Ann Math Artif Intell
DOI 10.1007/s10472-009-9162-5

On the computational complexity of weighted voting games

Edith Elkind · Leslie Ann Goldberg · Paul W. Goldberg · Michael Wooldridge

Abstract Coalitional games provide a useful tool for modeling cooperation in multiagent systems. An important special class of coalitional games is weighted voting games, in which each player has a weight (intuitively corresponding to its contribution), and a coalition is successful if the sum of its members' weights meets or exceeds a given threshold. A key question in coalitional games is finding coalitions and payoff division schemes that are stable, i.e., no group of players has any rational incentive to leave. In this paper, we investigate the computational complexity of stability-related questions for weighted voting games. We study problems involving the *core*, the *least core*, and the *nucleolus*, distinguishing those that are polynomial-time computable from those that are NP-hard or coNP-hard, and providing pseudopolynomial and approximation algorithms for some of the computationally hard problems.

Keywords Weighted voting games · Core · Least core · Nucleolus

Mathematics Subject Classification (2000) 91B14

1 Introduction

Coalitional games provide a simple but rich mathematical framework within which issues related to cooperation in multi-agent systems can be investigated [6, 15]. Crudely, a coalitional game can be understood as a game in which players can benefit

E. Elkind (✉)
Division of Mathematical Sciences, Nanyang Technological University,
637 371, Singapore, Singapore
e-mail: eelkind@gmail.com

L. A. Goldberg · P. W. Goldberg · M. Wooldridge
Department of Computer Science, University of Liverpool,
Liverpool L69 3BX, UK

Published online: 20 October 2009

from cooperation. More formally, a coalitional game is described by a function that assigns a value to each coalition (subset) of players: this value corresponds to the payoff that the members of this coalition can achieve by working together. The key questions in such games relate to *which coalitions will form*, and *how the benefits of cooperation will be shared*.

In more detail, an outcome of a coalitional game is a coalition or, sometimes, a coalition structure (a partition of the players into disjoint coalitions), together with an *imputation*, i.e., a payoff distribution vector which specifies how to share the value of each coalition among its members. In many settings, including the one considered in this paper, it is customary to stipulate that the *grand coalition*, i.e., the coalition that consists of all players, should form.[1] In this case, an outcome can be identified with a payoff vector that distributes the value of the grand coalition among all players.

A natural plausibility criterion for an outcome of a game is stability: given an outcome, no subset of players should be able to profit by deviating from it. There are several solution concepts that attempt to capture this intuition, the most important of which is perhaps the core. There are also other approaches to choosing the best way of distributing payoffs: for example, one can use the Shapley value, which is, in a sense, the most "fair" way to share the value of the grand coalition.

From a computational perspective, the key issues relating to coalitional games are, first, how such games should be *represented* (since the obvious representation is exponentially large in the number of players, and is hence infeasible); and second, the extent to which cooperative solution concepts can be *efficiently computed*.

A particular class of succinctly representable coalitional games is that of *weighted voting games*. A weighted voting game is one in which each player is given a numeric weight, and a coalition takes the value 1 if the sum of its weights exceeds a particular threshold, and the value 0 otherwise. Weighted voting games are widely used in practice. For example, the voting system of the European Union is a combination of weighted voting games [4]. They can also be used to model resource allocation in multiagent systems: in this case, the weight of each agent corresponds to the amount of resources available to that agent, the threshold corresponds to the total amount of resources needed to achieve a given task, and the total value of the task is normalised to 1.

The computational properties of the fairness-related solution concepts for weighted voting games (such as, e.g., the Shapley value) are well-studied (see Section 6 for an overview). However, much less is known about the complexity of the stability-related solution concepts for these games. The goal of this paper is to fill this gap. We focus on three solution concepts—the *core*, the *least core*, which is a natural relaxation of the core, and the *nucleolus*, which can be thought of as the single "most stable" payoff vector. Although the complexity of determining non-emptiness of the core has been studied for a variety of representations, comparatively little research has considered the least core and the nucleolus.

The remainder of the paper is structured as follows. Section 2 introduces the relevant concepts and definitions from coalitional game theory. Section 3 investigates the complexity of computing the core and the least core. Theorems 1 and 2 together with Corollary 1 give a polynomial-time algorithm for determining whether the core

[1] For an analysis of weighted voting games with coalition structures, see [9].

is empty and computing the nucleolus if the core is not empty. Theorem 3 gives intractability results (NP-hardness and coNP-hardness) for determining whether the ϵ-core of a weightd voting game is non-empty, determining whether a given imputation is in the ϵ-core, constructing an imputation in the ϵ-core, determining whether a given imputation is in the least core, and constructing an imputation in the least core. We mitigate these hardness results by showing, in Theorem 5, that all of these problems can be solved in *pseudopolynomial* time. That is, all of the problems can be solved in polynomial time for weighted voting games in which the weights are at most polynomially larger than the number of players. In Section 3.2 we present a fully polynomial time approximation scheme (FPTAS) for the value of the least core. Section 4 investigates the complexity of computing the nucleolus payments of agents. Theorem 7 shows that it is coNP-hard to determine whether the nucleolus payoff of a given agent is 0, which implies that it is computationally hard to compute the nucleolus payment of an agent, or to approximate this payment within any factor. Nevertheless, we show in Theorem 8 that, for a wide class of weighted voting games, it is easy to approximate the nucleolus payment of a minimal winning coalition. Section 5 extends some of our results to *k-vector weighted voting games*, a class of coalitional games with more representational power than weighted voting games. Corollary 3 gives a polynomial-time algorithm for determining whether the core of such a game is empty and for computing the nucleolus if the core is not empty. Theorem 9 gives intractability results (NP-hardness and coNP-hardness) for determining whether the ϵ-core of a k-vector weighted voting game is non-empty, determining whether a given imputation is in the ϵ-core, constructing an imputation in the ϵ-core, determining whether a given imputation is in the least core, and constructing an imputation in the least core. Theorem 11 shows that all of these problems can be solved in pseudopolynomial time (for fixed k). Finally, Theorem 10 shows that it is coNP-hard to determine whether the nucleolus payoff of a given agent is 0.

Throughout the paper, we assume some familiarity with computational complexity [19] and approximation algorithms [1].

2 Preliminary definitions

Except where explicitly stated otherwise, we assume numbers are rationals. Our results extend straightforwardly to any "sensible" representation of real numbers, but we use rationals to avoid tangential representational issues. Unless stated otherwise (specifically, Theorem 5), we assume the rational values are represented in binary. This allows us to use the machinery of polynomial-time reductions and NP-hardness. In all of our proofs, "polynomial" means "polynomial in the size of the input". Some of the problems we consider are *function problems*, rather than decision problems [19, Chapter 10]. We use the standard notion of NP-hardness for function computation, so when we say it is NP-hard to compute a function, we mean that it is NP-hard with respect to polynomial-time Turing reductions. (Thus, the existence of a polynomial-time algorithm for computing the function would imply P=NP.)

