001**	6	2	2	6	3	3	19
10	0.008	**0.003**	0.004	6	2	2	7	4	4	199
15	0.21	0.075	**0.073**	6	2	2	6	3	3	6774
20	OoT	3.2	**3.1**	OoT	2	2	OoT	4	4	212983
25	-	**134**	143	-	2	2	-	3	3	6908114
26	-	847	**555**	-	2	2	-	4	4	13631479
27	-	**4381**	5332	-	2	2	-	3	3	27615684
28	-	OoM	OoT	-	OoM	OoT	-	OoM	OoT	OoT

In Table 5.5 we can clearly see the effect of the compact representation in *IKAcr*, however, it does not always translate to faster computations. This is a reasonable trade-off between computational time and memory usage, eventually leading to *IKA* running out of memory for $n = 28$, whereas *IKAcr* running out of time instead.

While we find that both methods are unable to verify the nucleolus for $n = 28$, the reason, *IKA* running out of memory while *IKAcr* running out of time, differs very much in quality. It means that with our available computational resources, and some additional time, we can expect *IKAcr* to complete the task. For this we find evidence in Table A.12, verifying not the nucleolus, but an element of the least core for the UN Security Council game. We find that *IKA* already runs out of memory for an element of the least core, while *IKAcr* is capable of completing the rejection in the second iteration. The difference is of no wonder, as we can note the very high usage of our compact representation of extremely large tight sets.

5.3 Numerical results of the sequential LP methods

We evaluate the performance of newly introduced variants Algorithms 12 (*PD*) and 13 (*DP*) compared to Solymosi (1993)'s classical sequential LP methods (*SP* and *SD*).

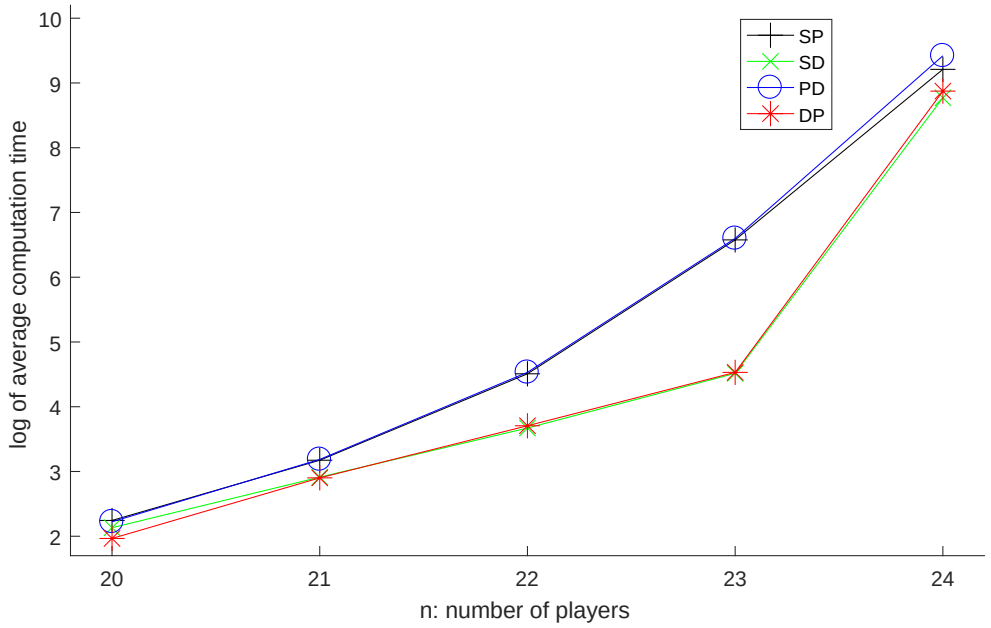
In general one can observe the effect of Remark 2.7: the number of iterations are the same and minimal for *SP*, *PD* and *DP*, while it can be slightly larger for purely dual method *SD*, but still below $n - 1$ in all Tables 5.6-5.10, as one would naturally expect so. Both actually holds not only for averages, but for every single instance.

From Table 5.6 and it is already apparent that results meet our first expectation: when it comes to the sequential LP methods the main driving source of computation time, and overall solvability for that matter, is the type of LP we are dealing with. Solving ($\mathbf{D^{(k)}}$) seems substantially easier as solving ($\mathbf{P^{(k)}}$), since the minimum computational times (highlighted with bold blue font) belong to methods solving dual LPs for all size $n < 25$, and these methods produce significantly faster completion overall.

Table 5.6: Computing the nucleolus of type I games

n	Time				Iterations				Pivots				Subrout.		
	SP	SD	PD	DP	SP	SD	PD	DP	SP	SD	PD	DP	SP	PD	DP
5	0.003	**0.001**	0.011	0.002	2.52	2.52	2.52	2.52	9.9	14.7	7.9	14.7	4.04	1.52	1.52
10	0.007	0.016	0.01	**0.006**	2.02	2.06	2.02	2.02	16.7	63.6	14.8	62.4	3.06	1.06	1.06
15	0.19	0.17	0.19	**0.14**	1.5	1.52	1.5	1.5	23.4	153	22.5	153	2	0.52	0.52
20	9.43	8.43	9.24	**7.14**	1.46	1.46	1.46	1.46	32.5	330	31.6	330	1.92	0.46	0.46
21	23.9	18.4	24.2	**18.2**	1.7	1.74	1.7	1.7	36.8	358	35.1	357	2.42	0.74	0.74
22	90.8	**39.3**	92.9	40.7	1.64	1.64	1.64	1.64	154	401	153	401	2.28	0.64	0.64
23	718	**91.4**	735	92.8	1.6	1.6	1.6	1.6	240	498	239	498	2.2	0.62	0.62
24	9996	**6443**	12329	7144	1.6	1.6	1.6	1.6	259	519	258	519	2.2	0.6	0.6
25	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoM	OoT

Figure 5.1: Computing the nucleolus of type I games with sequential LP methods



This can be seen on Figure 5.1 as well, showing the computational times on a logarithmic scale. Sequential LP methods handle a simplex tableau of $\mathcal{O}(n) \times \mathcal{O}(2^n)$ size, and dual methods seem to scale somewhat more stable as the number of players increases, however, there is a change appearing reaching close to the limitations of these algorithms.

Evaluating the dual methods, notice that the gap between iteration numbers for *SP-PD* and *SD-DP* is basically negligible. That leads to *SD* slightly outperforming *DP* after a while, since the additional work with the primal solution and running subroutines is rarely rewarded with decrease in the number of iterations.

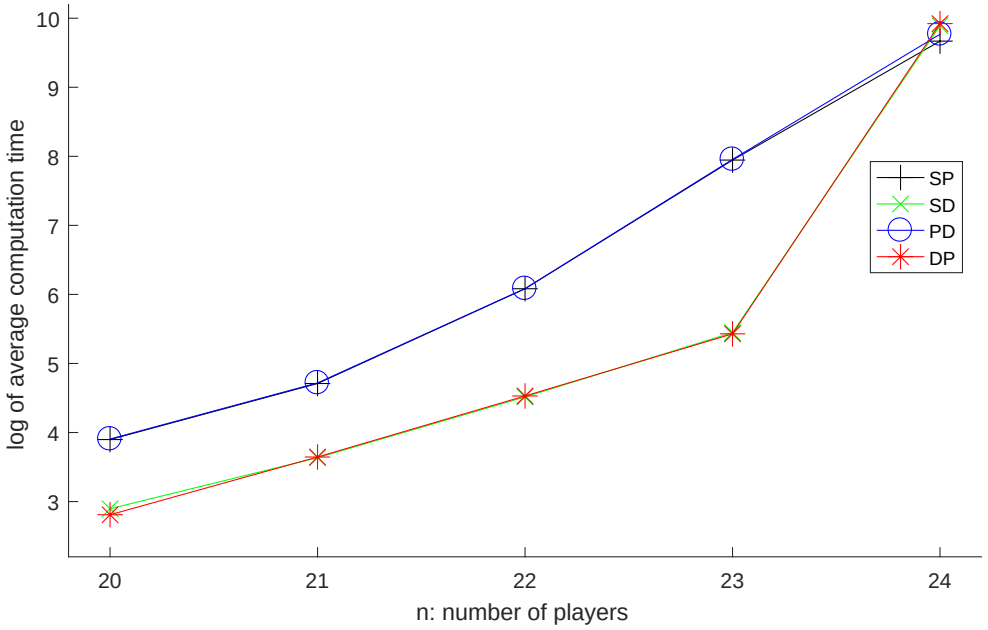
Turning to type II games and the results in Table 5.7, we notice a similar pattern, mainly due to the iteration gap between primal and dual methods being similarly narrow as in type I games. However, one noticeable difference is that on the largest games these sequential LP methods can still handle, i.e. $n = 24$, *SP* produces the best computation

Table 5.7: Computing the nucleolus of type II games

n	Time				Iterations				Pivots				Subrout.		
	SP	SD	PD	DP	SP	SD	PD	DP	SP	SD	PD	DP	SP	PD	DP
5	0.003	**0.001**	0.003	0.002	2.56	2.6	2.56	2.56	9.88	14.7	8.26	14.4	3.86	1.56	1.58
10	0.011	0.007	0.011	**0.007**	2.66	2.72	2.66	2.66	17.8	53.3	15	51.9	4.32	1.7	1.72
15	0.81	0.3	0.81	**0.26**	2.7	2.74	2.7	2.7	26.1	114	22.6	113	4.4	1.78	1.78
20	49.3	18.1	49.7	**16.6**	3	3.02	3	3	36.7	207	31.6	207	5.02	2.06	2.06
21	111	**37.9**	112	38.3	3.24	3.24	3.24	3.24	45	277	39.2	277	5.5	2.24	2.24
22	438	**91.4**	439	92.7	3.3	3.32	3.3	3.3	76.7	328	70.8	323	5.6	2.34	2.32
23	2823	232	2851	**228**	2.96	2.98	2.96	2.96	136	290	131	283	4.92	1.96	1.98
24	**15809**	19795	17469	20373	2.96	2.96	2.96	2.96	111	343	103	350	4.94	2	2
25	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoM	OoT

time, ahead of *PD*, and the dual methods perform worse, while requiring exactly the same number of iterations. It is a warning sign, that while we managed to produce a dual method with minimal number of iterations, the trade-off between primal and dual methods might have another significant aspect to it, that we have not identified. We return to this in Section 5.5.

Figure 5.2: Computing the nucleolus of type II games with sequential LP methods



When it comes to assessing *PD*, we first and foremost compare it to *SP*. As we expected, exploiting the dual solution likely will not reduce computation time significantly, and the number of iterations required are the same by design. However, one can notice a substantial difference in the number of subroutine iterations and a minor one in pivots required, which in certain cases can translate to a slight advantage in computation time.

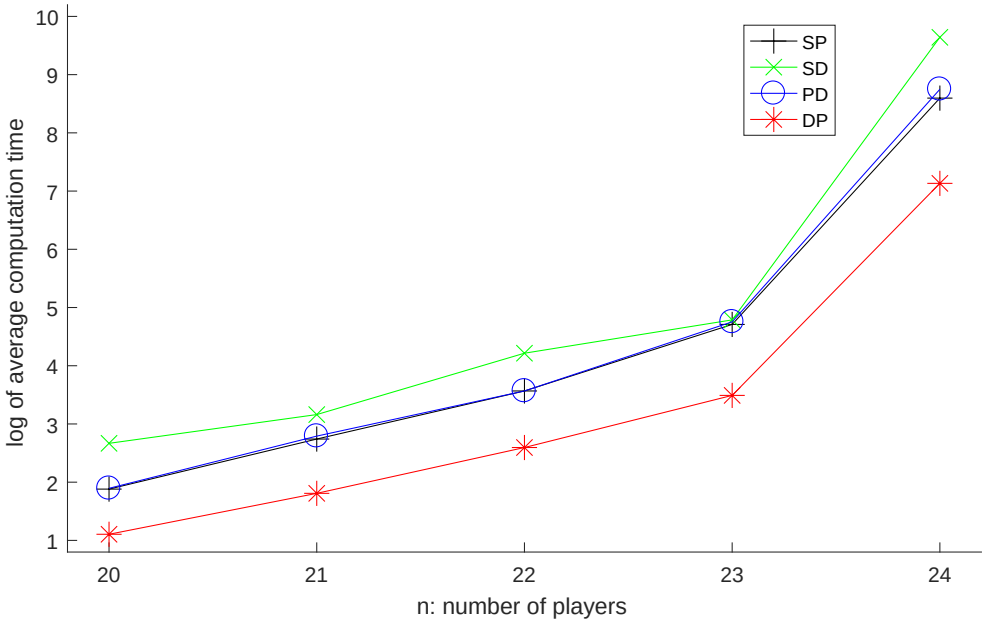
Type III games are due to Derks and Kuipers (1997), they designed it in a way that it challenges (purely) dual methods, with the increased number of iterations required.

Table 5.8: Computing the nucleolus of type III games

n	Time				Iterations				Pivots				Subrout.		
	SP	SD	PD	DP	SP	SD	PD	DP	SP	SD	PD	DP	SP	PD	DP
5	0.002	**0.001**	**0.001**	**0.001**	1.02	1.9	1.02	1.02	7.5	8.04	7.5	4.22	1.96	0.42	0.9
10	**0.004**	0.007	**0.004**	0.005	1	2.9	1	1	11.9	57.5	11.9	17.1	2.84	1.9	2.3
15	0.13	0.21	0.13	**0.071**	1	2.34	1	1	17	95.4	17	25.5	2.82	2.96	2.52
20	6.56	14.4	6.65	**3.02**	1	2.86	1	1	24.3	707	24.3	37.6	2.74	3.08	3.22
21	15.5	23.6	16.3	**6.1**	1	2.34	1	1	41.8	469	41.8	37.4	2.78	3.02	3.42
22	35.4	67.7	35.4	**13.4**	1	2.66	1	1	44.1	885	44.1	41.2	3.12	2.9	3.42
23	111	120	116	**32.8**	1	2.18	1	1	44.9	782	44.9	41.6	3.04	3.08	3.7
24	5414	15380	6302	**1252**	1	2.72	1	1	46.6	1589	46.6	45	3.1	3.24	3.88
25	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoT	OoM	OoM	OoT

On the other hand, these games are basically trivial for primal methods, as well as *DP*. As a result, outside of a few exceptions, we again do not experience notable difference between *SP* and *PD*. However, the advantage of *DP* compared to *SD*, or any of the other sequential LP methods for that matter, is crystal clear in the case of type III games.

Figure 5.3: Computing the nucleolus of type III games with sequential LP methods

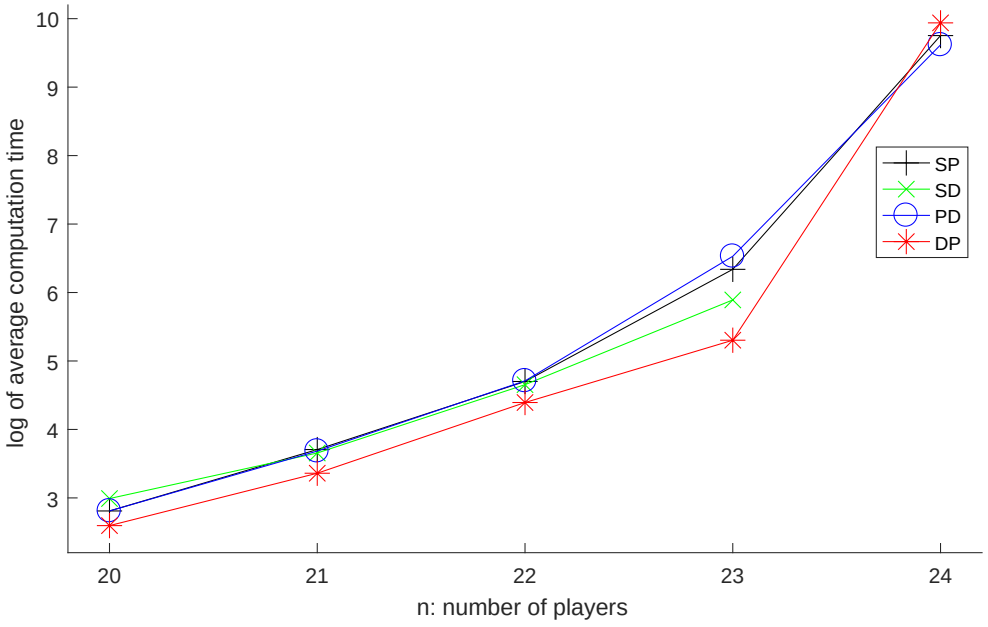


Obviously type III games are designed to produce these kind of results. Therefore we introduce type IV games, offering moderate distinction in the number of iterations between the purely dual and the other methods. Outside of subroutine iterations and pivots, *SP* and *PD* again return similar results. For the case of *DP*, we see that even a moderate difference in the number of iterations is enough to distinguish it from *SD* in the games we are capable to solve. When the number of iterations are the same, *SD* and *DP* also return the same kind of results, but as soon as a noticeable difference appears in the number of iterations, *DP* clearly outperforms *SD*, along with the other sequential LP methods.