We briefly review relevant definitions from coalitional game theory [18, pp. 255–298]. A *coalitional game* consists of a set I of players, or agents, and a total function $v : 2^I \to \mathbb{R}$, which assigns a real value to every coalition (subset of the agents).

Intuitively, $\nu(S)$ is the value that could be obtained by coalition $S \subseteq I$ if its members chose to cooperate, or form a coalition. However, it does not address the question of *how* the agents cooperate to obtain this value. The *grand coalition* is the set I of all agents. A coalitional game is *monotone* if $\nu(S) \leq \nu(S \cup \{i\})$ for all $i \in I$ and all $S \subseteq I \setminus \{i\}$, and *simple* if we have $\nu(S) \in \{0, 1\}$ for all $S \subseteq I$. In a simple game, a coalition is called *winning* if $\nu(S) = 1$ and *losing* otherwise. Also, in a simple game, a player i is called a *veto* player if we have $\nu(S) = 0$ for all $S \subseteq I \setminus \{i\}$. Intuitively, veto players are the most powerful players in the game. Conversely, if a player i has no power at all, i.e., if $\nu(S) = \nu(S \cup \{i\})$ for all $S \subseteq I \setminus \{i\}$, then i is called a *dummy* player.

A *weighted voting game* is a coalitional game G given by a set of agents $I = \{1, \ldots, n\}$, their non-negative *weights* $\mathbf{w} = \{w_1, \ldots, w_n\}$, and a *threshold* T; we write $G = (I; \mathbf{w}; T)$. For a coalition $S \subseteq I$, its value $\nu(S)$ is 1 if $\sum_{i \in S} w_i \geq T$; otherwise, $\nu(S) = 0$. Without loss of generality, we assume that the value of the grand coalition $\{1, \ldots, n\}$ is 1, i.e., $\sum_{i \in I} w_i \geq T$. Note that any weighted voting game is monotone and simple. As argued in the introduction, these games can be used to model settings where the agents need to pool resources to accomplish a task, and the value of this task is normalised to 1.

An *imputation* is a division of the value of the grand coalition amongst the agents. In particular, for a weighted voting game, an imputation is a vector of non-negative numbers $\mathbf{p} = (p_1, \ldots, p_n)$, one for each agent in I, such that $\sum_{i \in I} p_i = 1$. We refer to p_i as the *payoff* of agent i: this is the total payment he receives when he participates in the grand coalition and the profits are divided according to $\mathbf{p}$. We write $w(S)$ to denote $\sum_{i \in S} w_i$. Similarly, $p(S)$ denotes $\sum_{i \in S} p_i$.

Given a coalitional game, the goal is typically to find a "fair" imputation, i.e., the share of each agent is proportional to his or her contribution, or "stable", in the sense that it provides little or no incentive for a group of agents to abandon the grand coalition and form a coalition of their own. There are many ways to formalise these ideas. These are known as *solution concepts*. In this paper, we study three solution concepts: the *core*, the *least core*, and the *nucleolus*, which we will now define.

Given an imputation $\mathbf{p} = (p_1, \ldots, p_n)$, the *excess* $e(\mathbf{p}, S)$ of a coalition S under $\mathbf{p}$ is defined as $p(S) - \nu(S)$. If a coalition's excess is positive, it means that the total payoff that its members receive under $\mathbf{p}$ is higher than what they can get on their own, while for a coalition with a negative excess the opposite is true. The *core* is a set of imputations under which the excess of each coalition is non-negative: an imputation $\mathbf{p}$ is in the *core* if for every $S \subseteq I$ we have $e(\mathbf{p}, S) \geq 0$. Informally, $\mathbf{p}$ is in the core if it is the case that no coalition can improve its payoff by breaking away from the grand coalition because its payoff $p(S)$ according to the imputation is at least as high as the value $\nu(S)$ that it would get by breaking away. The *excess vector* of an imputation $\mathbf{p}$ is the vector $\mathbf{e}(\mathbf{p}) = (e(\mathbf{p}, S_1), \ldots, e(\mathbf{p}, S_{2^n}))$, where $S_1, \ldots, S_{2^n}$ is a list of all subsets of I ordered so that $e(\mathbf{p}, S_1) \leq e(\mathbf{p}, S_2) \leq \cdots \leq e(\mathbf{p}, S_{2^n})$. In other words, the excess vector lists the excesses of all coalitions from the smallest (which may be negative) to the largest. The *nucleolus* is an imputation $\mathbf{x} = (x_1, \ldots, x_n)$ that satisfies $\mathbf{e}(\mathbf{x}) \geq_{\text{lex}} \mathbf{e}(\mathbf{y})$ for any other imputation $\mathbf{y}$, where $\geq_{\text{lex}}$ is the lexicographic order. It is known [23] that the nucleolus is well-defined (i.e., an imputation with a lexicographically maximal excess vector always exists) and is unique.

Intuitively, the nucleolus is a good imputation because it balances the excesses of the coalitions, making them as equal as possible. It is easy to see that the nucleolus

is in the core whenever the core is non-empty. Furthermore, the core is non-empty if and only if the first entry of the excess vector that corresponds to the nucleolus is non-negative.

While the notion of the core is perhaps the most natural way to capture stability in coalitional games, it suffers from a serious drawback: the core of a given game can be empty. Moreover, as we will see in the next section, this situation is quite typical for weighted voting games: it is known (see, for example, Theorem 1) that the core of a simple game is empty unless there is a veto player, a situation that is unlikely to occur in weighted voting games that arise in real-life scenarios. It is therefore natural to ask if the notion of the core can be relaxed in a way that ensures non-emptiness. A positive answer to this question is provided by the notion of the *least core*. We say that an imputation $\mathbf{p}$ is in the ϵ-core if $e(\mathbf{p}, S) \geq -\epsilon$ for all $S \subseteq I$; it is in the least core, if it is in the ϵ-core for some $\epsilon \geq 0$ and the ϵ'-core is empty for any $\epsilon' < \epsilon$. We say that the *value* of the least core is ϵ. Clearly, the least core is always non-empty and contains the nucleolus. Observe that the notion of the least core can be used to model stability in settings where there is a cost associated with deviating from the grand coalition: if this cost is larger than the value of the least core, all outcomes in the least core are likely to be stable.

We conclude this section with an example that illustrates the notions that have been introduced here.