Table 5.9: Computing the nucleolus of type IV games

n	Time				Iterations				Pivots				Subrout.		
	SP	SD	PD	DP	SP	SD	PD	DP	SP	SD	PD	DP	SP	PD	DP
5	0.002	**0.001**	0.002	0.002	1.78	2.42	1.8	1.8	7.98	11.7	7.28	8.46	2.94	1.18	1.38
10	0.007	0.006	0.007	**0.005**	1.88	2.4	1.88	1.88	15.9	42.6	13.8	36.4	3.34	1.88	1.78
15	0.26	0.27	0.26	**0.19**	2.06	2.52	2.06	2.06	23.9	90.1	21.2	80.8	3.8	2.32	2.22
20	16.6	19.9	16.6	**13.4**	2.6	3.46	2.6	2.6	36.3	245	29.7	192	5.08	3.38	3.38
21	40.7	38.7	39.8	**28.8**	2.6	3.38	2.6	2.6	48.2	231	41.7	192	4.94	3.3	3.26
22	110	105	111	**80.9**	3.02	3.88	3.02	3.02	65.8	331	57.1	250	5.98	3.84	3.86
23	566	362	685	**201**	2.72	3.76	2.72	2.72	96.2	362	88.3	259	5.42	3.52	3.7
24	17201	OoT	**15016**	20713	2.78	OoT	2.78	2.78	97.9	OoT	89.5	356	5.66	3.76	4
25	OoM	-	OoM	OoT	OoM	-	OoM	OoT	OoM	-	OoM	OoT	OoM	OoM	OoT

Figure 5.4: Computing the nucleolus of type IV games with sequential LP methods



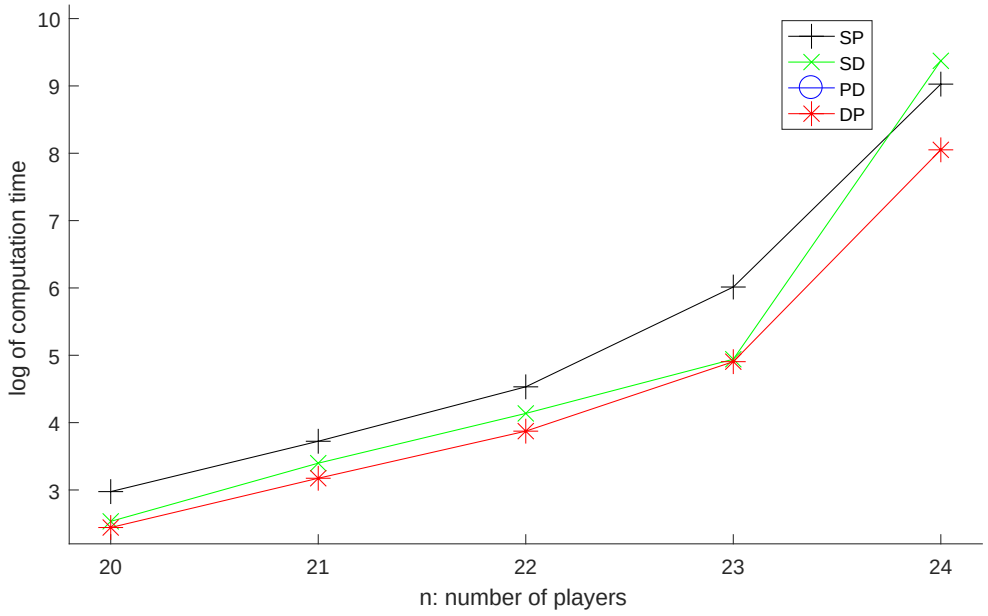
Following Derks and Kuipers (1997), we also introduce games to challenge our proposed algorithm, and in fact every primal method. Type V games are generalisations of the UN Security Council voting mechanism, converted into arbitrarily large ($n \geq 7$) voting games with non-empty core. As a result, finding the nucleoli of such games are straightforward (cf. Elkind et al. (2007)). Nevertheless these games are computationally interesting, because one can experience extremely large tight sets during the computation of the nucleolus using the methods discussed.

Comparing the different sequential LP methods, one finds that they all break down at the same level ($n = 25$), except for the somewhat surprising struggle of PDat $n = 20$, especially that it performs similarly to SP up to that point, and the former is at least comparable to SD and DP all the way through.

Table 5.10: Computing the nucleolus of type V games

n	Time				Iterations				Pivots				Subrout.		
	SP	SD	PD	DP	SP	SD	PD	DP	SP	SD	PD	DP	SP	PD	DP
7	0.006	0.04	**0.004**	0.05	2	3	2	2	12	22	11	16	4	2	2
10	0.013	0.21	**0.01**	0.63	2	3	2	2	16	47	17	28	6	3	3
15	0.42	0.41	0.68	**0.24**	2	5	2	2	21	144	22	26	4	5	4
20	19.6	12.6	OoT	**11.5**	2	3	OoT	2	26	156	OoT	17	5	OoT	5
21	41.4	29.9	-	**23.9**	2	4	-	2	35	63	-	17	5	-	4
22	92.9	62.6	-	**48.1**	2	4	-	2	39	82	-	22	5	-	4
23	409	140	-	**135**	2	4	-	2	42	52	-	21	5	-	5
24	8323	11762	-	**3137**	2	4	-	2	45	94	-	10	5	-	4
25	OoM	OoT	-	OoT	OoM	OoT	-	OoT	OoM	OoT	-	OoT	OoM	-	OoT

Figure 5.5: Computing the nucleolus of type V games with sequential LP methods



It should be noted that testing these methods on these games carries mostly only theoretical interest due to the triviality of the task of finding the nucleoli of these games. Nevertheless, for $20 \leq n < 25$ *DP* is the fastest among the 4 sequential LP variant. Thus we again find that among the methods that explicitly solve the classical sequence of LPs, *DP* mostly outperforms the others in computation times, and in general performs at worst comparable to the other 3 sequential LP methods.

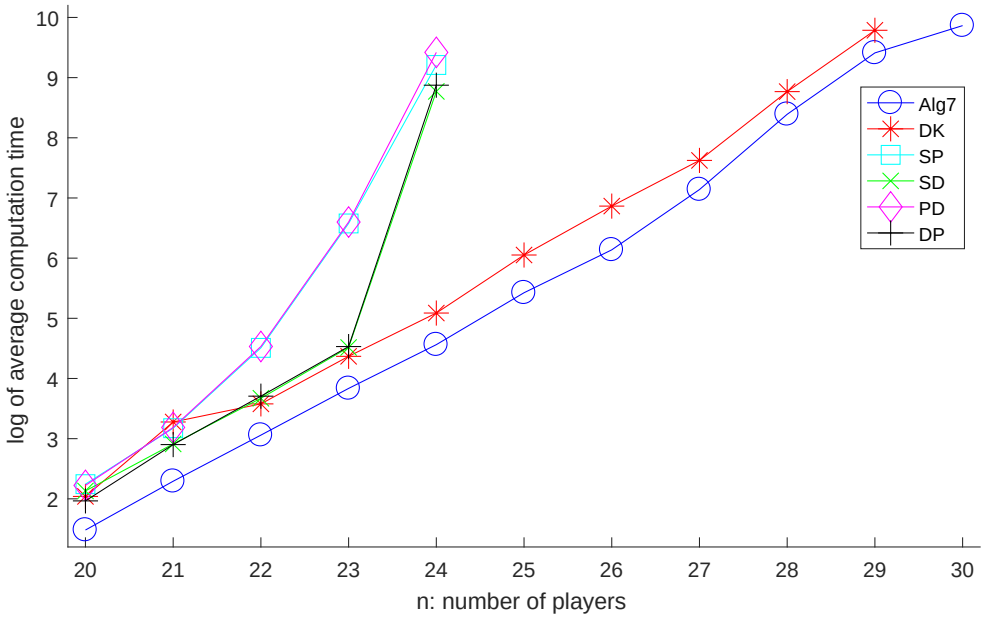
5.4 Numerical results of Algorithm 7

In this section we compare Algorithm 7 (*Alg7*), the active-set method we introduced in Section 3.1. Our first and foremost comparison is to Derks and Kuipers (1997)'s method (*DK*), because we believe it is the state-of-the-art method when it comes to

computing the (pre)nucleolus of any given cooperative game. However, for sake of complete comparison, we include the results of the sequential LP methods as well.

The first thing we notice, which is generally true throughout types I to IV, is that *Alg7* is the only algorithm capable of solving 30-player instances, even on a regular desktop PC with the given time restrictions. While sequential LP methods can not handle even 25-player games, even *DK* is incapable of computing the nucleolus of a 30-player instance within the time limit of 18 hours, irrespective of the type of the game, as Tables 5.11-5.13 show.

Figure 5.6: Computing the nucleolus of type I games



Focusing on type I games first in Table 5.11, we are not surprised that *Alg7* requires minimal number of iterations being a primal method in terms of guaranteed to update with the whole minimal tight set in between iterations. Note however, that even though the gap between primal and dual methods, such as *DK*, in the number of iterations required is almost negligible, *Alg7* already clearly outperforms *DK* in computation time.

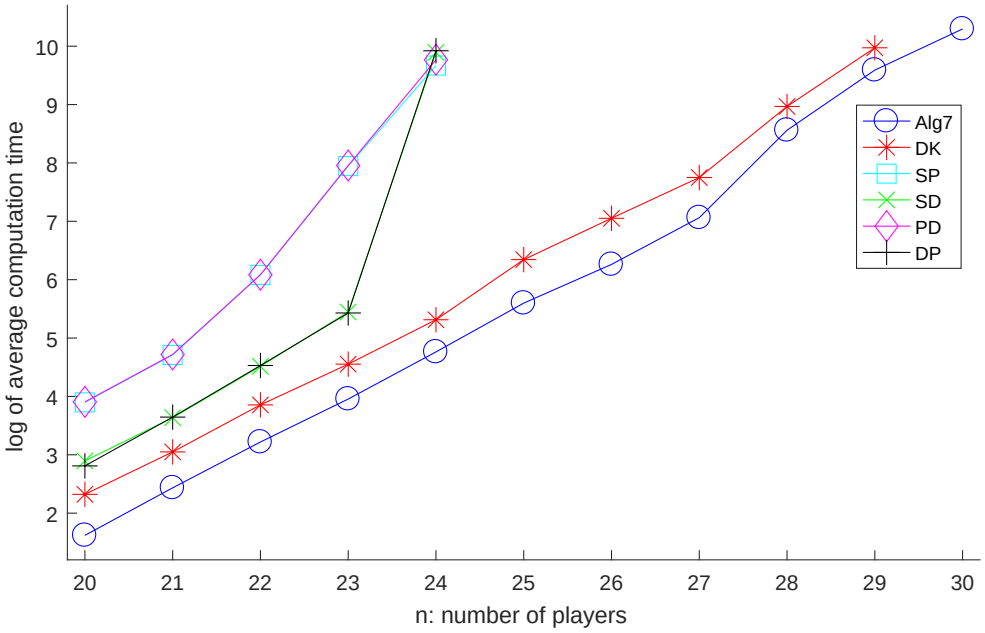
Type II games seem to paint a very similar picture to type I games, with the biggest difference is the additional required number of iterations, manifesting in larger computational times, but comparatively amongst the various methods the picture is very similar to type I.

Thus it is not surprising, that Figures 5.6 and 5.7, showing computational times on a logarithmic scale, are very much alike. On both figures it is apparent how much more stably active-set methods scale as the number of players increase. Naturally, both *Alg7* and *DK* still require exponential running time, but the advantage of not having to deal

Table 5.11: Computing the nucleolus of type I and II games

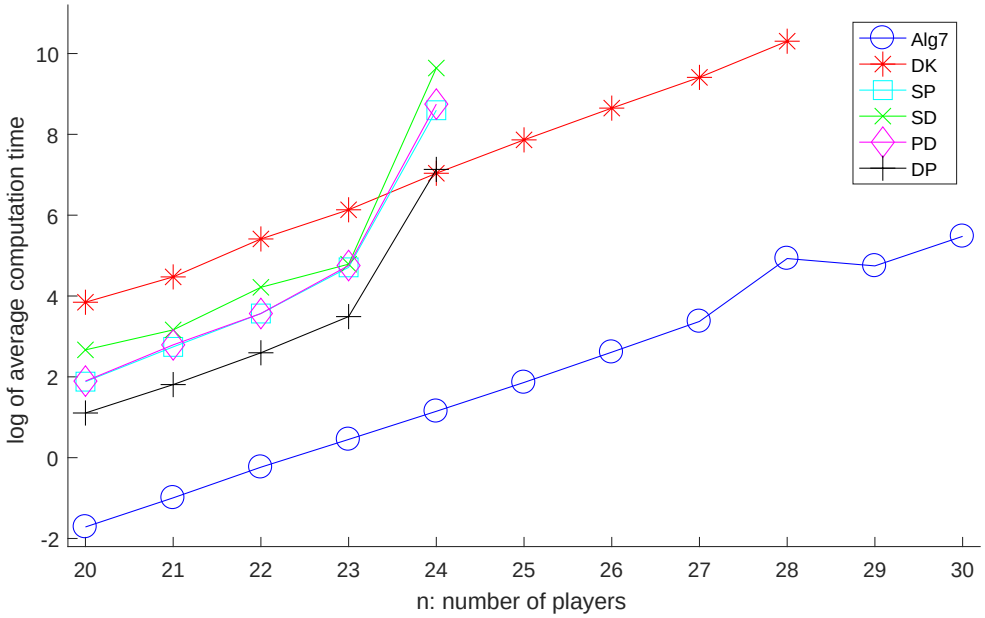
n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
5	0.005	0.006	0.003	**0.001**	0.011	0.002	2.52	2.52	2.52	2.52	2.52	2.52	5.42	6.06	9.86	14.7	7.92	14.7	7.98	4.04	1.52	1.52
10	0.011	0.014	0.007	0.016	0.01	**0.006**	2.02	2.04	2.02	2.06	2.02	2.02	11.8	14.1	16.7	63.6	14.8	62.4	14.3	3.06	1.06	1.06
15	**0.1**	0.17	0.19	0.17	0.19	0.14	1.5	1.52	1.5	1.52	1.5	1.5	19.2	25.9	23.4	153	22.5	153	21	2	0.52	0.52
20	**4.4**	7.69	9.43	8.43	9.24	7.14	1.46	1.46	1.46	1.46	1.46	1.46	28.9	40.6	32.5	330	31.6	330	31.4	1.92	0.46	0.46
21	**9.9**	26.5	23.9	18.4	24.2	18.2	1.7	1.84	1.7	1.74	1.7	1.7	30.9	113	36.8	358	35.1	357	33.8	2.42	0.74	0.74
22	**21.3**	35.8	90.8	39.3	92.9	40.7	1.64	1.64	1.64	1.64	1.64	1.64	33.7	49.6	154	401	153	401	36.6	2.28	0.64	0.64
23	**46.3**	78.9	718	91.4	735	92.8	1.6	1.6	1.6	1.6	1.6	1.6	34.8	52.8	240	498	239	498	37.3	2.2	0.62	0.62
24	**95.8**	162	9996	6443	12329	7144	1.6	1.6	1.6	1.6	1.6	1.6	35.6	53.5	259	519	258	519	38.1	2.2	0.6	0.6
25	**227**	425	OoM	OoT	OoM	OoT	1.78	1.8	OoM	OoT	OoM	OoT	42.1	56.5	OoM	OoT	OoM	OoT	47.6	OoM	OoM	OoT
26	**463**	958	-	-	-	-	1.6	1.6	-	-	-	-	40.6	71.8	-	-	-	-	43.3	-	-	-
27	**1261**	2047	-	-	-	-	1.9	1.9	-	-	-	-	57.1	70.4	-	-	-	-	69	-	-	-
28	**4406**	6421	-	-	-	-	1.9	1.9	-	-	-	-	48.5	82.1	-	-	-	-	53.7	-	-	-
29	**12220**	17796	-	-	-	-	1.7	1.7	-	-	-	-	58.1	78	-	-	-	-	66.4	-	-	-
30	**19232**	OoT	-	-	-	-	1.4	OoT	-	-	-	-	44.4	OoT	-	-	-	-	47.4	-	-	-
5	0.004	0.005	0.003	**0.001**	0.003	0.002	2.56	2.6	2.56	2.6	2.56	2.56	5.12	5.74	9.88	14.7	8.26	14.4	7.7	3.86	1.56	1.58
10	0.011	0.014	0.011	0.007	0.011	**0.007**	2.66	2.72	2.66	2.72	2.66	2.66	12	14.7	17.8	53.3	15	51.9	14.9	4.32	1.7	1.72
15	**0.11**	0.2	0.81	0.3	0.81	0.26	2.7	2.72	2.7	2.74	2.7	2.7	21.3	24.4	26.1	114	22.6	113	25.8	4.4	1.78	1.78
20	**5.05**	10.2	49.3	18.1	49.7	16.6	3	3.02	3	3.02	3	3	33.1	39.2	36.7	207	31.6	207	41.1	5.02	2.06	2.06
21	**11.4**	21.1	111	37.9	112	38.3	3.24	3.26	3.24	3.24	3.24	3.24	36.6	43	45	277	39.2	277	45.6	5.5	2.24	2.24
22	**25**	47.2	438	91.4	439	92.7	3.3	3.32	3.3	3.32	3.3	3.3	38.7	48.1	76.7	328	70.8	323	49.3	5.6	2.34	2.32
23	**52.2**	94.9	2823	232	2851	228	2.96	2.96	2.96	2.98	2.96	2.96	37.9	48.4	136	290	131	283	45.6	4.92	1.96	1.98
24	**117**	203	15809	19795	17469	20373	2.96	2.98	2.96	2.96	2.96	2.96	43.1	51	111	343	106	350	54.1	4.94	2	2
25	**270**	569	OoM	OoT	OoM	OoT	3.42	3.42	OoM	OoT	OoM	OoT	49.4	62.7	OoM	OoT	OoM	OoT	62	OoM	OoM	OoT
26	**524**	1155	-	-	-	-	3.1	3.1	-	-	-	-	42.7	65.6	-	-	-	-	49	-	-	-
27	**1167**	2322	-	-	-	-	3	3.1	-	-	-	-	52.3	69.1	-	-	-	-	66.3	-	-	-
28	**5222**	7846	-	-	-	-	4.8	4.8	-	-	-	-	58.3	61.7	-	-	-	-	77.6	-	-	-
29	**14624**	21429	-	-	-	-	4.4	4.4	-	-	-	-	69.7	78.3	-	-	-	-	96.7	-	-	-
30	**29593**	OoT	-	-	-	-	3.8	OoT	-	-	-	-	68.1	OoT	-	-	-	-	90.7	-	-	-