Example 1 Consider a weighted voting game with five players $\{1, 2, 3, 4, 5\}$ whose weights are $w_1 = 2$, $w_2 = 2$, $w_3 = 2$, $w_4 = 3$, and $w_5 = 3$, and a threshold $T = 6$. In this game, both $\{1, 2, 3\}$ and $\{4, 5\}$ are winning coalitions. Therefore, the core of this game is empty, as under any imputation the total payoff to at least one of these coalitions will be at most $\frac{1}{2}$, so its excess will be $-\frac{1}{2}$ or less. On the other hand, it is not hard to construct an imputation in the $\frac{1}{2}$-core of this game: consider, for example, $\mathbf{p}^1 = (\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{4}, \frac{1}{4})$ or $\mathbf{p}^2 = (\frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{3})$. It is straightforward to verify that under both $\mathbf{p}^1$ and $\mathbf{p}^2$ the payoff of each winning coalition is at least $\frac{1}{2}$. Indeed, the coalition $\{4, 5\}$ gets payoff $\frac{1}{2}$ under both imputations. Every other winning coalition has at least three participants, so its payoff under either of these imputations is at least $3 * \frac{1}{6} = \frac{1}{2}$. We conclude that the least core of this game is equal to its $\frac{1}{2}$-core.

We will now show that $\mathbf{p}^1$ is the nucleolus of this game. First observe that $\mathbf{p}^1$ is a "more stable" imputation than $\mathbf{p}^2$. Indeed, under $\mathbf{p}^1$ there are exactly two winning coalitions that get a payoff of $\frac{1}{2}$ (namely, $\{1, 2, 3\}$ and $\{4, 5\}$) and other winning coalitions get a payoff of at least $\frac{7}{12}$. On the other hand, under $\mathbf{p}^2$ there are five winning coalitions that get a payoff of exactly $\frac{1}{2}$ (namely, $\{4, 5\}$, $\{1, 2, 3\}$, $\{1, 2, 4\}$, $\{1, 3, 4\}$, and $\{2, 3, 4\}$). Hence, the excess vector for $\mathbf{p}^1$ is lexicographically larger than that for $\mathbf{p}^2$.

Now, let $\mathbf{p}$ be the nucleolus; we will argue that $\mathbf{p} = \mathbf{p}^1$. First, if $p(\{1, 2, 3\}) > \frac{1}{2}$, then $p(\{4, 5\}) < \frac{1}{2}$, so $\mathbf{p}$ is not in the least core and hence cannot be the nucleolus. Therefore, $p(\{1, 2, 3\}) \leq \frac{1}{2}$, and by a similar argument $p(\{4, 5\}) \leq \frac{1}{2}$. Now, the first two elements of $(e(\mathbf{p}^1, S_1), \ldots, e(\mathbf{p}^1, S_{2^n}))$ are $-\frac{1}{2}$, so it must be the case that $p(\{1, 2, 3\}) = p(\{4, 5\}) = \frac{1}{2}$. Moreover, if $\mathbf{p} \neq \mathbf{p}^1$, then the third element of the excess vector for $\mathbf{p}$ must be greater than the third element of the excess vector for $\mathbf{p}^1$, i.e., $-\frac{5}{12}$. We will now show that this is impossible.

First, suppose that $p_4 \neq p_5$. As we have $p_4 + p_5 = \frac{1}{2}$, this implies that at least one of them is less than $\frac{1}{4}$; without loss of generality, assume $p_4 < \frac{1}{4}$. Now, if we have $p_1 = p_2 = p_3 = \frac{1}{6}$, then $p(\{1, 2, 4\}) < \frac{7}{12}$. If p_1, p_2, p_3 are not all equal, as $p_1 + p_2 + p_3 = \frac{1}{2}$, at least one of them (assume without loss of generality that it is p_3) exceeds $\frac{1}{6}$, so $p_1 + p_2 < \frac{1}{3}$ and, again, $p(\{1, 2, 4\}) < \frac{7}{12}$. We conclude that for $\mathbf{p}$ to be the nucleolus, it must be the case that $p_4 = p_5 = \frac{1}{4}$. Finally, suppose that $p_4 = p_5 = \frac{1}{4}$, but p_1, p_2, p_3 are not all equal. Then the largest of them (without loss of generality, p_3) satisfies $p_3 > \frac{1}{6}$, so we have $p_1 + p_2 < \frac{1}{3}$ and $p(\{1, 2, 4\}) < \frac{7}{12}$, a contradiction. We conclude that $\mathbf{p} = \mathbf{p}^1$.

3 The core and the least core

We start by considering the core—perhaps the best known and most-studied solution concept in coalitional game theory. As argued above, asking whether the core of the game is non-empty amounts to asking whether the grand coalition is stable.

Name: EmptyCore.
Instance: Weighted voting game $(I; \mathbf{w}; T)$.
Question: Is the core empty?

There is a well-known folk theorem that provides a criterion for non-emptiness of the core of a simple game. For the sake of completeness, we reproduce both the theorem and its proof here.

Theorem 1 *The core of a weighted voting game $G = (I; \{w_1, \ldots, w_n\}; T)$ is non-empty if and only if G has a veto player, i.e., there is an agent i such that $i \in \cap_{\nu(S)=1} S$.*

Proof If i is a veto player, then the allocation $x_i = 1$, $x_j = 0$ for $i \neq j$ is obviously in the core. Conversely, suppose that there are no veto players in G. Consider an arbitrary imputation $\mathbf{p} = (p_1, \ldots, p_n)$ and an agent i with $p_i > 0$. Since i is not a veto player, there is a coalition S such that $i \notin S$, $\nu(S) = 1$. We have $p(S) \leq 1 - p_i < 1$, so $e(\mathbf{p}, S) < 0$. □

We can use Theorem 1 to describe the nucleolus of a weighted voting game with non-empty core.

Theorem 2 *If the core of a weighted voting game $G = (I; \mathbf{w}; T)$ is non-empty, then its nucleolus is given by $x_i = \frac{1}{k}$ if i is a veto player and $x_i = 0$ otherwise, where k is the number of veto players in G. That is, the nucleolus shares the value of the grand coalition equally among the veto players.*

Proof Let V be the set of veto players in G. Any imputation $(p_1, \ldots, p_n)$ that has $p_i > 0$ for some $i \notin V$ is not in the core of G, as there exists a set S with $\nu(S) = 1$, $i \notin S$, for which we have $e(\mathbf{p}, S) \leq -p_i$. Hence, as the nucleolus $\mathbf{x}$ is always in the core, it satisfies $x_i = 0$ for all $i \notin V$. Now, consider a vector $\mathbf{p}$ with $p_i = 0$ for $i \notin V$,

and suppose that $p_i \neq \frac{1}{k}$ for some $i \in V$. Pick any $j \in \operatorname{argmin}\{p_i \mid i \in V\}$. We have $p_j < \frac{1}{k}$. Let $t = |\{S \mid v(S) = 1\}| + |\{S \mid S \subseteq I \setminus V\}|$. The excess vectors for $\mathbf{p}$ and $\mathbf{x}$ start with t zeros, followed by p_j and $\frac{1}{k}$, respectively. Hence, the excess vector for $\mathbf{p}$ is lexicographically smaller than the excess vector for $\mathbf{x}$. □

It follows that one can decide EmptyCore in polynomial time. Moreover, if the core is non-empty, the nucleolus can be easily computed.

Corollary 1 *There exists a polynomial-time algorithm that checks whether the core of a weighted voting game is non-empty. Moreover, if the core is non-empty, the nucleolus has polynomially-bounded rational coordinates and can be computed in polynomial time.*

Proof To verify the non-emptiness of the core, for each agent i, we check if i is present in all winning coalitions, i.e., if $w(I \setminus \{i\}) < T$. If no such agent has been found, the core is empty. Clearly, this algorithm runs in polynomial time. Similarly, to compute the nucleolus, it suffices to identify all veto players. The coordinates of the nucleolus are then rational numbers of the form $1/k$, where $k \leq n$ is the total number of veto players. □

We will now formulate several natural computational problems related to the notions of the ϵ-core and the least core.

Name: EpsilonCore.
Instance: Weighted voting game $(I; \mathbf{w}; T)$, and rational value $\epsilon \geq 0$.
Question: Is the ϵ-core of $(I; \mathbf{w}; T)$ non-empty?

Clearly, (G, ϵ) is a "yes"-instance of EpsilonCore if and only if the value of the least core of G is at most ϵ. In Corollary 2 we will show that if all weights are rational numbers, then the value of the least core is a rational number as well; moreover it can be represented using polynomially many bits.

Rather than just verifying that the ϵ-core is non-empty, one may be interested in verifying whether a particular imputation is in the ϵ-core, or in constructing an imputation in the ϵ-core.

Name: In-EpsilonCore.
Instance: Weighted voting game $(I; \mathbf{w}; T)$, rational value $\epsilon \geq 0$, and imputation $\mathbf{p}$.
Question: Is $\mathbf{p}$ in the ϵ-core of $(I; \mathbf{w}; T)$?

Name: Construct-EpsilonCore.
Instance: Weighted voting game $(I; \mathbf{w}; T)$ and rational value $\epsilon \geq 0$.
Output: An imputation $\mathbf{p}$ in the ϵ-core of $(I; \mathbf{w}; T)$ or $\perp$ if the ϵ-core of $(I; \mathbf{w}; T)$ is empty.

Moreover, we can formulate the same questions for the least core. Note that the least core is always non-empty, and therefore Construct-LeastCore is phrased somewhat differently from Construct-EpsilonCore.

Name: In-LeastCore.
Instance: Weighted voting game $(I; \mathbf{w}; T)$, imputation $\mathbf{p}$.
Question: Is $\mathbf{p}$ in the least core of $(I; \mathbf{w}; T)$?

Name: Construct-LeastCore.
Instance: Weighted voting game $(I; \mathbf{w}; T)$.
Output: An imputation $\mathbf{p}$ in the least core of $(I; \mathbf{w}; T)$.

There are obvious relationships between some of these problems. For example, if a given imputation $\mathbf{p}$ is in the least core of G, and the ϵ-core of G is non-empty, then $\mathbf{p}$ is in the ϵ-core of G (since if the ϵ-core of a game is non-empty, it contains the least core of that game). Moreover, there is a similar relationship between Construct-LeastCore and Construct-EpsilonCore. Also, Construct-EpsilonCore can be seen as a generalisation of EpsilonCore. However, being able to construct an imputation in the ϵ-core (respectively, the least core) of a game does not provide us with a direct method for checking whether an *arbitrary* imputation is in the ϵ-core (respectively, the least core) of that game. Therefore, in what follows, we study the computational complexity of these problems independently (though using similar constructions).

We will now show that all the 5 problems listed above are computationally hard. We reduce from the well-known NP-complete Partition problem, in which we are given positive integers $a_1, \ldots, a_n$ such that $\sum_{i=1}^{n} a_i = 2K$, and asked whether there is a subset of indices J such that $\sum_{i \in J} a_i = K$ [13, p. 223].

Given an instance $(a_1, \ldots, a_n; K)$ of Partition, let $I = \{1, \ldots, n, n+1\}$ be a set of agents. Let $G = (I; \mathbf{w}; T)$ be the weighted voting game with $T = K$, $w_i = a_i$ for $i = 1, \ldots, n$ and $w_{n+1} = K$. We will use the following lemmas.

Lemma 1 *For a "yes"-instance of* Partition*, the value of the least core of G is $\frac{2}{3}$, and any imputation $\mathbf{q} = (q_1, \ldots, q_{n+1})$ in the least core satisfies $q_{n+1} = \frac{1}{3}$.*

Proof Consider the imputation $\mathbf{p}$ given by $p_i = \frac{w_i}{3K}$ for $i = 1, \ldots, n+1$ (this is a valid imputation, as $\sum_{i=1}^{n+1} w_i = 3K$). For any set S with $v(S) = 1$ we have $\sum_{i \in S} w_i \geq K$, so $\sum_{i \in S} p_i \geq \frac{1}{3}$ and $e(\mathbf{p}, S) \geq -\frac{2}{3}$; for any set S with $v(S) = 0$ we have $e(\mathbf{p}, S) \geq 0$. We conclude that the value of the least core of G is at most $\frac{2}{3}$.

On the other hand, for a "yes"-instance of Partition, there are three disjoint coalitions in I that have value 1: $S_1 = J$, $S_2 = \{1, \ldots, n\} \setminus J$, and $S_3 = \{n+1\}$. Any imputation $\mathbf{p}$ such that $p_{n+1} \neq \frac{1}{3}$ has $p(S_i) < \frac{1}{3}$ for some $i = 1, 2, 3$ and hence $e(\mathbf{p}, S_i) < -\frac{2}{3}$. Hence, any imputation $\mathbf{q}$ that maximises the minimum excess satisfies $q_{n+1} = \frac{1}{3}$. Consequently, the value of the least core, ϵ, satisfies $\epsilon = \frac{2}{3}$, and any imputation in the least core has $q_{n+1} = \frac{1}{3}$. □

Lemma 2 *For a "no"-instance of* Partition*, the value of the least core of G is at most $\frac{2}{3} - \frac{1}{6K}$ and any imputation $\mathbf{q}$ in the least core satisfies $q_{n+1} \geq \frac{1}{3} + \frac{1}{6K}$.*

Proof We will start with the imputation $p_i = \frac{w_i}{3K}$ defined in the proof of the previous lemma, and modify it so as to ensure that for a new imputation $\mathbf{p}'$, the excess of each coalition is at least $-\frac{2}{3} + \frac{1}{6K}$. The imputation $\mathbf{p}'$ will serve as a witness that the value of the least core of G is at most $\frac{2}{3} - \frac{1}{6K}$. Consequently, for any imputation $\mathbf{q}$ in the least core we have $e(\mathbf{q}, S) \geq -\frac{2}{3} + \frac{1}{6K}$ for all $S \subseteq I$. In particular, taking $S = \{n+1\}$, we obtain $q_{n+1} \geq \frac{1}{3} + \frac{1}{6K}$.