Figure 5.7: Computing the nucleolus of type II games



with the large simplex tableaux, even handled with a state-of-the-art commercial LP solver, is crystal clear.

Figure 5.8: Computing the nucleolus of type III games

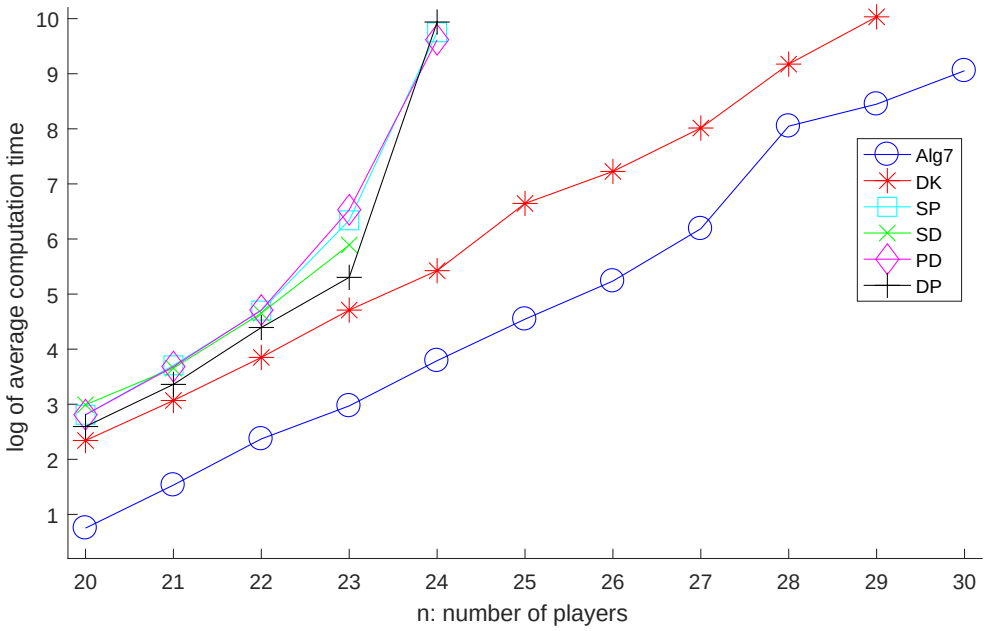


The difference in the number of pivots is fluctuating in size, but it is always in favour of *Alg7*. That comes with a price of notably more subroutine iterations, than compared to sequential LP methods, where the role of the subroutine is completely different: while

Table 5.12: Computing the nucleolus of type III and IV games

n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
5	**0.0009**	0.005	0.002	0.001	0.001	0.001	1.02	1.86	1.02	1.9	1.02	1.02	0.02	6.36	7.5	8.04	7.5	4.22	1.94	1.96	0.42	0.9
10	**0.001**	0.05	0.004	0.007	0.004	0.005	1	5.78	1	2.9	1	1	0	51.8	11.9	57.5	11.9	17.1	2.88	2.84	1.9	2.3
15	**0.007**	0.72	0.13	0.21	0.13	0.071	1	6.64	1	2.34	1	1	0	106	17	95.4	17	25.5	3.56	2.82	2.96	2.52
20	**0.18**	46.8	6.56	14.4	6.65	3.02	1	10.8	1	2.86	1	1	0	230	24.3	707	24.3	37.6	4.2	2.74	3.08	3.22
21	**0.37**	87.6	15.5	23.6	16.3	6.1	1	9.48	1	2.34	1	1	0	216	41.8	469	41.8	37.4	4.04	2.78	3.02	3.42
22	**0.79**	224	35.4	67.7	35.4	13.4	1	11.8	1	2.66	1	1	0	272	44.1	885	44.1	41.2	4.36	3.12	2.9	3.42
23	**1.57**	461	111	120	116	32.8	1	10.5	1	2.18	1	1	0	272	44.9	782	44.9	41.6	4.5	3.04	3.08	3.7
24	**3.15**	1138	5414	15380	6302	1252	1	12.8	1	2.72	1	1	0	331	46.6	1589	46.6	45	4.72	3.1	3.24	3.88
25	**6.43**	2600	OoM	OoT	OoM	OoT	1	11.7	OoM	OoT	OoM	OoT	0	354	OoM	OoT	OoM	OoT	4.82	OoM	OoM	OoT
26	**13.6**	5705	-	-	-	-	1	13.5	-	-	-	-	0	385	-	-	-	-	5.1	-	-	-
27	**29.1**	12208	-	-	-	-	1	12.6	-	-	-	-	0	408	-	-	-	-	5.2	-	-	-
28	**138**	29867	-	-	-	-	1	14.7	-	-	-	-	0	471	-	-	-	-	5.4	-	-	-
29	**115**	OoT	-	-	-	-	1	OoT	-	-	-	-	0	OoT	-	-	-	-	6	-	-	-
30	**239**	-	-	-	-	-	1	-	-	-	-	-	0	-	-	-	-	-	6	-	-	-
5	0.003	0.005	0.002	**0.001**	0.002	0.002	1.78	2.26	1.78	2.42	1.8	1.8	3.36	5.54	7.98	11.7	7.28	8.46	5.5	2.94	1.18	1.38
10	0.007	0.013	0.007	0.006	0.007	**0.005**	1.88	2.66	1.88	2.4	1.88	1.88	5.8	14.5	15.9	42.6	13.8	36.4	8.5	3.34	1.88	1.78
15	**0.052**	0.19	0.26	0.27	0.26	0.19	2.06	2.94	2.06	2.52	2.06	2.06	8.18	24.7	23.9	90	21.2	80.8	11.3	3.8	2.32	2.22
20	**2.12**	10.4	16.6	19.9	16.6	13.4	2.6	3.76	2.6	3.46	2.6	2.6	11.9	42	36.3	245	29.7	192	16.4	5.08	3.38	3.38
21	**4.63**	21.5	40.7	38.7	39.8	28.8	2.6	3.84	2.6	3.38	2.6	2.6	13.9	48	48.2	231	41.7	192	19	4.94	3.3	3.26
22	**10.7**	47	110	105	111	80.9	3.02	4.18	3.02	3.88	3.02	3.02	14.9	48.3	65.8	331	57.1	250	20.6	5.98	3.84	3.86
23	**19.5**	111	566	362	685	201	2.72	4.24	2.72	3.76	2.72	2.72	14.1	57.9	96.2	362	88.3	259	19.8	5.42	3.52	3.7
24	**44.3**	227	17201	OoT	15016	20713	2.78	4.18	2.78	OoT	2.78	2.78	15.3	58	97.9	OoT	89.5	356	21	5.66	3.76	4
25	**94.1**	769	OoM	-	OoM	OoT	3.2	4.64	OoM	-	OoM	OoT	16.1	125	OoM	-	OoM	OoT	23.3	OoM	OoM	OoT
26	**188**	1375	-	-	-	-	3.4	5.2	-	-	-	-	16.4	70.5	-	-	-	-	22.6	-	-	-
27	**486**	3024	-	-	-	-	3.6	5.4	-	-	-	-	19.7	78.3	-	-	-	-	31	-	-	-
28	**3128**	9648	-	-	-	-	3.7	6	-	-	-	-	19.3	106	-	-	-	-	26.3	-	-	-
29	**4665**	22760	-	-	-	-	3.5	6.1	-	-	-	-	21.3	89	-	-	-	-	32.1	-	-	-
30	**8550**	OoT	-	-	-	-	2.8	OoT	-	-	-	-	18.9	OoT	-	-	-	-	25.4	-	-	-

Figure 5.9: Computing the nucleolus of type IV games



SP, *PD* and *DP* uses the subroutine to find the minimal tight set after solving the large LP, instead *Alg7* uses it to solve the large LP *and* to find the minimal tight set during the process.

This phenomena can actually be seen throughout the spectrum of type I-V games and $5 \leq n \leq 30$ players: in every row of Tables 5.11-5.13, *Alg7* requires the minimal number of pivots, and the maximal number of subroutine iterations; with the exception in the latter for 5-player type III games and the 10-player type V game, where *SP* requires slightly more subroutine calls.

From Table 5.12 we can note that the aim of Derks and Kuipers (1997) to find games where the number of iterations required increase more or less linearly with the number of players were successful, at least regarding to their approach. Type III games show the worst-case scenario for dual methods, especially *DK*. On the other hand, these games can be regarded as trivial for primal methods, as they are capable of solving these games in basically a single iteration.

It seems even more outstanding the zero pivots by *Alg7*. As it turns out, for type III games, the ESD starting point is the nucleolus with overwhelmingly large probability. Nevertheless, sequential LP methods still has to perform some, while *DK* quite a significant amount of work, to reach this conclusion, indicating what one can expect worst case from these methods, no matter *how good* starting point is available. However, irrespectively of the special design of these games, Figure 5.8 shows that the scaling of the various algorithms are very similar to what we find in other type of games, with the only main difference of *DK* starts from a clear disadvantage.

Obviously these games are designed to enlarge the iteration gap between primal and dual methods as much as possible, thereby leading to magnitude differences in computation times. Type IV games are indicators of what can we expect when we have a much more moderate iteration gap, and what effect that has on computational times, number of pivots, etc. As expected, after observing the performance of the different methods with small iteration gap (type I-II games), we find that *Alg7* outperforms *DK* and all the other algorithms quite significantly. Figure 5.9 again showing stable scaling for the active-set methods with a noticeable gap between them, while sequential LP methods *fall of a cliff* at 24 players.

At the same time also providing means to solve fairly general cooperative games ($v(S)$ is a random integer between 1 and n for all $|S| > 1$) with 30 players on a desktop PC within a matter of few hours. Putting that into context, it means solving at most 29 linear programs with 31 variables and over a billion constraints for the first LP, while still having at least half a billion in the last one.

Table 5.12 also highlights the advantage of *DP* among the sequential LP methods. It benefits from the good properties of both primal and dual methods to the extent that type III games become trivial as well, while among type IV games it still produces the best computation times in most cases. Even though *DP* makes the classical trade-off between primal and dual methods disappear, the limit of the algorithm is still only 24 players, furthermore, the advantage in computational time for type IV games disappears as well in that limit. This is due to another side of that trade-off, that we revisit in Section 5.5.

Finally we turn to the curious case of the type V, UN Security Council voting games, with the expectation that it primarily challenges *Alg7*, offering a chance to explore the limitations of our algorithm in terms of worst case scenarios.

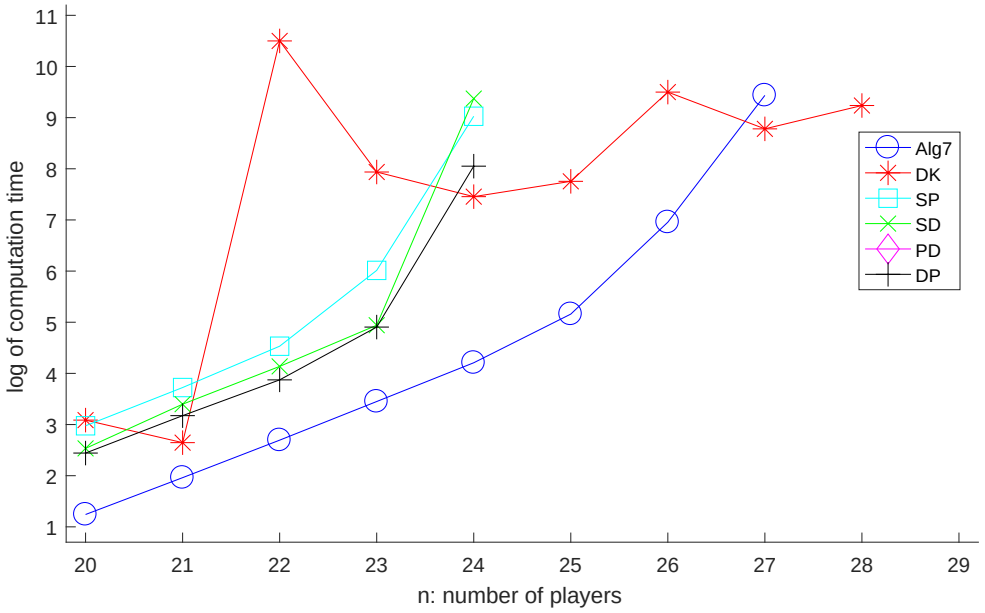
Our attempts to find a game our proposed method struggles with were somewhat successful, meaning that while we have a sizeable advantage in computation time over other algorithms for $n \leq 26$, this advantage vanishes at $n = 27$, until eventually *Alg7* runs out of time for $n = 28$. It is apparent from Figure 5.10 as well, since while *Alg7* scaled reasonably well for all the other type of games, it scales on this special weighted voting game similarly to the way sequential LP methods scale to all types of games.