The imputation $\mathbf{p}'$ is defined as follows: $p'_i = p_i - \frac{1}{6nK}$ for $i = 1, \ldots, n$, $p'_{n+1} = p_{n+1} + \frac{1}{6K}$. To see that $\mathbf{p}'$ is a valid imputation, note that $\sum_{i \in I} p'_i = \sum_{i \in I} p_i = 1$, and $p'_i = \frac{w_i}{3K} - \frac{1}{6nK} > 0$. Now, consider any set S such that $v(S) = 1$. If $S \subseteq \{1, \ldots, n\}$, as our game was constructed from a "no"-instance of Partition, we have $\sum_{i \in S} w_i \geq K + 1$. Hence,

$$\sum_{i \in S} p'_i = \sum_{i \in S} \left(\frac{w_i}{3K} - \frac{1}{6nK} \right) \geq \frac{K+1}{3K} - \frac{|S|}{6nK}$$
$$\geq \frac{1}{3} + \frac{1}{3K} - \frac{1}{6K} = \frac{1}{3} + \frac{1}{6K}.$$

Consequently, $e(\mathbf{p}', S) \geq -\frac{2}{3} + \frac{1}{6K}$. On the other hand, if $n + 1 \in S$, we have $p'(S) \geq \frac{1}{3} + \frac{1}{6K}$. □

Theorem 3 *The decision problems* EpsilonCore *and* In-EpsilonCore *are* coNP*-hard. The decision problem* In-LeastCore *is* NP*-hard. The function problems* Construct-EpsilonCore *and* Construct-LeastCore *are* NP*-hard.*

Proof By combining Lemmas 1 and 2, we conclude that the $\frac{2}{3} - \frac{1}{6K}$-core of $G = (I; \mathbf{w}; T)$ is nonempty if and only if we started with a "no"-instance of Partition. Also, the imputation $\mathbf{p}'$ constructed in the proof of Lemma 2 ($p'_i = \frac{w_i}{3K} - \frac{1}{6nK}$ for $i = 1, \ldots, n$, $p'_{n+1} = \frac{1}{3} + \frac{1}{6K}$) is in the $\frac{2}{3} - \frac{1}{6K}$-core if and only if we started with a "no"-instance of Partition. Hence, both EpsilonCore and In-EpsilonCore are coNP-hard.

The imputation $\mathbf{p}$, where $p_i = \frac{w_i}{3K}$, $i = 1, \ldots, n$, $p_{n+1} = \frac{1}{3}$, is in the least core if and only if the game G was constructed from a "yes"-instance of Partition. Hence, In-LeastCore is NP-hard.

By solving Construct-EpsilonCore with $\epsilon = \frac{2}{3} - \frac{1}{6K}$, we can solve Partition as follows: if our algorithm outputs $\perp$, then we started with a "yes"-instance of Partition, and if it outputs an imputation, we started with a "no"-instance of Partition. Similarly, if we can construct an imputation $\mathbf{p}$ in the least core, we can solve Partition by looking at its last component p_{n+1}: if $p_{n+1} = \frac{1}{3}$, we started with a "yes"-instance of Partition, and if $p_{n+1} > \frac{1}{3}$, we started with a "no"-instance of Partition. □

We can prove a matching upper bound on the complexity of one of these problems. Namely, we can show that In-EpsilonCore is in coNP, and therefore coNP-complete. To see this, note that one can demonstrate that an imputation $\mathbf{p}$ is not in the ϵ-core of the game G by guessing a set S such that $e(\mathbf{p}, S) < -\epsilon$.

For EpsilonCore, the situation is somewhat different. To show that the ϵ-core of a game G is non-empty, we can guess an imputation $\mathbf{p}$ and then show that *for all* coalitions S we have $e(\mathbf{p}, S) \geq -\epsilon$. If we could prove that it is enough to restrict our attention to imputations with polynomial bit complexity, this argument would show that EpsilonCore is in Σ_2^p. To see that this is indeed the case, note that if the ϵ-core of the game is non-empty then it contains the least core, which, by Corollary 2 (see the next section), contains an imputation with polynomial-size rational values.

This imputation can be used as a witness that the ϵ-core is non-empty. Hence, EpsilonCore is indeed in Σ_2^p.

Similarly, we will now show that In-LeastCore is in Π_2^p. We will do so by proving that the complementary problem, i.e., checking that a given imputation $\mathbf{p}$ is *not* in the least core of G is in Σ_2^p. This follows from the fact that, to demonstrate that $\mathbf{p}$ is not in the least core, it suffices to produce another imputation $\mathbf{q}$ whose worst-case performance is better than that of $\mathbf{p}$. In other words, $\mathbf{p}$ is not in the least core if and only if there exists an imputation $\mathbf{q}$ and a set S such that for all $T \subseteq I$ we have $e(\mathbf{q}, T) > e(\mathbf{p}, S)$.

Indeed, suppose that $\mathbf{p}$ is not in the least core of G. Let ϵ be the value of the least core. Then the least core contains an imputation $\mathbf{q}$ that satisfies $e(\mathbf{q}, T) \geq -\epsilon$ for all T, whereas there exists an S such that $e(\mathbf{p}, S) < -\epsilon$. Observe that we can choose our witness imputation $\mathbf{q}$ to be the element of the least core constructed in the proof of Theorem 5; by Corollary 2 it has polynomial bit complexity.

Conversely, suppose that $\mathbf{p}$ is in the least core of G. We will now show that for any imputation $\mathbf{q}$ and any set $S \subseteq I$, there exists a $T \subseteq I$ such that $e(\mathbf{q}, T) \leq e(\mathbf{p}, S)$. Indeed, let ϵ be the value of the least core. We have $e(\mathbf{p}, S) \geq -\epsilon$ for all $S \subseteq I$. Now, suppose that there exists an imputation $\mathbf{q}$ such that for some $S \subseteq I$ and any $T \subseteq I$ we have $e(\mathbf{q}, T) > e(\mathbf{p}, S)$. This would imply $e(\mathbf{q}, T) > -\epsilon$ for all T, a contradiction with ϵ being the value of the least core.