This is due to the fact that these games have extremely large tight sets, which severely affects *Alg7* through Algorithm 5 with a very large $|\mathcal{T}^{(1)}|$ of exponential size, while by the nature of Derks and Kuipers (1997)'s method, working with dual non-tight sets, this does not necessarily effect *DK* directly, allowing the method to find the nucleolus fast for $n = 27$, and even manage 28 players. Notice however the wild swings in performance for *DK*, for the very same reason *Alg7* suffers. If we are lucky enough, the purely dual method finds a much more restricted sized dual non-tight set, within the exponential sized primal tight set, for which the dual relaxation is feasible. However, if we are unlucky, that search

Table 5.13: Computing the nucleolus of type V games

n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
7	0.005	0.01	0.006	0.04	**0.004**	0.05	2	3	2	3	2	2	2	7	12	22	11	16	5	4	2	2
10	**0.005**	0.046	0.013	0.21	0.01	0.63	2	6	2	3	2	2	2	34	16	47	17	28	5	6	3	3
15	**0.083**	0.45	0.42	0.41	0.68	0.24	2	7	2	5	2	2	2	60	21	144	22	26	5	4	5	4
20	**3.46**	21.9	19.6	12.6	OoT	11.5	2	4	2	3	OoT	2	2	155	26	156	OoT	17	6	5	OoT	5
21	**7.11**	14.1	41.4	29.9	-	23.9	2	3	2	4	-	2	2	58	35	63	-	17	5	5	-	4
22	**14.8**	36298	92.9	62.6	-	48.1	2	13	2	4	-	2	2	76371	39	82	-	22	5	5	-	4
23	**31.5**	2799	409	140	-	135	2	15	2	4	-	2	2	3320	42	52	-	21	5	5	-	5
24	**67.2**	1729	8323	11762	-	3137	2	3	2	4	-	2	2	800	45	94	-	10	5	5	-	4
25	**174**	2332	OoM	OoT	-	OoT	2	12	OoM	OoT	-	OoT	2	458	OoM	OoT	-	OoT	5	OoM	-	OoT
26	**1049**	13348	-	-	-	-	2	10	-	-	-	-	2	1328	-	-	-	-	5	-	-	-
27	12519	**6506**	-	-	-	-	2	15	-	-	-	-	2	260	-	-	-	-	5	-	-	-
28	OoT	**10269**	-	-	-	-	OoT	10	-	-	-	-	OoT	148	-	-	-	-	OoT	-	-	-
29	-	OoT	-	-	-	-	-	OoT	-	-	-	-	-	OoT	-	-	-	-	-	-	-	-

Figure 5.10: Computing the nucleolus of type V games



for the small non-tight set within the vast tight-set can take a lot of time, as it happens for 22 players, and produce strangely non-monotonic computation times throughout the player set sizes, mainly driven by similarly hectic number of iterations and pivots.

5.5 Increasing the number of iterations

In this section we focus our attention on the task required to perform Step 5 of Algorithm 1, that is, finding span($\Delta_r^{(k-1)}$), or equivalently, all coalitions outside of this subspace. This problem is largely overlooked by the literature, except for a few special cases (e.g. Nguyen and Thomas (2016) treat this together with their constraint generation). We already dealt with this problem in Section 2.4, offering a necessary, and a sufficient condition, however, we return to this to consider a conceptually new approach, that can prove useful in practice. Moreover, the analysis of the results reveal another aspect of the classical primal-dual trade-off that we considered in Section 4.2.

Let us start with returning to the sequence of (primal) LPs ($\mathbf{P^{(k)}}$) and it's similarity to Solymosi (1993)'s primal LP sequence. In Section 2.1 we noted that the main difference between the two sequences is ($\mathbf{P^{(k)}}$) having less number of equalities for settled coalitions, relying only on it's representative set, while the other has more inequalities.

Although, another slight difference appears in the wording as well, since Solymosi (1993) formulates $\Sigma^{(k-1)}$ as all coalitions that do not have constant excess $E(S, x)$ throughout

the previous solution space $x \in \mathcal{P}^{(k-1)*}$. It is easy to see that $S \in \text{span}(\Delta_r^{(k-1)})$ implies $x(S)$ being constant in $\mathcal{P}^{(k-1)*}$, thus having constant excess as well. In fact, the implication the other way around also holds true, for any LP in general.

Therefore let us now turn to the equivalence of constraints with constant, but not necessarily zero slack values throughout the solution set of a general LP and the set of constraints spanned by the minimal tight set, or as also known in optimisation, the *optimal partition*. As we already noted, the results of Lemma 2.3 can be applied to any LP, as we have not exploited any special structure of the LPs related to the computation of the nucleolus. The same can be noted for Propositions 2.5 and Corollary 2.6 as well. As the following proposition considers the general case, accordingly we are referring to these results in their general form.

Consider the feasible primal-dual pair $\min_x\{c^\top x : Ax \leq b\}$ and $\max_y\{b^\top y : y^\top A = c, y \geq 0\}$. Suppose $A \in \mathbb{R}^{m\times n}$, denote it's rows with $a_j, j = 1, \ldots, m$ and denote with T^* the minimal tight set, the indices of primal constraints which are active for all primal solutions.

Proposition 5.1. *Suppose there exists $\beta \in \mathbb{R}$ and $j \notin T^*$ such that $a_j^\top x = \beta < b_j$ for all primal solution x. Then $a_j \in \text{span}\{a_i : i \in T^*\}$.*

Proof. We claim that $\exists \mu : a_j = \sum_{i\in T^*} \mu_i a_i$. If not, then by the Fredholm alternative there exists a d such that $a_i^\top d = 0$ for all $i \in T^*$ and $a_j^\top d \neq 0$.

Take a primal solution x^* such that $a_i^\top x^* = b_i \iff i \in T^*$, such a choice is possible due to Lemma 2.3. It follows that there exists $\delta \neq 0$ such that $A(x + \delta d) \leq b$, while for any δ we have $a_i^\top(x+\delta d) = b_i$ for all $i \in T^*$ and $a_j^\top(x+\delta d) \neq \beta$. Hence $x+\delta d$ is feasible in the primal problem for some non-zero δ, thus we also have $c^\top x \leq c^\top(x + \delta d)$. In order to get contradiction, we need equality, or in other words, to reach a contradiction it remains to be seen that $\delta c^\top d = 0$.

According to Proposition 2.5 and Corollary 2.6 for every $i \in T^*$ there exists a dual solution with $y_i > 0$, and vice versa, for every dual solution y with $y_i > 0$, we have $i \in T^*$. It follows that $y_j = 0$ for any dual solution y and all $j \notin T^*$. Denote with y_{T^*} and A_{T^*} the truncated vector and matrix we get from y and A by only considering coordinates/rows belonging to T^*. Then $c = y_{T^*}^\top A_{T^*}$ by dual feasibility. However, by the choice of d we have $c^\top d = 0$ and the proof is complete. □

Now we turn our attention back to the task of finding all *unsettled* coalitions that are in the linear span of the *settled* coalitions, i.e. *Step 5* of Algorithms 1 and 7, and *Step 7* of Algorithms 12 and 13.

We argue that, in practice, it might be worthwhile to not perform this task at all. Notice that omitting this step from the algorithms will come with a consequence that results

like Proposition 2.4 or Theorem 3.13 will not hold any more. In fact, the number of iterations required immediately jumps from at most $n-1$ to $\mathcal{O}(2^n)$, that is from surely linear to possibly exponential. However, we argue that the actual increase in the number of iterations will be tractable. Moreover, we further argue that the additional LPs that we have to solve throughout the process can be regarded as *trivial*.

Consider the modified version of ($\mathbf{P^{(k)}}$)

$$
\begin{array}{lrll}
\min\limits_{x^{(k)},\epsilon^{(k)}} & \epsilon^{(k)} & & \\
\text{s.t.} & \epsilon^{(k)} + x^{(k)}(S) & \geq v(S) & \forall S \notin \widehat{\Delta}^{(k-1)} \\
& x_i^{(k)} & \geq v(\{i\}) & \forall \{i\} \notin \widehat{\Delta}^{(k-1)} \\
& x^{(k)}(S) & = v(S) - \widehat{y}_S^* & \forall S \in \widehat{\Delta}_r^{(k-1)}
\end{array}
\qquad (\widehat{\mathbf{P}}^{(\mathbf{k})})
$$

where

- $\widehat{\mathcal{T}}^{(0)*} = N$ and $\widehat{\widetilde{\mathcal{T}}}^{(0)*} = \emptyset$,
- $\widehat{\Delta}^{(k)} = \bigcup\limits_{i=0}^{k} \left(\widehat{\mathcal{T}}^{(i)*} \cup \widehat{\widetilde{\mathcal{T}}}^{(i)*} \right)$
- for all k: $\operatorname{span}(\widehat{\Delta}_r^{(k)}) = \operatorname{span}(\widehat{\Delta}^{(k)})$ and $|\widehat{\Delta}_r^{(k)}| = \operatorname{rank}(\widehat{\Delta}_r^{(k)})$, and
- $\widehat{y}_S^* = \epsilon^{(k)*}$ for all $S \in \widehat{\mathcal{T}}^{(k)*}$.

The only difference is the additional constraints, not being removed, for $S \in \operatorname{span}(\widehat{\Delta}_r^{(k-1)}) \setminus \widehat{\Delta}^{(k-1)}$.

For the sake of keeping track of the iterations in the two sequence, let us suppose that they *more or less coincide* in the first $(k-1)$ iteration, that is $\operatorname{span}(\Delta_r^{(k-1)}) = \operatorname{span}(\widehat{\Delta}_r^{(k-1)})$. In order for ($\widehat{\mathbf{P}}^{(\mathbf{k})}$), the LP we need to solve in the k-th iteration of the new sequence, to be an *unnecessary* extra, additional large LP to solve compared to the original scheme, $\widehat{\mathcal{T}}^{(k)*} \cup \widehat{\widetilde{\mathcal{T}}}^{(k)*}$ needs to lie entirely in $\operatorname{span}(\Delta_r^{(k-1)})$. As soon as there is any coalition in the minimal tight set outside of the coalition subspace $\operatorname{span}(\Delta_r^{(k-1)})$, then we would have to solve the k-th LP in the original scheme anyway.

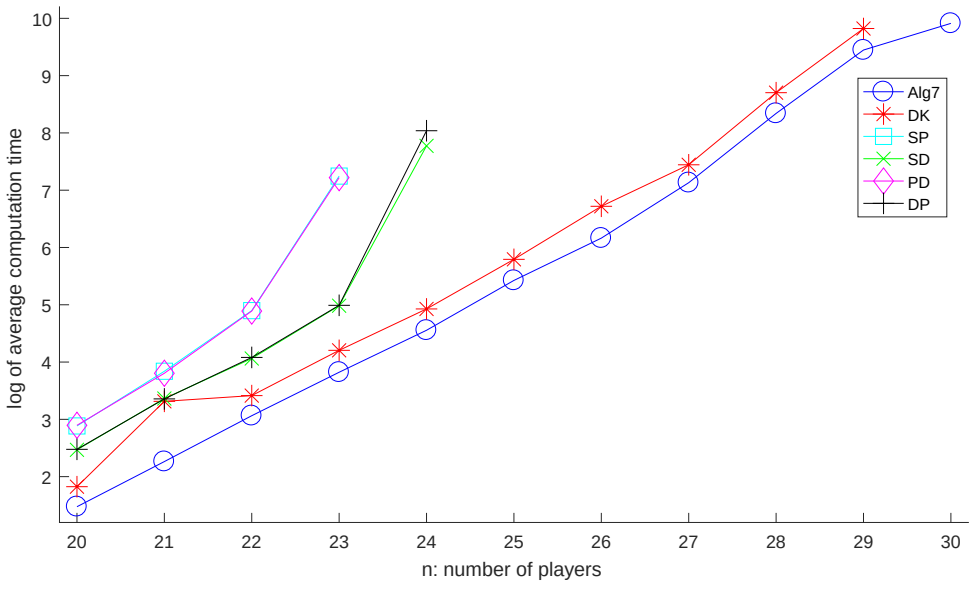
Furthermore, if indeed $\widehat{\mathcal{T}}^{(k)*} \cup \widehat{\widetilde{\mathcal{T}}}^{(k)*} \subseteq \operatorname{span}(\widehat{\Delta}_r^{(k-1)})$, then we know that all coalitions in the minimal tight set $\widehat{\mathcal{T}}^{(k)*} \cup \widehat{\widetilde{\mathcal{T}}}^{(k)*}$ already had constant excess among the solutions of the $(k-1)$-th LP. This can be straightforwardly seen by noting that Proposition 5.1 works the other way around as well: naturally if a coalition is in the linear span of a minimal tight set, then it has constant excess throughout the solution space. Or in another words, the optimal partition of the LP lies completely in the span of the equality constraints. Therefore we argue that solving these additional LPs require very little extra effort, particularly in terms of additional pivots compared to the additional

iterations. Therefore our expectation is that the ratio of extra number of pivots over extra number of iterations is negligible (i.e. close to 0).

In the followings we report results from running the same algorithms on the same instances with the single modification of omitting the so-called *linear speed-up*, that is only removing constraints/variables corresponding to $\widehat{\mathcal{T}}^{(k)*} \cup \widehat{\widehat{\mathcal{T}}}^{(k)*}$ or $\widehat{\mathcal{NT}}^{(k)}(u^{(k)}) \cup \widehat{\widehat{\mathcal{NT}}}^{(k)}(u^{(k)})$ instead of removing everything in $\text{span}(\widehat{\Delta}_r^{(k-1)})$. Additionally, in Tables 5.14-5.18 we present the change in the results from the original to the modified version. The changes are colour coded as green indicates increase, red indicate decrease, and black means no change. Our expectation is that while iteration, pivot and subroutine numbers can certainly increase, not having to perform the difficult task of the linear speed-up, combined with the expected, only marginal increase in pivots per increase in iterations can lead to a decrease in overall computation time.

Starting with general observations overarching types and player set sizes, we while number of iterations, pivots and subroutines indeed increase with this change outside of a few exceptions, and often quite significantly, we still find numerous cases of decreasing computational times. However, this decrease seem to concentrate mostly on the active-set methods (*Alg7* and *DK*) and somewhat on dual sequential LP methods (*SD* and *DP*), as it rarely ever appears for primal sequential LP methods (*SP* and *PD*).

Figure 5.11: Computing the nucleolus of type I games – no linear speed-up



Another general observation is that the change of omitting linear speed-up does not have much effect on what instances each method can handle. Outside of the curious type V games, active-set methods *Alg7* and *DK* can still go up to 30- and 29-players respectively (28 for *DK* on type III games). Primal sequential LP methods, with their mostly increased running time start struggling earlier at $n = 24$ outside of type III and

Table 5.14: Computing, and change in computing the nucleolus of type I games – no linear speed-up

n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
5	0.006	0.006	0.003	**0.002**	0.003	**0.002**	2.62	2.62	2.62	2.76	2.62	2.76	5.46	6.14	10.1	16.6	8.02	16.6	8.08	4.24	1.62	1.76
10	0.011	0.015	0.008	**0.005**	0.007	0.007	2.56	2.58	2.56	2.6	2.56	2.56	11.9	14.7	18.9	76.2	15.3	75.5	14.5	4.14	1.6	1.6
15	**0.1**	0.13	0.23	0.13	0.23	0.13	1.92	1.98	1.92	1.96	1.92	1.92	19.3	24.9	25.4	177	22.9	176	21.2	2.84	0.96	0.96
20	**4.37**	6.2	17.9	11.8	18.1	11.9	3.1	3.16	3.1	3.12	3.1	3.1	28.9	41.9	43.5	421	33.3	421	32	5.2	2.16	2.12
21	**9.61**	27.5	46.5	28.9	44.9	28.7	3.54	15.1	3.54	3.62	3.54	3.54	31.1	139	48.7	473	37	470	34.7	6.1	2.64	2.62
22	**21.4**	30.4	134	58.1	133	59.2	3.12	3.14	3.12	3.12	3.12	3.12	33.9	50.9	164	580	154	580	37	5.24	2.14	2.12
23	**45.7**	67	1403	146	1366	147	3.5	3.56	3.5	3.52	3.5	3.5	35.1	54.7	258	693	241	692	38.6	6	2.56	2.52
24	**95.1**	138	OoT	2378	OoT	3097	2.74	2.74	OoT	2.74	OoT	2.74	35.7	54.6	OoT	666	OoT	666	38.5	OoT	OoT	1.74
25	**227**	328	-	OoT	-	OoT	5.64	5.74	-	OoT	-	OoT	42.3	61.5	-	OoT	-	OoT	47.9	-	-	OoT
26	**475**	828	-	-	-	-	5.7	5.8	-	-	-	-	43.6	76.6	-	-	-	-	49.3	-	-	-
27	**1250**	1709	-	-	-	-	11.9	11.9	-	-	-	-	57.1	77.5	-	-	-	-	69	-	-	-
28	**4189**	6017	-	-	-	-	8.5	8.7	-	-	-	-	49.2	89	-	-	-	-	57.7	-	-	-
29	**12666**	18444	-	-	-	-	8.3	8.3	-	-	-	-	58.1	85.9	-	-	-	-	66.4	-	-	-
30	**20177**	OoT	-	-	-	-	3	OoT	-	-	-	-	44.4	OoT	-	-	-	-	47.4	-	-	-
5	14%	9%	8%	22%	-76%	9%	4%	4%	4%	10%	4%	10%	0.7%	1%	3%	13%	1%	13%	1%	5%	7%	16%
10	0.4%	8%	18%	-68%	-31%	18%	27%	26%	27%	26%	27%	27%	0.8%	4%	13%	20%	4%	21%	2%	35%	51%	51%
15	-2%	-20%	21%	-22%	24%	-7%	28%	30%	28%	29%	28%	28%	0.2%	-4%	9%	15%	2%	15%	0.8%	42%	85%	85%
20	-0.6%	-19%	90%	40%	96%	66%	112%	116%	112%	114%	112%	112%	0.1%	3%	34%	28%	5%	28%	2%	171%	370%	361%
21	-3%	4%	94%	57%	86%	58%	108%	722%	108%	108%	108%	108%	0.7%	23%	32%	32%	5%	32%	2%	152%	257%	254%
22	0.7%	-15%	47%	48%	44%	45%	90%	91%	90%	90%	90%	90%	0.6%	3%	7%	44%	1%	44%	1%	130%	234%	231%
23	-1%	-15%	95%	60%	86%	59%	119%	122%	119%	120%	119%	119%	1%	3%	7%	39%	0.8%	39%	4%	173%	313%	306%
24	-0.8%	-15%	-	-63%	-	-57%	71%	71%	-	71%	-	71%	0.4%	2%	-	28%	-	28%	1%	-	-	190%
25	-0.4%	-23%	-	-	-	-	217%	219%	-	-	-	-	0.4%	9%	-	-	-	-	0.8%	-	-	-
26	3%	-14%	-	-	-	-	256%	262%	-	-	-	-	7%	7%	-	-	-	-	14%	-	-	-
27	-0.9%	-17%	-	-	-	-	526%	526%	-	-	-	-	0%	10%	-	-	-	-	0%	-	-	-
28	-5%	-6%	-	-	-	-	347%	358%	-	-	-	-	1%	8%	-	-	-	-	7%	-	-	-
29	4%	4%	-	-	-	-	388%	388%	-	-	-	-	0%	10%	-	-	-	-	0%	-	-	-
30	5%	-	-	-	-	-	114%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-

V games, while the only change we experience for *SD* is it can not handle 24-player type IV games, and even the 20-player type V game already makes it run out of memory. On the other hand, *DP*, likewise *Alg7*, not effected by the change whatsoever, in terms of what instances can it successfully tackle.

From Tables 5.14 and 5.15 it is apparent that most of our general observations from before still stand: when the gap in the number of iterations required for purely dual methods and the rest is marginal, the dual based sequential methods outperform the primal based ones. However, even though the decrease in computation time is significantly larger for *DK*, than for *Alg7*, we still find significantly better computation times in the case of our proposed method *Alg7*.

Looking how the various modified methods scale based on Figures 5.11 and 5.12, comparing them to Figures 5.6 and 5.7 including the linear speed-up, they look very much the same. The only notable differences for both type I and II games are the downturn of primal sequential LP methods and the slightly smaller gap between the active-set methods.

While looking at the difference linear speed-up makes in Tables 5.14 and 5.15, we find results matching our expectations. Number of iterations, pivots and subroutines are naturally increasing without the speed-up (outside of few exceptions), but the relative increase of pivots per increase of iterations are almost negligible for most methods. For type I and II games the relative increase of pivots for *Alg7* is around 1%, for *DK* around 7-8%, for *SP* this ranges in 30-50%, for *PD* it is 6-9%, while for both *SD* and *DP* this is around 43% for type I and around 102% for type II games. However, this only translates to a decrease in computational time mostly for *Alg7* and *DK*, especially the latter.

Figure 5.12: Computing the nucleolus of type II games – no linear speed-up

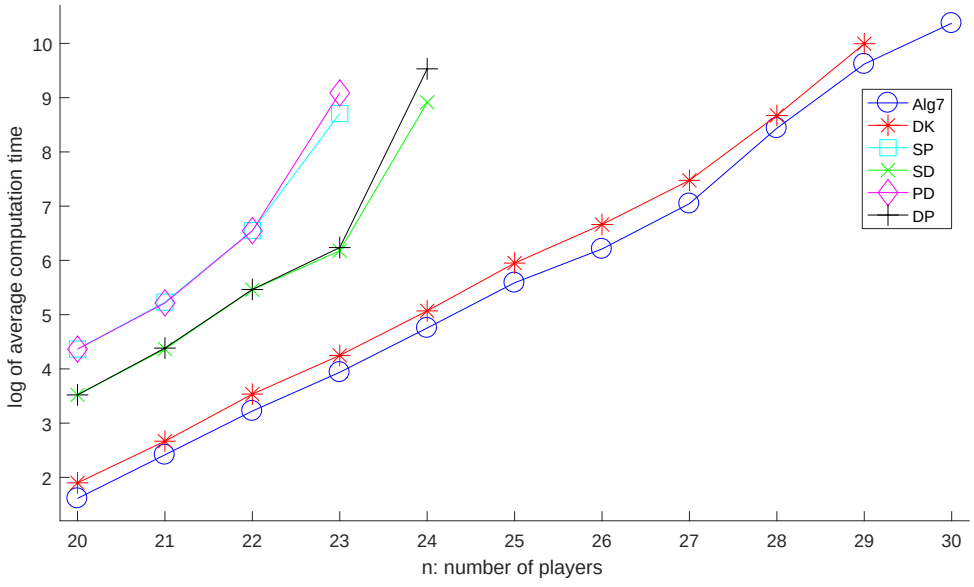


Table 5.15: Computing, and change in computing the nucleolus of type II games – no linear speed-up

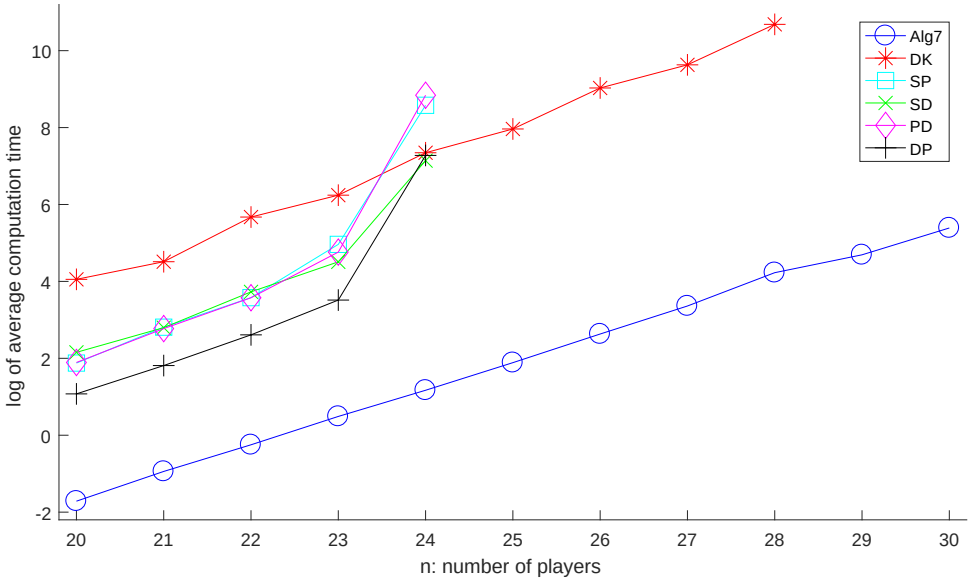
n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
5	0.005	0.006	0.003	**0.002**	0.003	**0.002**	2.62	2.66	2.62	2.78	2.62	2.74	5.14	5.78	10	15.9	8.32	15.7	7.78	3.96	1.62	1.76
10	0.011	0.015	0.012	**0.006**	0.012	**0.006**	3.18	3.28	3.18	3.26	3.18	3.18	12.1	14.9	19.3	65	15.5	63.7	15.3	5.26	2.24	2.24
15	**0.12**	0.14	1.14	0.34	1.17	0.35	5.84	6	5.84	5.94	5.84	5.84	22	26.3	44.1	254	25.7	248	27.9	10.7	5	4.94
20	**5.01**	6.68	78.7	33.8	78.6	33.8	8.38	8.54	8.38	8.5	8.38	8.38	33.4	46.8	73.6	600	37	592	41.8	15.8	7.52	7.48
21	**11.2**	14.4	187	78.7	184	80.1	9.16	9.22	9.16	9.2	9.16	9.16	36.7	50.4	93.4	775	45.3	773	45.9	17.3	8.2	8.2
22	**25.1**	34.3	699	235	697	236	12.1	12.3	12.1	12.2	12.1	12.1	40	58.2	155	1119	79.5	1110	52.1	23.1	11.3	11.2
23	**51.3**	70	6045	482	8850	511	10.6	10.7	10.6	10.7	10.6	10.6	39.7	57.1	208	832	138	1066	50.3	20.2	9.74	9.7
24	**116**	159	OoT	7435	OoT	13785	11.9	12.3	OoT	12.2	OoT	11.9	43.8	63.6	OoT	1329	OoT	1302	55.7	OoT	OoT	11.2
25	**267**	384	-	OoT	-	OoT	14	14.1	-	OoT	-	OoT	49.8	74.7	-	OoT	-	OoT	63.8	-	-	OoT
26	**498**	782	-	-	-	-	10.5	10.6	-	-	-	-	45.6	73.6	-	-	-	-	56.1	-	-	-
27	**1148**	1765	-	-	-	-	13.9	14.9	-	-	-	-	52.3	83.5	-	-	-	-	66.3	-	-	-
28	**4610**	5833	-	-	-	-	36.6	36.7	-	-	-	-	67.2	98.2	-	-	-	-	104	-	-	-
29	**15058**	21897	-	-	-	-	27	27.3	-	-	-	-	69.7	109	-	-	-	-	96.7	-	-	-
30	**31962**	OoT	-	-	-	-	26.5	OoT	-	-	-	-	71.7	OoT	-	-	-	-	98.2	-	-	-
5	11%	18%	9%	25%	0.8%	2%	2%	2%	2%	7%	2%	7%	0.4%	0.7%	1%	8%	0.7%	9%	1%	3%	4%	11%
10	2%	5%	8%	-14%	5%	-14%	20%	21%	20%	20%	20%	20%	0.8%	1%	9%	22%	3%	23%	3%	22%	32%	30%
15	1%	-33%	40%	16%	44%	34%	116%	121%	116%	117%	116%	116%	3%	8%	69%	122%	14%	121%	8%	143%	181%	178%
20	-0.7%	-35%	59%	87%	58%	103%	179%	183%	179%	181%	179%	179%	0.7%	19%	100%	190%	17%	186%	2%	214%	265%	263%
21	-1%	-32%	68%	108%	65%	109%	183%	183%	183%	184%	183%	183%	0.1%	17%	107%	180%	16%	179%	0.6%	215%	266%	266%
22	0.5%	-27%	59%	157%	59%	154%	265%	271%	265%	268%	265%	265%	3%	21%	102%	241%	12%	244%	6%	313%	382%	384%
23	-2%	-26%	114%	108%	210%	124%	257%	263%	257%	259%	257%	257%	5%	18%	53%	187%	6%	276%	10%	310%	397%	390%
24	-0.4%	-21%	-	-62%	-	-32%	303%	314%	-	311%	-	303%	2%	25%	-	287%	-	272%	3%	-	-	458%
25	-1%	-32%	-	-	-	-	308%	313%	-	-	-	-	0.9%	19%	-	-	-	-	3%	-	-	-
26	-5%	-32%	-	-	-	-	239%	242%	-	-	-	-	7%	12%	-	-	-	-	14%	-	-	-
27	-2%	-24%	-	-	-	-	363%	381%	-	-	-	-	0%	21%	-	-	-	-	0%	-	-	-
28	-12%	-26%	-	-	-	-	662%	665%	-	-	-	-	15%	59%	-	-	-	-	34%	-	-	-
29	3%	2%	-	-	-	-	514%	520%	-	-	-	-	0%	39%	-	-	-	-	0%	-	-	-
30	8%	-	-	-	-	-	597%	-	-	-	-	-	5%	-	-	-	-	-	8%	-	-	-

Table 5.16: Computing, and change in computing the nucleolus of type III games – no linear speed-up

n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
5	**0.001**	0.007	0.002	0.002	0.001	0.002	1.02	1.86	1.02	1.9	1.02	1.02	0.02	6.36	7.5	7.38	7.5	4.22	1.94	1.96	0.42	0.9
10	**0.001**	0.077	0.004	0.009	0.004	0.003	1	15.2	1	2.9	1	1	0	71.6	11.9	55.3	11.9	17.1	2.88	2.84	1.9	2.3
15	**0.007**	0.7	0.14	0.12	0.13	0.058	1	21.4	1	2.38	1	1	0	149	17	93.8	17	25.5	3.56	2.82	2.96	2.52
20	**0.18**	57.6	6.53	8.63	6.63	2.93	1	72.9	1	2.84	1	1	0	472	24.3	590	24.3	37.6	4.2	2.74	3.08	3.22
21	**0.39**	91.1	16.6	16.4	16	6.12	1	54.4	1	2.3	1	1	0	382	41.8	395	41.8	37.4	4.04	2.78	3.02	3.42
22	**0.78**	291	35.8	41.6	35.7	13.6	1	90.1	1	2.52	1	1	0	600	44.1	641	44.1	41.2	4.36	3.12	2.9	3.42
23	**1.63**	514	143	90.8	117	33.6	1	68.8	1	2.32	1	1	0	510	44.9	930	44.9	41.6	4.5	3.04	3.08	3.7
24	**3.22**	1549	5321	1262	6909	1445	1	108	1	2.7	1	1	0	754	46.6	1381	46.6	45	4.72	3.1	3.24	3.88
25	**6.6**	2875	OoM	OoT	OoM	OoT	1	85.6	OoM	OoT	OoM	OoT	0	665	OoM	OoT	OoM	OoT	4.82	OoM	OoM	OoT
26	**13.9**	8346	-	-	-	-	1	131	-	-	-	-	0	966	-	-	-	-	5.1	-	-	-
27	**28.8**	15235	-	-	-	-	1	104	-	-	-	-	0	860	-	-	-	-	5.2	-	-	-
28	**68.5**	43648	-	-	-	-	1	154	-	-	-	-	0	1231	-	-	-	-	5.4	-	-	-
29	**109**	OoT	-	-	-	-	1	OoT	-	-	-	-	0	OoT	-	-	-	-	6	-	-	-
30	**219**	-	-	-	-	-	1	-	-	-	-	-	0	-	-	-	-	-	6	-	-	-
5	11%	28%	4%	35%	-6%	47%	0%	0%	0%	0%	0%	0%	0%	0%	0%	-8%	0%	0%	0%	0%	0%	0%
10	5%	53%	-0.5%	24%	1%	-41%	0%	164%	0%	0%	0%	0%	0%	38%	0%	-4%	0%	0%	0%	0%	0%	0%
15	-0.3%	-2%	10%	-44%	3%	-18%	0%	222%	0%	2%	0%	0%	0%	40%	0%	-2%	0%	0%	0%	0%	0%	0%
20	-0.8%	23%	-0.5%	-40%	-0.2%	-3%	0%	573%	0%	-0.7%	0%	0%	0%	105%	0%	-16%	0%	0%	0%	0%	0%	0%
21	6%	4%	7%	-30%	-1%	0.3%	0%	473%	0%	-2%	0%	0%	0%	77%	0%	-16%	0%	0%	0%	0%	0%	0%
22	-1%	30%	1%	-39%	0.7%	1%	0%	662%	0%	-5%	0%	0%	0%	120%	0%	-28%	0%	0%	0%	0%	0%	0%
23	3%	12%	28%	-24%	1%	2%	0%	553%	0%	6%	0%	0%	0%	87%	0%	19%	0%	0%	0%	0%	0%	0%
24	2%	36%	-2%	-92%	10%	15%	0%	749%	0%	-0.7%	0%	0%	0%	128%	0%	-13%	0%	0%	0%	0%	0%	0%
25	3%	11%	-	-	-	-	0%	633%	-	-	-	-	0%	88%	-	-	-	-	0%	-	-	-
26	3%	46%	-	-	-	-	0%	870%	-	-	-	-	0%	151%	-	-	-	-	0%	-	-	-
27	-1%	25%	-	-	-	-	0%	726%	-	-	-	-	0%	111%	-	-	-	-	0%	-	-	-
28	-50%	46%	-	-	-	-	0%	946%	-	-	-	-	0%	162%	-	-	-	-	0%	-	-	-
29	-5%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-
30	-8%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-

Type III games are meant to make dual methods suffer, while primal methods and *DP* were already very efficient, as can be seen from Table 5.12. Therefore it is not that surprising that the change does not effect the latter methods too much: there is no dramatic, only occasional mild changes in computation time, while there is absolutely no change in the number of iterations, pivots or subroutines. For dual methods, we can fortify that the authors Derks and Kuipers (1997) were successful in challenging *DK* with these games, as the change made computation times, iterations and pivots all significantly increase. Meanwhile *SD* actually benefits in basically all aspects from omitting the linear speed-up.