We summarise our results in the following theorem.

Theorem 4 *The problem* In-EpsilonCore *is in* coNP *(and hence* coNP*-complete). The problem* EpsilonCore *is in* Σ_2^p *and the problem* In-LeastCore *is in* Π_2^p.

3.1 Pseudopolynomial time algorithm for the ϵ-core and the least core

We will now describe a pseudopolynomial time algorithm for the problems introduced in the previous section. This means that all five problems can be solved in polynomial time if the weights are at most polynomially large in n, or (equivalently) if they are represented in unary notation.

Theorem 5 *If all weights are given in unary, the problems* Construct-LeastCore*,* In-LeastCore*,* Construct-EpsilonCore*,* In-EpsilonCore*, and* EpsilonCore *are in P.*

Proof Consider the following linear program:

$$\begin{aligned} \max \quad & C; \\ p_1 + \cdots + p_n &= 1 \\ p_i &\geq 0 \text{ for all } i = 1, \ldots, n \\ \sum_{i \in J} p_i \geq C \text{ for all } J \subseteq I \text{ such that } & \sum_{i \in J} w_i \geq T \end{aligned} \tag{1}$$

This linear program attempts to maximise the minimum excess by computing the greatest lower bound C on the payment to each winning coalition (i.e., a coalition whose total weight is at least T). Any solution to this linear program is a vector

of the form $(p_1, \ldots, p_n, C)$; clearly, the imputation $(p_1, \ldots, p_n)$ is in the least core, which has value $1 - C$.

Unfortunately, the size of this linear program may be exponential in n, as there is a constraint for each winning coalition. Nevertheless, we will now show how to solve it in time polynomial in n and $\sum_{i \in I} w_i$, by constructing a *separation oracle* for it. A separation oracle for a linear program is an algorithm that, given a putative feasible solution, checks whether it is indeed feasible, and if not, outputs a violated constraint [14]. It is known that a linear program can be solved in polynomial time as long as it has a polynomial-time separation oracle [14, Thm. 6.4.9, p. 179]. In our case, this means that we need an algorithm that given a vector $(p_1, \ldots, p_n, C)$, checks if there is a winning coalition J such that $\sum_{i \in J} p_i < C$.

To construct the separation oracle, we use dynamic programming to determine $P_0 = \min p(J)$ over all winning coalitions J. If $P_0 < C$, then the constraint that corresponds to $\operatorname{argmin}_{w(J) \geq T} p(J)$ is violated. Let $W = \sum_{i \in I} w_i$. For $k = 1, \ldots, n$ and $w = 1, \ldots, W$, let $x_{k,w} = \min\{p(J) \mid J \subseteq \{1, \ldots, k\}, w(J) = w\}$. Clearly, we have $P_0 = \min_{w=T,\ldots,W} x_{n,w}$. It remains to show how to compute $x_{k,w}$. For $k = 1$, we have $x_{1,w} = p_1$ if $w = w_1$ and $x_{1,w} = +\infty$ otherwise. Now, suppose we have computed $x_{k,w}$ for all $w = 1, \ldots, W$. Then we can compute $x_{k+1,w}$ as $\min\{x_{k,w}, p_{k+1} + x_{k,w-w_k}\}$. The running time of this algorithm is polynomial in n and W, i.e., in the size of the input.

Now, consider the application of the linear program for a weighted voting game $G = (I; \mathbf{w}; T)$. The constructed imputation $\mathbf{p}$ is a solution for Construct-LeastCore with instance G. Also, the solution to EpsilonCore with instance (G, ϵ) should be "yes" iff $\epsilon \geq 1 - C$. The solution to In-LeastCore with instance $(G, \mathbf{p}')$ should be "yes" if and only if every winning coalition $S \subseteq I$ has $\mathbf{p}'(S) \geq C$. This can be checked in polynomial time using the separation oracle from the proof of Theorem 5.

We will now show that we can use the same approach to solve Construct-EpsilonCore and In-EpsilonCore. Indeed, to solve In-EpsilonCore, we can simply run the dynamic programming algorithm described above for the given input vector $\mathbf{p}$ and check if $P_0 \geq 1 - \epsilon$. For Construct-EpsilonCore we modify the linear program by removing the objective function (so that the linear program becomes a *linear feasibility program*) and replacing C with $1 - \epsilon$ in all constraints involving C. Clearly, any solution to this program is an imputation in the ϵ-core of the game, and the program has no solutions if and only if the ϵ-core of the game is empty. □

The linear program constructed in the proof of Theorem 5 has coefficients in $\{0, 1\}$. It follows that this linear program has solutions that are rational numbers with polynomial bit complexity. (Recall that the bit complexity of an integer number r is given by $1 + \lceil \log_2(|r| + 1) \rceil$, and the bit complexity of a rational number p/q, where p and q are integers, is the sum of the bit complexities of p and q.)

Corollary 2 *Suppose that ϵ is the value of the least core of a game G. Then ϵ is a rational number whose bit complexity is at most $12n^2(n + 1)$. Moreover, there is an imputation $\mathbf{p} = (p_1, \ldots, p_n)$ in the least core of G such that each p_i has bit complexity at most $12n^2(n + 1)$.*

Proof Corollary 10.2a in [24, p. 122] states that a linear program of the form

$$\max_{\mathbf{x} \in \mathbb{R}^n} \{(\mathbf{c}, \mathbf{x}) \mid A\mathbf{x} \leq \mathbf{b}\},$$

where A is an $m \times n$ integer matrix, $\mathbf{b}$ is an integer vector of length m, $\mathbf{c}$ is an integer vector of length n, and $(\mathbf{c}, \mathbf{x})$ denotes the inner product of $\mathbf{c}$ and $\mathbf{x}$, has an optimal solution given by a vector with rational coordinates that have bit complexity at most $4n^2(n+1)(\sigma+1)$, where σ is the maximum bit complexity of the entries in A, $\mathbf{b}$ and $\mathbf{c}$. Observe that this bound does not depend on m (which in our case is exponential in n). For the linear program that characterises the least core, all entries of the corresponding matrix and vectors are 1, 0, or -1 (in particular, we have $\mathbf{c} = (0, \ldots, 0, 1)$), so we have $\sigma = 2$. Hence, the bit complexity of the vector $(p_1, \ldots, p_n, C)$ is at most $12n^2(n+1)$. As $\epsilon = 1 - C$, the result follows. □

3.2 Approximation scheme for the least core

In this section, we show that the pseudopolynomial algorithm of the previous section can be converted into an approximation scheme. More precisely, we construct an algorithm that, given a game $G = (I; \mathbf{w}; T)$ and a $\delta > 0$, outputs ϵ' such that if ϵ is the value of the least core of G then $\epsilon \leq \epsilon' \leq \epsilon + 2\delta$. The running time of our algorithm is polynomial in the size of the input as well as $1/\delta$, i.e., it is a fully polynomial additive approximation scheme. Subsequently, we show that it can be modified into a fully polynomial multiplicative approximation scheme (FPTAS), i.e., an algorithm that outputs ϵ' satisfying $\epsilon \leq \epsilon' \leq (1+\delta)\epsilon$ [1, p. 111].