Figure 5.13: Computing the nucleolus of type III games – no linear speed-up



Even though number of iterations and pivots are increasing at a considerably high rate for *DK* with the modification, this leads to a rather limited increase in computation time, as again, the only notable difference between the Figures of 5.8 and 5.13 is the advantage *SD* gains, eventually reaching the level of *DP* at $n = 24$, making it the best among sequential LP methods, as opposed to being worst with linear speed-up.

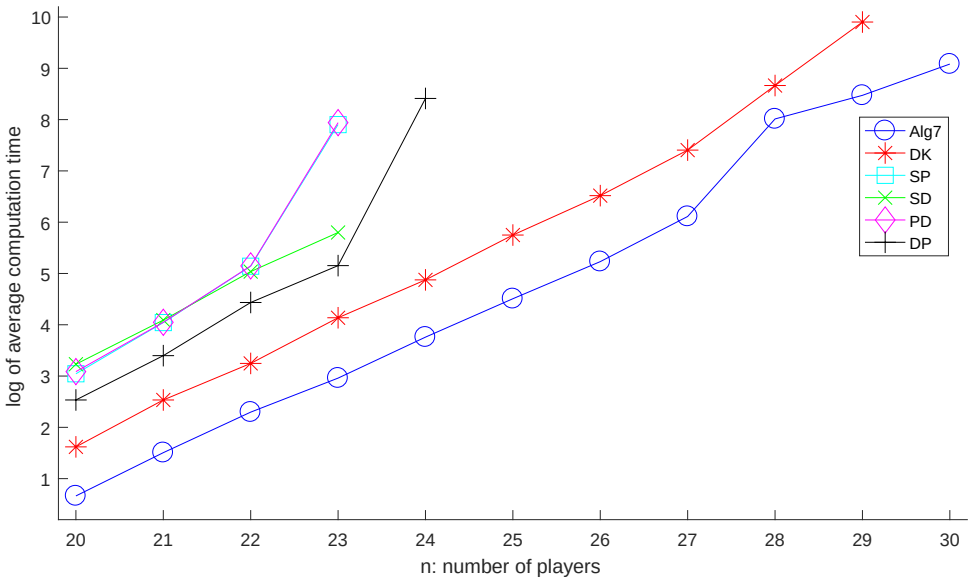
Type IV games, introduced as games with more realistic distinguishing power in the number of iterations between primal methods and *DP* against the purely dual methods, lie somewhere between type I-II games and type III games. The general view from Table 5.17 is that the increase in iteration numbers (without exception), pivots and subroutines can still lead to a decrease in computation time for *Alg7*, *DK* and the dual sequential methods. In fact, for the former two, in most cases computation times are decreasing, mostly almost by half for *DK*.

Obviously the higher the number of iterations were, the more beneficial omitting the linear speed-up could be, provided that the additional, intuitively trivial LPs are indeed

Table 5.17: Computing, and change in computing the nucleolus of type IV games – no linear speed-up

n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
5	0.004	0.006	**0.002**	**0.002**	0.004	**0.002**	1.84	2.4	1.84	2.7	1.84	1.86	3.36	5.6	8.16	13.4	7.34	9.32	5.54	3.06	1.22	1.46
10	0.006	0.015	0.007	0.006	0.01	**0.004**	2.12	4.08	2.12	3.42	2.12	2.12	5.8	15.2	17.3	66.2	14.1	42.1	8.56	3.84	2.26	2.2
15	**0.047**	0.11	0.32	0.26	0.33	0.15	2.52	5.86	2.52	4.62	2.52	2.52	8.22	27.5	27.9	159	21.7	95.4	11.6	4.74	3	2.98
20	**1.94**	5.05	21	25.3	21.9	12.6	3.4	9.58	3.4	6.88	3.4	3.4	12.1	44.6	43.1	494	30.5	245	16.9	6.8	4.62	4.58
21	**4.51**	12.6	57	59.8	57.3	29.9	3.92	11.2	3.92	7.72	3.92	3.92	13.9	56.9	61.1	558	43.5	284	19.2	7.64	5.38	5.1
22	**9.94**	25.7	171	154	172	84.3	4.66	13.5	4.66	8.64	4.66	4.66	15	57.4	80	736	59	422	20.9	9.36	6.44	6.12
23	**19.3**	62.6	2705	330	2810	173	4.16	14	4.16	8.38	4.16	4.16	14.1	67.5	109	728	89.9	371	19.9	8.4	5.58	5.52
24	**42.9**	131	OoT	OoT	OoT	4504	4.28	14.1	OoT	OoT	OoT	4.28	15.4	68.4	OoT	OoT	OoT	538	21.3	OoT	OoT	6.3
25	**90.7**	314	-	-	-	OoT	5.88	18.1	-	-	-	OoT	16.1	80	-	-	-	OoT	23.6	-	-	OoT
26	**187**	677	-	-	-	-	4.5	15.6	-	-	-	-	16.4	84.1	-	-	-	-	22.6	-	-	-
27	**451**	1645	-	-	-	-	9.6	29.4	-	-	-	-	19.7	99.7	-	-	-	-	31.2	-	-	-
28	**3020**	5795	-	-	-	-	5.6	25.7	-	-	-	-	19.4	134	-	-	-	-	27.2	-	-	-
29	**4785**	19982	-	-	-	-	8.3	28.6	-	-	-	-	21.3	120	-	-	-	-	32.1	-	-	-
30	**8807**	OoT	-	-	-	-	4.6	OoT	-	-	-	-	18.9	OoT	-	-	-	-	25.4	-	-	-
5	16%	22%	2%	29%	100%	1%	3%	6%	3%	12%	2%	3%	0%	1%	2%	15%	0.8%	10%	0.7%	4%	3%	6%
10	-4%	12%	2%	-14%	42%	-12%	13%	53%	13%	42%	13%	13%	0%	5%	9%	55%	2%	16%	0.7%	15%	20%	24%
15	-8%	-41%	20%	-3%	24%	-24%	22%	99%	22%	83%	22%	22%	0.5%	11%	17%	77%	3%	18%	2%	25%	29%	34%
20	-9%	-51%	27%	27%	32%	-6%	31%	155%	31%	99%	31%	31%	2%	6%	19%	102%	3%	28%	3%	34%	37%	36%
21	-2%	-42%	40%	55%	44%	4%	51%	191%	51%	128%	51%	51%	0.4%	19%	27%	141%	4%	48%	0.8%	55%	63%	56%
22	-7%	-45%	56%	46%	56%	4%	54%	222%	54%	123%	54%	54%	0.5%	19%	22%	122%	3%	69%	2%	57%	68%	59%
23	-1%	-43%	378%	-9%	310%	-14%	53%	230%	53%	123%	53%	53%	0%	17%	13%	101%	2%	43%	0.4%	55%	59%	49%
24	-3%	-42%	-	-	-	-78%	54%	238%	-	-	-	54%	0.7%	18%	-	-	-	51%	2%	-	-	57%
25	-4%	-59%	-	-	-	-	84%	291%	-	-	-	-	0.1%	-36%	-	-	-	-	1%	-	-	-
26	-0.4%	-51%	-	-	-	-	32%	200%	-	-	-	-	0%	19%	-	-	-	-	0%	-	-	-
27	-7%	-46%	-	-	-	-	167%	444%	-	-	-	-	0%	27%	-	-	-	-	0.6%	-	-	-
28	-3%	-40%	-	-	-	-	51%	328%	-	-	-	-	0.5%	26%	-	-	-	-	3%	-	-	-
29	3%	-12%	-	-	-	-	137%	369%	-	-	-	-	0%	35%	-	-	-	-	0%	-	-	-
30	3%	-	-	-	-	-	64%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-

Figure 5.14: Computing the nucleolus of type IV games – no linear speed-up



can be handled efficiently. From larger computation times in Tables 5.12, it is easier for *DK* to reach a considerable decrease, while one could argue that *Alg7* being already efficient in the previous setting, it is less of a surprising to see milder changes in the results. However, it is noteworthy that while number of iterations still increase significantly for *Alg7* as well, the number of pivots and subroutine iterations barely increase, thereby leading to a moderate decrease in computation time for most cases.

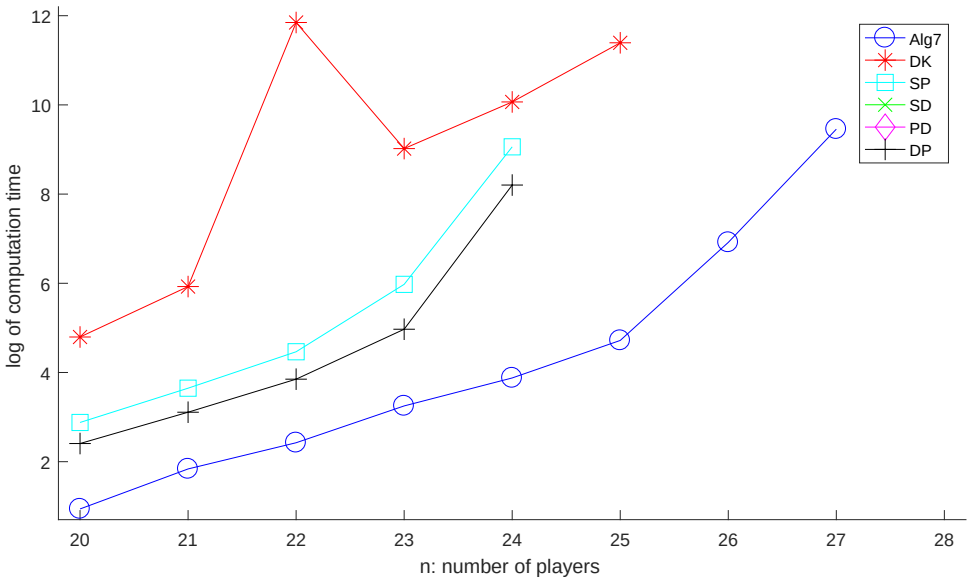
As a result, while the gap is narrowing, as can also be seen by comparing Figures 5.9 and 5.14, our proposed method still produces the best computation times by a considerable margin. Additionally, note how much *DP* can gain from dropping the linear speed-up at the limit of 24 players, while the other sequential LP methods can not handle the extra iterations: the 4504 seconds are less than a third of the best 24-player, type IV average computation time amongst sequential LP methods with linear-speed up (15016, produced by *PD*).

Finally we return to the peculiar case of type V games, voting games manufactured to produce extremely large tight sets. It is notable that while these games first and foremost came to life to challenge primal methods using dual subroutines, without the linear speed-up it actually makes purely dual methods collapse. *DK* is capable of finding a solution for these games up to 25 players, but computation times are often extremely high due to the very high number of iterations and inconsistent number of pivots, while *SD* can not even compute the solution for 20-players. The struggle of *PD* at the same level of 20 players remains unchanged. Meanwhile for the other primal methods (and *DP*), similarly to type III games, number of iterations, pivots (except for *DP*) and subroutines do not change, but computation times decrease in most cases.

Table 5.18: Computing, and change in computing the nucleolus of type V games – no linear speed-up

n	Time						Iterations						Pivots						Subroutine iterations			
	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	DK	SP	SD	PD	DP	Alg7	SP	PD	DP
7	0.006	0.014	0.005	**0.004**	0.005	**0.004**	2	4	2	6	2	2	2	7	12	36	11	16	5	4	2	2
10	**0.006**	0.073	0.012	0.022	0.01	0.01	2	28	2	20	2	2	2	36	16	299	17	28	5	6	3	3
15	**0.058**	1.4	0.42	16.4	0.52	0.22	2	1014	2	386	2	2	2	188	21	8157	22	36	5	4	5	4
20	**2.56**	121	17.8	OoM	OoT	11.1	2	32753	2	OoM	OoT	2	2	155	26	OoM	OoT	20	6	5	OoT	5
21	**6.28**	375	38.4	-	-	22.4	2	65520	2	-	-	2	2	58	35	-	-	15	5	5	-	4
22	**11.3**	139732	86.9	-	-	47	2	65519	2	-	-	2	2	305515	39	-	-	22	5	5	-	4
23	**25.8**	8268	394	-	-	144	2	65518	2	-	-	2	2	3494	42	-	-	26	5	5	-	5
24	**48.3**	23536	8602	-	-	3651	2	65517	2	-	-	2	2	800	45	-	-	15	5	5	-	4
25	**112**	88649	OoM	-	-	OoT	2	65516	OoM	-	-	OoT	2	1212	OoM	-	-	OoT	5	OoM	-	OoT
26	**1009**	OoM	-	-	-	-	2	OoM	-	-	-	-	2	OoM	-	-	-	-	5	-	-	-
27	**12783**	-	-	-	-	-	2	-	-	-	-	-	2	-	-	-	-	-	5	-	-	-
28	OoT	-	-	-	-	-	OoT	-	-	-	-	-	OoT	-	-	-	-	-	OoT	-	-	-
7	20%	40%	-17%	-90%	25%	-92%	0%	33%	0%	100%	0%	0%	0%	0%	0%	64%	0%	0%	0%	0%	0%	0%
10	20%	59%	-8%	-90%	-0%	-98%	0%	367%	0%	567%	0%	0%	0%	6%	0%	536%	0%	0%	0%	0%	0%	0%
15	-30%	210%	-0.5%	3872%	-24%	-11%	0%	14386%	0%	7620%	0%	0%	0%	213%	0%	5565%	0%	38%	0%	0%	0%	0%
20	-26%	453%	-9%	-	-	-3%	0%	818725%	0%	-	-	0%	0%	0%	0%	-	-	18%	0%	0%	-	0%
21	-12%	2551%	-7%	-	-	-7%	0%	2183900%	0%	-	-	0%	0%	0%	0%	-	-	-12%	0%	0%	-	0%
22	-24%	285%	-6%	-	-	-2%	0%	503892%	0%	-	-	0%	0%	300%	0%	-	-	0%	0%	0%	-	0%
23	-18%	195%	-4%	-	-	7%	0%	436687%	0%	-	-	0%	0%	5%	0%	-	-	24%	0%	0%	-	0%
24	-28%	1261%	3%	-	-	16%	0%	2183800%	0%	-	-	0%	0%	0%	0%	-	-	50%	0%	0%	-	0%
25	-35%	3702%	-	-	-	-	0%	545867%	-	-	-	-	0%	165%	-	-	-	-	0%	-	-	-
26	-4%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-
27	2%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-	-	-	0%	-	-	-
28	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 5.15: Computing the nucleolus of type V games – no linear speed-up



Looking at Figure 5.15, especially in comparison to Figure 5.10, one can somewhat surprisingly notice, that while computation times are drastically increased for *DK*, apart from that single outlier for 22 players, it seems to actually scale more stably without performing the linear speed-up, than in the original setting. However, producing extremely high[3] computation times more stably is questionable whether to regard as an advantage: on one hand we have a better grasp on what to expect from the method in terms of performance than in the previous setting, however, numerical results indicate that omitting the linear speed-up for *DK* when computing the nucleolus of instances with extremely large tight sets is not beneficial.