Consider the linear program (1) in which C is some fixed integer multiple of δ and the goal is to find a feasible solution for this value of C or report than none exists. It is known that problems of this type can be solved in polynomial time as long as they have a polynomial-time separation oracle. We will describe a subroutine $\mathcal{A}$ that, given C, either outputs a feasible solution $\mathbf{p}$ for $C - \delta$ or correctly solves the problem for C. Our algorithm runs $\mathcal{A}$ for $C = 0, \delta, 2\delta, \ldots, 1$ and outputs $\epsilon' = 1 - C'$, where C' is the maximum value of C for which $\mathcal{A}$ finds a feasible solution.

Clearly, we have $\epsilon \leq 1 - C'$. Now, let $C^* = 1 - \epsilon$ be the optimal solution to the original linear program and let $k^* = \max\{k \mid k\delta \leq C^*\}$. As $k^*\delta \leq C^*$, there is a feasible solution for $k^*\delta$. When $\mathcal{A}$ is given $k^*\delta$, it either solves the linear program correctly, i.e., finds a feasible solution for $k^*\delta$, or finds a feasible solution for $k^*\delta - \delta$. In any case, we have $C' \geq (k^* - 1)\delta \geq C^* - 2\delta$, i.e., $1 - C' \leq \epsilon + 2\delta$.

It remains to describe the subroutine $\mathcal{A}$. Given a $C = k\delta$, it attempts to solve the linear program using the ellipsoid method. However, whenever the ellipsoid method calls the separation oracle for some payoff vector $\mathbf{p}$, we simulate it as follows. We set $\delta' = \delta/n$ and round down $\mathbf{p}$ to the nearest multiple of δ', i.e., set $p'_i = \max\{j\delta' \mid j\delta' \leq p_i\}$. We have $0 \leq p_i - p'_i \leq \delta'$. Let $x_{i,j} = \max\{w(J) \mid J \subseteq \{1, \ldots, j\}, \mathbf{p}'(J) = i\delta'\}$. The values $x_{i,j}$ are easy to compute by dynamic programming. Consider $U = \max\{x_{i,n} \mid i = 1, \ldots, (k-1)n - 1\}$. This is the maximum weight of a coalition whose total payoff under $\mathbf{p}'$ is at most $C - \delta - \delta'$. Since all entries of the modified payoff vector $\mathbf{p}'$ are multiples of δ', this is the maximum weight of a coalition whose total payoff under $\mathbf{p}'$ is less than $C - \delta$. If $U < T$, the payoff to each winning coalition under $\mathbf{p}'$ is at least $C - \delta$; as $p_i \geq p'_i$ for all $i \in I$, the same is true for $\mathbf{p}$. Hence, $\mathbf{p}$ is a feasible solution for $C - \delta$, so $\mathcal{A}$ outputs $\mathbf{p}$ and stops.

If $U \geq T$, there exists a winning coalition J such that $\mathbf{p}'(J) < C - \delta$. Since $|p'_i - p_i| \leq \delta' = \delta/n$ for all $i \in I$, this implies $p(J) < C$. Moreover, this J can be found using standard dynamic programming techniques. This means that that we have found a

violated constraint, i.e., successfully simulated the separation oracle and can continue with the ellipsoid method.

To convert this algorithm into an FPTAS, we need the following lemma.

Lemma 3 *If the core of a weighted voting game $G = (I; \mathbf{w}; T)$ is empty, i.e., the value of the least core of G is greater than 0, then this value is at least $\frac{1}{n}$.*

Proof The proof is similar to that of Theorem 2. Consider any imputation $\mathbf{p}$ in the least core of G. There exists an agent i such that $p_i \geq \frac{1}{n}$. As the core of G is empty, it cannot be the case that i is present in all winning coalitions, i.e., there exists a coalition J such that $i \notin J$, $w(J) \geq T$. On the other hand, we have $p(J) \leq 1 - \frac{1}{n}$, so the value of the least core of G is at least $\frac{1}{n}$. □

We are now ready to prove the main result of this section.

Theorem 6 *There exists a fully polynomial time approximation scheme for the value of the least core of a weighted voting game, i.e., an algorithm that, given a game $G = (I; \mathbf{w}; T)$ and a parameter δ, outputs an ϵ' that satisfies $\epsilon \leq \epsilon' \leq (1 + \delta)\epsilon$, where ϵ is the value of the least core of G, and runs in time* $\mathrm{poly}(n, \log w(I), 1/\delta)$.

Proof Set $\delta' = 2\delta/n$ and run the algorithm described above on G, δ'. Clearly, the running time of this algorithm is polynomial in n, $\log w(I)$, and $1/\delta$. Moreover, it outputs ϵ' that satisfies $\epsilon \leq \epsilon' \leq \epsilon + 2\delta'$. We have $\epsilon + 2\delta' = \epsilon + \delta/n \leq \epsilon(1 + \delta)$, and therefore $\epsilon' \leq \epsilon(1 + \delta)$. □

4 The nucleolus

Consider the following computational problems:

Name: Nucleolus.
Instance: Weighted voting game $(I; \mathbf{w}; T)$, agent $i \in I$.
Output: The nucleolus payoff of agent i in $(I; \mathbf{w}; T)$.

Name: IsZero-Nucleolus.
Instance: Weighted voting game $(I; \mathbf{w}; T)$, agent $i \in I$.
Question: Is the nucleolus payoff of agent i in $(I; \mathbf{w}; T)$ equal to 0?

We will show that IsZero-Nucleolus is coNP-hard. Clearly, this implies that the function problem Nucleolus is NP-hard. We start with the following lemma.