Again, these games are more interesting from a theoretical point of view, as finding their nucleoli is trivial. However, this computational experiment shows the sensitivity of certain methods to linear speed-up. When we expect to deal with very large tight sets, certain methods are mostly benefiting from simply omitting the task, while this modification makes purely dual methods unusable.

Finally we return to evaluate whether the additional LPs occurring in the computation of the nucleolus by omitting the linear speed-up indeed were rather limited in number, and were those extra LPs trivial in terms of negligible amount of additional pivots needed to solve them. In terms of the former, notice that outside of the most extreme cases (type III games for *DK*, and type V games for *DK* and *SD*), all methods on ally types of games for all player set sizes were actually below the $n-1$ limit, or at least reasonably close to it in a few exceptions: *Alg7* and *DK* for 28-player type II games and *DK* for 27- and 29-player type IV games.

[3]For a more complete comparison, we took an exception to apply time restrictions on *DK* for type V games without linear speed-up.

Table 5.19: Additional pivot/iteration ratios and time changes – no linear speed-up

		Alg7	DK	SP	SD	PD	DP
Extra pivot / extra iteration ratio	Type I	0.8%	8%	26%	43%	5%	44%
	Type II	1%	7%	46%	96%	9%	101%
	Type III	0%	18%	0%	953%	0%	0%
	Type IV	1%	6%	54%	103%	9%	99%
	Type V	0%	0.4%	0%	84%	0%	0%
	Average	**0.6%**	**8%**	**25%**	**256%**	**5%**	**49%**
Time change	Type I	-0.6%	-12%	61%	7%	51%	26%
	Type II	-0.4%	-25%	58%	57%	74%	68%
	Type III	0.5%	22%	6%	-35%	2%	-6%
	Type IV	-4%	-39%	87%	17%	85%	-18%
	Type V	-16%	1090%	-5%	1891%	-12%	-14%
	Average	**-4%**	**207%**	**41%**	**387%**	**40%**	**11%**

In terms of the latter, Table 5.19 offers a summary of the results in terms of the ratios of additional number of pivots required per additional number of iterations required, and the respective changes in computational times. The colour coding is slightly different, than it was previously, for the first part of Table 5.19: as the target being close to 0% for the ratio, we highlight these results with red, the opposite case with green, in-between cases with black.

Even though the averages contain considerable distortions from the extreme type of games like III and V, Table 5.19 seem to indicate that our expectations were clearly met in the case of *Alg7*, *DK* and *PD*, to some extent *SP* also, while it does not seem to follow through on dual based sequential LP methods *DP* and *SD*. The reason for this phenomena is an aspect of the primal-dual trade-off, that has not been explicitly analysed, neither identified by the literature: the effect of *warm-starting* on the various methods computing the nucleolus.

First note, that by the design of active-set methods *Alg7* and *DK*, effective warm-starting is automatically part of their scheme by solving the large LPs in the primal and dual sequences $(\mathbf{P^{(k)}})$ and $(\mathbf{D^{(k)}})$ without formulating them explicitly. When it comes to evaluating the same, the efficiency of warm-starting for the various sequential LP methods, note that while a solution of $\mathbf{P^{(k)}}$ is feasible in $\mathbf{P^{(k+1)}}$, and among those a solution with minimal tight set actually offers the best starting point, this does not hold true for the dual. A solution of $\mathbf{D^{(k)}}$, even with maximal support, is infeasible in $\mathbf{D^{(k+1)}}$. Having the solution for the previous LP only translates to offering a lower bound on the objective function value, as some sort of warm start.

However, firstly this does not affect significantly the number of pivots, even when it comes to solving the additional *trivial* LPs, thereby leading to that while most methods (*Alg7*, *DK*, *SP* and *PD*) produce marginal increase in the number of pivots for the additional iterations, this is not the case for the dual sequential methods *SD* and *DP*.

Moreover, a lower bound on the objective function is available for primal sequential LP methods as well, simply by the design of the objective function: $\varepsilon^{(k)*} > \varepsilon^{(k+1)*}$.

The literature generally regards dual methods to be the beneficiary of the classical trade-off between primal and dual sequential LP methods (cf. Remark 2.7), and comparing the numerical results of *SP* and *SD* in Tables 5.6-5.10 we have found some evidence of this being correct. Subsequently we introduce *DP*, that dissolves that classical primal-dual trade-off, again with plenty of evidence in the numerical results of Section 5.2. However, the effect of efficient warm-starting is another, overlooked aspect of the very same primal-dual trade-off, that also clears the picture of why we see both *SP* and *PD* outperforming both *SD* and *DP* for 24-player type II and type IV games. Moreover, warm-start is also the reason why *SP* and *PD* can handle efficiently the trivial additional LPs appearing with omitting linear speed-up, leading to the results in Table 5.19.

We close by noting that again, while the extreme case of type V games highly distort the average time changes in Table 5.19, it is clear that in many cases it is beneficial not to perform the hard task of finding all coalitions outside of the linear span of the settled coalitions, as omitting can lead to a, sometimes quite significant decrease in computational times.

Overall, decrease in computation time rarely, only in the most peculiar settings seem to appear for primal sequential methods *SP* and *PD*, indicating that mostly dual methods and *Alg7* can benefit from omitting the linear speed-up from these algorithms, despite the issues with warm starting dual sequential methods. However, one has to be careful, as dual methods can also suffer greatly from the change.

Note that the driving force of decrease in computation time is not performing the difficult task of finding all coalitions within the span of settled coalitions. Since with linear speed-up we perform this in every iteration, it is not surprising that primal methods (and *DP*) can benefit less from dropping this task, since they require minimal number of iterations by design, as opposed to purely dual methods. Therefore it is expected that when *DK* benefits from the modification, it benefits more compared to *Alg7*. However, as this is apparent from Tables 5.14-5.18, outside of the smallest instances it is still *Alg7* producing the best computation times by far.

It is noteworthy, that decrease in computation times mostly happens to the methods that we could regard the best in certain aspects: the two active-set methods and dual-primal hybrid sequential LP method. Especially remarkable is that while our proposed method *Alg7* produced the best computational times in most cases already with linear speed-up, and were capable of solving larger instances than the other methods, on average it's computational time further decreases with omitting the linear speed-up for almost every type of games, with the sole exception being the trivially handled type III games.

Chapter 6

Conclusions

In this thesis we considered the computation of one of most widespread single-valued solution concept of cooperative game theory (with transferable utilities), the *nucleolus*. Finding the nucleolus is essentially solving an optimisation problem, where the solution space is an $(n-1)$-dimensional hyperplane, while the objective function is in a much higher dimensional space, and the means of optimality is lexicographic.

While the optimisation problem itself is immensely complex, the technique of formalising the same problem as solving a sequence of (large) LPs have appeared practically together with the solution concept, and became the dominant, overwhelmingly successful approach thanks to the continuous development by the literature over the past five decades. We presented a comprehensive literature review, primarily focusing on methods that can compute the nucleolus of a general cooperative game, without assuming any structure on the characteristic function.

Moreover, we categorise these methods as primal and dual not only and necessarily by the large LPs each method use to formalise the LP sequence, but more so involving the manner of the updating procedure in between two LPs in the sequence: that is, whether a given method find the entire minimal tight set, guaranteeing maximal rank increase and thereby minimal number of iterations; or whether it suffices for a given method to identify part of the minimal tight set, as long as it guarantees *some* increase in the rank of the settled collection of coalitions eventually identifying the nucleolus, since both approaches lead to the same worst-case number of iterations being at most $n-1$. The classical trade-off between primal and dual methods is centred around this disparity in the number of iterations required by the two type of methods.

To dissolve this trade-off, we introduce hybrid variants of the classical sequential LP algorithms, that can achieve minimal number of iterations required while dealing with the computationally preferable type of LPs at the same time. We find that while this *dual-primal* hybrid variant seems the best approach in most of the cases, it basically has

the same limitations as other sequential LP methods in terms of player set sizes; moreover, in certain cases, as approaching that limiting boundary can even be outperformed by other, *primal* variants, indicating that while we managed to eliminate *a* classical compromise between primal and dual methods, we have not yet identified every aspect of that trade-off.

Somewhat independently of this, we introduced a new conceptual approach to one of the main tasks involved in the computation of the nucleolus, that requires finding all coalitions within the linear span of a collection of coalitions, and that guarantees the rank increase of the collection through the iterations, leading eventually to the $(n-1)$ bound on the number of iterations required. We argued, that in practice, it can be beneficial not to perform this difficult task, leading to exponential number of iterations worst case, but on average, we have reason to expect a decrease in computation time, despite required number of iterations, pivots and subroutine iterations are all increasing. The behaviour of certain methods lead to the identification of that new aspect in the primal-dual trade-off, namely *warm starting*.

The main contribution of the thesis, however, is the development of a new, constructive approach: an active-set method that serves as the state-of-the-art implementation of the simplex method for solving the sequence of LPs involved in the computation of the nucleolus. It is a primal based approach with a dual subroutine, that guarantees minimal number of iterations, and while it benefits from efficient warm-starting, it does not suffer from computational disadvantage from the primal LPs, as it solves the large LPs without explicit formulation, rather focusing on dual relaxations and small linear systems.

The resulting method outperforms sequential LP algorithms, as well as Derks and Kuipers (1997)'s method, that in certain aspects we could view as the dual counterpart of our proposed algorithm. Moreover, the results on balancedness laying the foundations of the new algorithm allow us to improve on the sole existing verification method for the nucleolus, the Kohlberg criterion.

Finally, we regard a valuable contribution of providing extensive numerical results comparing these algorithms, but even more so, the online repository Benedek (2018) containing all of the algorithms involved freely available as open source codes, together with the sizeable set of instances used for producing the numerical results. We regard both this online repository, both developing open source code as a long term project, with continuous contributions, from which the entire field could benefit. Moreover, ideally this could be the gateway to those, outside of the field, who apply the solution concept and rely on these computations. Our proposed algorithm is a powerful asset that, by all means, one should use whenever facing a cooperative game without the favourable structure allowing to use poly-time algorithms. In terms of practical applicability, the new

algorithm being easily able to adapt to available characterisation sets is an additional favourable property.

An aspect only briefly mentioned so far is the question of what solver one could use, or more precisely how dependent each approach is on the available solvers. For example the various possibilities offered in Section 2.3 allow us not to necessarily use interior point methods when solving large LPs involved in the primal or dual sequence, when looking for the minimal tight set. A related, more significant problem is that classical sequential LP methods are much more dependent on an efficient LP solver, which are mostly commercial products not available for the public. On the contrary for the active-set methods of Chapter 3, focusing on much smaller linear systems and LPs, consequently being almost independent of the available solver. That sort of (almost) solver-independency is again an attractive feature.

We close the conclusion by outlining possible future directions of research. Beyond the overall development of the online repository with various other methods and options, we feel that all the possibilities of the new method are not yet fully exploited. The first challenge of the new approach lies in finding the tight sets efficiently. As long as this task involves checking all the coalitions' excesses in $\mathcal{N}$, our possibilities for improvement are heavily limited. Once resolved, subsequently one has to also find a way to carry out similarly challenging step size calculations efficiently. Applying heuristic approaches can possibly enable us to tackle instances of much larger size, while from another angle it would be of great importance to find structured games such that these tasks can be performed more efficiently.

Appendix A

Numerical results of the Kohlberg algorithms

As we mentioned in Section 5.2, in the followings we focus our attention to the performance of the various Kohlberg algorithms (*Kohlberg*, *IKA* and *IKAcr*) on solutions randomly chosen from the imputations set, the least core, and the so-called *least-least core* (with $\mathcal{T}^1 \cup \mathcal{T}^2$ being $\mathcal{T}^0$-balanced). Also as we mentioned, rejecting a random imputation being trivial, and the original Kohlberg algorithm unusable even for moderate sized games ($n \geq 15$), when analysing the results we mainly compare *IKA* and *IKAcr* on the remaining two solutions. The notation is in line with Section 4.1.

A.1 Type I and II games

We observe that other solutions from the least core greatly resemble what we have noticed in Tables 5.1 and 5.2 with the nucleolus. As the number of iterations are the same for *IKA* and *IKAcr*, since there is only a slight edge in the number of subroutines

Table A.1: Original and improved Kohlberg algorithms on type I games and random imputation

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0006	0.0006	**0.0004**	1	1	1	1	1	1	0
10	0.022	0.0011	**0.0009**	1	1	1	1	1	1	0
15	0.024	**0.023**	0.023	1	1	1	1	1	1	0
20	0.94	**0.92**	0.95	1	1	1	1	1	1	0
25	36.6	**36**	36.7	1	1	1	1	1	1	0
26	**75.7**	76	76	1	1	1	1	1	1	0
27	158	**157**	158	1	1	1	1	1	1	0
28	754	755	**746**	1	1	1	1	1	1	0
29	122	110	**107**	1	1	1	1	1	1	0
30	472	454	**446**	1	1	1	1	1	1	0

Table A.2: Original and improved Kohlberg algorithms on type II games and random imputation

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0006	0.0003	**0.0001**	1	1	1	1	1	1	0
10	0.0018	0.0009	**0.0007**	1	1	1	1	1	1	0
15	0.024	0.023	**0.023**	1	1	1	1	1	1	0
20	0.95	**0.92**	0.93	1	1	1	1	1	1	0
25	36.6	**36**	36.9	1	1	1	1	1	1	0
26	**75.4**	76.3	75.8	1	1	1	1	1	1	0
27	158	**157**	158	1	1	1	1	1	1	0
28	**728**	777	752	1	1	1	1	1	1	0
29	113	108	**108**	1	1	1	1	1	1	0
30	**456**	492	464	1	1	1	1	1	1	0

Table A.3: Original and improved Kohlberg algorithms on type I games and least core solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.012	0.0013	**0.0011**	6.9	1.8	1.8	12.2	2.6	2	1.2
10	1.3	**0.0026**	0.003	385.6	1.8	1.8	527.8	2.5	1.8	1.2
15	OoT	**0.052**	0.052	OoT	1.6	1.6	OoT	2.2	1.6	1
20	-	2.9	**2.9**	-	1.6	1.6	-	2.3	1.7	1.1
25	-	124	**124**	-	1.6	1.6	-	2.3	1.7	1.3
26	-	235	**228**	-	1.5	1.5	-	1.9	1.5	1.2
27	-	500	**487**	-	1.6	1.6	-	2.1	1.6	1.1
28	-	2459	**2289**	-	1.7	1.7	-	2.2	1.7	1.2
29	-	543	**529**	-	1.1	1.1	-	1.5	1.4	1
30	-	**1601**	1652	-	1.1	1.1	-	1.5	1.3	72842

Table A.4: Original and improved Kohlberg algorithms on type II games and least core solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0026	**0.0011**	0.0013	4.7	2.2	2.2	8.2	3.1	2.2	0.7
10	1.7	0.0025	**0.0023**	383	2	1.9	792	2.9	1.9	1
15	OoT	0.081	**0.081**	OoT	2	2	OoT	3.1	2.4	1.4
20	-	3.5	**3.5**	-	1.9	1.9	-	3	2.1	1.2
25	-	111	**110**	-	1.8	1.8	-	2.9	2.1	1.3
26	-	313	**302**	-	1.9	1.9	-	2.7	2.1	1.3
27	-	534	**526**	-	1.8	1.8	-	2.7	1.9	1.1
28	-	2491	**2318**	-	2	2	-	3.3	2.4	1.4
29	-	216	**214**	-	1	1	-	2	2	1
30	-	1327	**1299**	-	1.1	1.1	-	2	1.9	1.1

Table A.5: Original and improved Kohlberg algorithms on type I games and least-least core solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.01	0.0016	**0.0013**	18.7	2.5	2.5	30.9	3.2	2.5	2.2
10	1.9	**0.003**	0.0038	672	2.2	2.2	909	2.9	2.2	1.9
15	OoT	**0.062**	0.063	OoT	1.8	1.8	OoT	2.4	1.8	1.7
20	-	3.2	**3.2**	-	1.7	1.7	-	2	1.7	1.7
25	-	147	**146**	-	1.8	1.8	-	2.3	1.8	1.6
26	-	259	**250**	-	1.6	1.6	-	1.8	1.6	1.5
27	-	603	**587**	-	1.8	1.8	-	2.3	1.8	1.7
28	-	2778	**2557**	-	1.9	1.9	-	2.3	1.9	1.8
29	-	**1302**	1317	-	1.4	1.4	-	1.8	1.6	1.4
30	-	**2686**	2750	-	1.3	1.3	-	1.6	1.4	72842

Table A.6: Original and improved Kohlberg algorithms on type II games and least-least core solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.009	**0.0015**	**0.0015**	17.1	2.7	2.6	31.4	3.8	2.6	1.4
10	1.9	**0.003**	**0.003**	483	2.4	2.3	1005	3.6	2.5	1.7
15	OoT	0.1	**0.1**	OoT	2.4	2.4	OoT	3.4	2.5	1.7
20	-	4.7	**4.7**	-	2.3	2.3	-	3.3	2.4	2
25	-	**211**	212	-	2.7	2.7	-	4.3	2.9	2.1
26	-	451	**443**	-	2.5	2.5	-	3.2	2.7	2.4
27	-	817	**733**	-	2.3	2.3	-	3.2	2.6	2.2
28	-	3536	**3307**	-	2.9	2.9	-	4.6	3.1	2.2
29	-	**2312**	2418	-	2	2	-	3.5	2.8	2
30	-	**5714**	5867	-	2	2	-	3	2.7	2

for *IKAcr* in type I games, we can note that *IKAcr* is still faster in most of the cases (especially for the least core), but the difference is not very significant either way.

Even though the slightest of difference between *IKA* and *IKAcr* in iterations appears for type II games, we observe practically the same regardless of solutions from the least or the least-least core. The outlier values of coalitions saved from storage by *IKAcr* at the 30-player type I games are due to a single game producing a tight set magnitudes larger than the average size, but this particular instance does not effect greatly the overall picture.

A.2 Type III and IV games

As type III games are rather trivial for primal methods (such as the various Kohlberg algorithms), it is of no surprise to find in Table 5.3 that the verification of type III games requires only 1 iteration from *IKA* and *IKAcr*. Bearing this in mind, it is evident that the results for type III games completely coincide in the case of least, least-least core and the nucleolus: the least core solution is already the nucleolus. Therefore, in order to

Table A.7: Original and improved Kohlberg algorithms on type III games and random imputation

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0007	**0.0003**	0.0006	1	1	1	1	1	1	0
10	0.61	**0.0011**	0.0014	1	1	1	1	1	1	0
15	0.024	0.023	**0.023**	1	1	1	1	1	1	0
20	0.94	**0.92**	0.93	1	1	1	1	1	1	0
25	36.5	**36**	36.9	1	1	1	1	1	1	0
26	**75.4**	76.2	75.7	1	1	1	1	1	1	0
27	158	**157**	158	1	1	1	1	1	1	0
28	**602**	623	670	1	1	1	1	1	1	0
29	116	**110**	114	1	1	1	1	1	1	0
30	235	**225**	227	1	1	1	1	1	1	0

Table A.8: Original and improved Kohlberg algorithms on type IV games and random imputation

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0006	0.0004	**0.0003**	1	1	1	1	1	1	0
10	0.0021	0.001	**0.0008**	1	1	1	1	1	1	0
15	0.024	**0.023**	0.023	1	1	1	1	1	1	0
20	0.94	**0.93**	0.93	1	1	1	1	1	1	0
25	36.5	**36**	36.9	1	1	1	1	1	1	0
26	**75.5**	76.2	75.7	1	1	1	1	1	1	0
27	158	**157**	158	1	1	1	1	1	1	0
28	**734**	736	738	1	1	1	1	1	1	0
29	116	110	**108**	1	1	1	1	1	1	0
30	436	448	**429**	1	1	1	1	1	1	0

Table A.9: Original and improved Kohlberg algorithms on type IV games and least core solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.0041	0.001	**0.001**	7.7	1.6	1.6	13.7	2.7	1.8	1.9
10	0.98	**0.0024**	0.0025	196	1.8	1.8	414	3.2	2.3	3.3
15	OoT	**0.056**	0.057	OoT	1.6	1.6	OoT	3.2	2.6	6
20	-	2.4	**2.4**	-	1.6	1.6	-	3.6	3.7	15.3
25	-	129	**129**	-	2	2	-	4.4	3.1	10.9
26	-	283	**275**	-	2	2	-	4.1	3.7	9.6
27	-	642	**635**	-	2.1	2.1	-	4.6	3.4	14.9
28	-	2482	**2403**	-	2	2	-	4.6	3.8	13.3
29	-	**416**	422	-	1.1	1.1	-	3.4	3.2	14.5
30	-	1454	**1405**	-	1.1	1.1	-	3.3	3	16.6

avoid unnecessary duplication, we omitted repeating tables for the least and least-least core solution with identical content to Table 5.3.

The results for type IV games are very similar to type II, with a little more distinguishment in the number of subroutines due to the increased added value of the compact representation of large tight sets in these games.

Table A.10: Original and improved Kohlberg algorithms on type IV games and least-least core solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
5	0.01	**0.0009**	0.0009	14.1	1.6	1.6	24.5	2.6	1.8	2.8
10	1.9	**0.0025**	0.0027	225	1.9	1.9	478	3.3	2.4	3.6
15	OoT	0.073	**0.073**	OoT	1.9	1.9	OoT	3.6	2.8	9.3
20	-	3.7	**3.6**	-	2.1	2.1	-	4.3	4	17.6
25	-	**199**	204	-	2.7	2.7	-	5.7	3.7	14.1
26	-	470	**455**	-	2.8	2.8	-	5.3	4.4	14.6
27	-	954	**946**	-	2.8	2.8	-	5.8	4	16
28	-	3569	**3360**	-	2.8	2.8	-	5.8	4.3	15.8
29	-	**2576**	2666	-	2.1	2.1	-	5.1	4.5	23.3
30	-	**5888**	6003	-	2	2	-	4.9	4	24.1

Table A.11: Original and improved Kohlberg algorithms on UN Security Council game and random imputation

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
7	**0.001**	**0.001**	0.019	1	1	1	1	1	1	0
10	**0.001**	**0.001**	**0.001**	1	1	1	1	1	1	0
15	0.024	0.024	**0.022**	1	1	1	1	1	1	0
20	0.99	0.96	**0.94**	1	1	1	1	1	1	0
25	38.8	**37.3**	37.3	1	1	1	1	1	1	0
26	81.1	88.9	**75.7**	1	1	1	1	1	1	0
27	**156**	157	162	1	1	1	1	1	1	0
28	378	**348**	351	1	1	1	1	1	1	0

A.3 UN Security Council game

As we mentioned above in Subsection 5.1.1.4, we introduce weighted voting games based on the UN Security Council voting mechanism with 5 *big* (veto) players, while the rest of the players are *small*. We expect that through the extremely large tight sets, these games not only challenge our proposed method of Algorithm 7, but capable of distinguishing between the two improved Kohlberg algorithms for the very same reason, the extremely large tight sets.

As we can see in Table 5.5, since *IKA* and *IKAcr* need only 2 iterations to verify the nucleolus, it is the only element in the least-least core, therefore we omit the table for the latter, to avoid unnecessary duplication.

While in Table 5.5 we only find a somewhat questionable distinguishment between *IKA* and *IKAcr*, that is one going out of memory, while the other is going out of time for 28-players, we find a clearer picture in Table A.12. Thanks to the intensive use of compact representation, *IKAcr* is capable of rejecting an element of the least core, that is not the nucleolus, for 28 players as well, while *IKA* runs out of memory for this task already. This is encouraging for *IKAcr* being capable of verification in games with large tight sets, when *IKA* becomes unusable.

Table A.12: Original and improved Kohlberg algorithms on UN Security Council game and least core solution

n	Time			Iterations			Subroutines			Repr.
	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	Kohlberg	IKA	IKAcr	IKAcr
7	**0.001**	0.14	0.19	2	2	2	5	4	3	3
10	0.003	**0.002**	0.081	2	2	2	7	5	4	43
15	**0.06**	0.081	0.13	2	2	2	5	4	3	1658
20	2.8	3.3	**2.8**	2	2	2	7	5	4	49147
25	**112**	134	130	2	2	2	5	4	3	1665238
26	1259	1570	**248**	2	2	2	7	5	4	3145723
27	OoT	4760	**2410**	OoT	2	2	OoT	4	3	6644168
28	-	OoM	**5967**	-	OoM	2	-	OoM	4	12582907
29	-	-	OoT	-	-	OoT	-	-	OoT	OoT

Bibliography

Fred Alois Behringer. A simplex based algorithm for the lexicographically extended linear maxmin problem. *European Journal of Operational Research*, 7(3):274–283, 1981.

Márton Benedek. Nucleolus. *GitHub repository*, https://github.com/blrzsvrzs/nucleolus, 2018.

Márton Benedek, Jörg Fliege, and Tri-Dung Nguyen. Finding and verifying the nucleolus of cooperative games. *University of Southampton*, https://eprints.soton.ac.uk/426473/, 2018.

Joseph Frédéric Bonnans and Matthieu André. Fast computation of the leastcore and prenucleolus of cooperative games. *RAIRO-Operations Research*, 42(3):299–314, 2008.

Siegfried Brune. On the regions of linearity for the nucleolus and their computation. *International Journal of Game Theory*, 12(1):47–80, 1983.

Guido Bruyneel. Computations of the nucleolus of a game by means of minimal balanced sets. *Operations Research Verfahren*, 34:35–51, 1979.

Jean Derks and Jeroen Kuipers. Implementing the simplex method for computing the prenucleolus of transferable utility games. *University of Maastricht*, https://www.researchgate.net/publication/265080719, 1997.

Irinel Dragan. A procedure for finding the nucleolus of a cooperativen person game. *Zeitschrift für Operations Research*, 25(5):119–131, 1981.

E. Elkind, L.A. Goldberg, P. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *Proceeding of the National Conference On Artificial Intelligence*, volume 22, page 718, 2007.

Edith Elkind and Dima Pasechnik. Computing the nucleolus of weighted voting games. *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 327–335, 2009.

Robert M Freund, Robin Roundy, and Michael J Todd. Identifying the set of always-active constraints in a system of linear inequalities by a single linear program robert m. freund. *Sloan W.P.*, (1674-85), 1985.

Bastian Fromen. Reducing the number of linear programs needed for solving the nucleolus problem of n-person game theory. *European Journal of Operational Research*, 98 (3):626–636, 1997.

Alan J Goldman and Albert William Tucker. *Theory of Linear Programming*, volume Annals of Mathematics Study No. 38, chapter Linear Inequalities and Related Systems, pages 53–98. 1956.

Jacek Gondzio. Interior point methods 25 years later. *European Journal of Operational Research*, 218(3):587–601, 2012.

Daniel Granot, Frieda Granot, and Weiping R Zhu. Characterization sets for the nucleolus. *International Journal of Game Theory*, 27(3):359–374, 1998.

Mario Guajardo and Kurt Jörnsten. Common mistakes in computing the nucleolus. *European Journal of Operational Research*, 241(3):931–935, 2015.

Philip Hall. On representatives of subsets. *J. London Math. Soc*, 10(1):26–30, 1935.

Åsa Hallefjord, Reidun Helming, and Kurt Jørnsten. Computing the nucleolus when the characteristic function is given implicitly: A constraint generation approach. *International Journal of Game Theory*, 24(4):357–372, 1995.

Gur Huberman. The nucleolus and the essential coalitions. In *Analysis and optimization of systems*, pages 416–422. Springer, 1980.

M Justman. Iterative processes with nucleolar restrictions. *International Journal of Game Theory*, 6(4):189–212, 1977.

Kazuo Kido, , and . A modified kohlberg criterion and a nonlinear method to compute the nucleolus of a cooperative game. *Taiwanese Journal of Mathematics*, pages 1581–1590, 2008.

E Kohlberg. On the nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 1(20):62–66, 1971.

Elon Kohlberg. The nucleolus as a solution of a minimization problem. *SIAM Journal on Applied Mathematics*, 23(1):34–39, 1972.

Alexander Kopelowitz. Computation of the kernels of simple games and the nucleolus of n-person games. Technical report, DTIC Document, 1967.

Phuoc Hoang Le, Tri-Dung Nguyen, and Tolga Bektaş. Generalized minimum spanning tree games. *EURO Journal on Computational Optimization*, 4(2):167–188, 2016.

Mingming Leng and Mahmut Parlar. Analytic solution for the nucleolus of a three-player cooperative game. *Naval Research Logistics (NRL)*, 57(7):667–672, 2010.

Michael Maschler, Bezalel Peleg, and Lloyd S Shapley. Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of operations research*, 4(4):303–338, 1979.

Tri-Dung Nguyen and Lyn Thomas. Finding the nucleoli of large cooperative games. *European Journal of Operational Research*, 3(248):1078–1092, 2016.

Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the k largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.

Jörg Oswald, Jean Derks, and Hans Peters. Prenucleolus and nucleolus of a cooperative game: characterizations by tight coalitions. 1995.

Guillermo Owen. A note on the nucleolus. *International Journal of Game Theory*, 3(2): 101–103, 1974.

Kanstantsin Pashkovich. Computing the nucleolus of weighted voting games in pseudo-polynomial time. *CoRR*, abs/1810.02670, 2018.

Bezalel Peleg and Peter Sudhölter. *Introduction to the Theory of Cooperative Games*, volume 34 of *Theory and Decision Library, Series C: Game Theory, Mathematical Programming and Operations Research*. Springer, 2 edition, 2007. ISBN 9783540729457.

Jos AM Potters, Johannes H Reijnierse, and Michel Ansing. Computing the nucleolus by solving a prolonged simplex algorithm. *Mathematics of operations research*, 21(3): 757–768, 1996.

Justo Puerto and Federico Perea. Finding the nucleolus of any n-person cooperative game by a single linear program. *Computers & Operations Research*, 40(10):2308–2313, 2013.

Hans Reijnierse and Jos Potters. The b-nucleolus of tu-games. *Games and Economic Behavior*, 24(1-2):77–96, 1998.

Johannes Herman Reijnierse. Games, graphs and algorithms. *Ph.D. Thesis, Nijmegen*, https://repository.ubn.ru.nl/handle/2066/145987, 1995.

Jayaram K Sankaran. On finding the nucleolus of an n-person cooperative game. *International Journal of Game Theory*, 19(4):329–338, 1991.

David Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal of Applied Mathematics*, 17(6):1163–1170, 1969.

Lloyd S Shapley. Markets as cooperative games. Technical report, RAND CORP SANTA MONICA CA, 1955.

Lloyd Stowell Shapley. *A value for n-person games. In: Kuhn HW and Tucker AW (eds.) Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, Princeton, 1953.

A I Sobolev. *The characterization of optimality principles in cooperative games by functional equations. In: Vorobiev NN (ed.) Mathematical Methods in the Social Sciences.*, volume 6, pages 95–151. 1975.

Tamás Solymosi. On computing the nucleolus of cooperative games. *University of Illinois*, https://www.researchgate.net/profile/Tamas_Solymosi/publication/318017147, 1993.

Tamás Solymosi and Tirukkannamangai ES Raghavan. An algorithm for finding the nucleolus of assignment games. *International Journal of Game Theory*, 23(2):119–143, 1994.

Tamás Solymosi and Balázs Sziklai. Universal characterization sets for the nucleolus in balanced games. *Institute of Economics, Centre for Economic and Regional Studies, Hungarian Academy of Sciences*, http://real.mtak.hu/23430/1/MTDP1512.pdf, 2015.

Tamás Solymosi and Balázs Sziklai. Characterization sets for the nucleolus in balanced games. *Operations Research Letters*, 44(4):520–524, 2016.

Albert William Tucker. *Dual Systems of Homogeneous Linear Relations*, volume Annals of Mathematics Study No. 38, chapter Linear Inequalities and Related Systems, pages 3–18. 1956.

LA Wolsey. The nucleolus and kernel for simple games or special valid inequalities for 0–1 linear integer programs. *International Journal of Game Theory*, 5(4):227–238, 1976.