Lemma 4 *For weighted voting games, if an agent i is a dummy, his nucleolus payoff is 0.*

Proof Suppose that i is a dummy, i.e., $v(S) = v(S \cup \{i\})$ for all $S \subseteq I$, but $x_i \neq 0$, and consider the excess vector for $\mathbf{x}$. Let $e(\mathbf{x}, S)$ be the first element of this vector; clearly, $v(S) = 1$. It is easy to see that $i \notin S$: otherwise, we would have $v(S \setminus \{i\}) = 1$ and moreover, $x(S \setminus \{i\}) = x(S) - x_i < x(S)$. Now, consider an imputation $\mathbf{q}$ given by $q_i = \frac{x_i}{2}$, $q_j = x_j + \frac{x_i}{2(n-1)}$ for $j \neq i$. For any non-empty coalition T such that $i \notin T$ we have

$q(T) > x(T)$. Moreover, as $q_i \neq 0$, using the same argument as for $\mathbf{x}$, we conclude that the first element of the excess vector $e(\mathbf{q}, T)$ satisfies $i \notin T$. Hence,

$$e(\mathbf{q}, T) = q(T) - v(T) > x(T) - v(T) = e(\mathbf{x}, T) \geq e(\mathbf{x}, S),$$

a contradiction with the definition of the nucleolus. □

Remark 1 The converse of Lemma 4 is not true. Consider the coalitional game with $I = \{1, 2, 3\}$, $\mathbf{w} = \{\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\}$ and $T = \frac{3}{4}$. Winning coalitions are those that contain agent 1 and at least one of agents 2 and 3. Players 3 is not a dummy because for $S = \{1, 3\}$ we have $v(S) = 1$, $v(S \setminus \{3\}) = 0$. However, since 1 is a veto player, by Theorem 2, the nucleolus payoff of player 3 is 0.

Theorem 7 *The problem* IsZero-Nucleolus *is* coNP*-hard.*

Proof As in the proof of Theorem 3, we construct a weighted voting game based on an instance of Partition. Given an instance $A = (a_1, \ldots, a_n; K)$ of Partition, let $G = (I; \mathbf{w}; T)$ be the weighted voting game with $I = \{1, \ldots, n, n+1\}$, $T = K + 1$, $w_{n+1} = 1$, and $w_i = a_i$ for $i = 1, \ldots, n$. We will show that $x_{n+1} \neq 0$ if and only if A is a "yes"-instance of Partition.

Suppose first that A is a "no"-instance of Partition. Consider any winning coalition $S \subseteq I$ such that $n + 1 \in S$. We have $w(S) \geq K + 1$. Moreover, if $w(S) = K + 1$, then $w(S \setminus \{n + 1\}) = K$, implying that there is a partition. Hence, $w(S) > K + 1$, or, equivalently, $v(S \setminus \{n + 1\}) = 1$. We conclude that the $(n + 1)$st agent is a dummy. By Lemma 4, this implies $x_{n+1} = 0$.

Now, suppose that A is a "yes"-instance of Partition. Let $I' = I - \{n + 1\}$ and let J be a partition of I' satisfying $w(J) = w(I' \setminus J) = K$. Consider an imputation $\mathbf{p}$ with $p_{n+1} = 0$. The sets $S_1 = J \cup \{n + 1\}$ and $S_2 = (I' \setminus J) \cup \{n + 1\}$ satisfy $v(S_1) = v(S_2) = 1$. As $p_{n+1} = 0$, we have $p(S_1) + p(S_2) = p(J) + p(I' \setminus J) = 1$, so $\min\{e(\mathbf{p}, S_1), (\mathbf{p}, S_2)\} \leq -\frac{1}{2}$. That is, for any imputation with $p_{n+1} = 0$ the minimum excess is at most $-\frac{1}{2}$. On the other hand, under the imputation $q_i = \frac{w_i}{2K+1}$ the payoff of each winning coalition is at least $\frac{K+1}{2K+1} > \frac{1}{2}$, i.e., for this imputation the minimum excess is strictly greater than $-\frac{1}{2}$. As we have $\min_{S \subseteq I} e(\mathbf{x}, S) \geq \min_{S \subseteq I} e(\mathbf{q}, S)$, we conclude that $x_{n+1} \neq 0$. □

Remark 2 Theorem 7 implies that the problem Nucleolus cannot be approximated within any constant factor unless P=NP. More formally, it is not in the complexity class APX [1, p. 91] unless P=NP.

Remark 3 We can use the construction in the proof of Theorem 3 to show that Nucleolus is NP-hard; however, it does not imply the coNP-hardness of IsZero-Nucleolus. Conversely, the proof of Theorem 7 does not immediately imply that the least core-related problems are computationally hard. Therefore, to prove that all of our problems are computationally hard, we need both constructions.

Remark 4 While we have proved that IsZero-Nucleolus is coNP-hard, it is not clear if it is in coNP. Indeed, to verify that the nucleolus payoff of an agent i is 0, we would have to prove that there is an imputation $\mathbf{x}$ (the nucleolus) with $x_i = 0$, and that any imputation $\mathbf{p}$ with $p_i > 0$ produces an excess vector that is lexicographically smaller than that of $\mathbf{x}$. The latter condition involves exponentially-long vectors.

It is natural to ask if the pseudopolynomial time algorithm for the least core-related problems that was presented in the previous section can be adapted to compute the nucleolus. Paper [12] shows that the answer to this question is "yes", but the generalisation is far from trivial.

4.1 Approximating the nucleolus

Without loss of generality, we can assume that the sum of the weights in a weighted voting game is 1. We will refer to such a game as a *normalised* weighted voting game. Note that any weighted voting game is equivalent to some normalised game.

For many normalised weighted voting games considered in the literature, the vector $\mathbf{w}$ coincides with the nucleolus. For example, consider the set $\mathcal{C}$ of *constant-sum* games. A normalised weighted voting game $G = (I; \mathbf{w}; T)$ is in $\mathcal{C}$ if, for all $S \subseteq I$, $v(S) + v(I \setminus S) = 1$. Peleg [20] shows the following. Suppose $G = (I; \mathbf{w}; T) \in \mathcal{C}$. Let x be the nucleolus for G and let $G' = (I; \mathbf{x}; T)$. Then the nucleolus of G' is also equal to $\mathbf{x}$. A similar result for the set $\mathcal{C}'$ of *symmetric* games is shown in [30].

A normalised weighted voting game $G = (I; \mathbf{w}; T)$ is in $\mathcal{C}'$ if $T = \frac{1}{2}$ and there is a coalition S with $v(S) = v(I \setminus S)$.

It is not true in general that the vector $\mathbf{w}$ coincides with the nucleolus. It is also not true that w_i is a good approximation to the nucleolus payoff x_i. For example, in the game considered in Remark 1 the nucleolus payment x_3 is 0 but $w_3 = \frac{1}{4}$ (so these are not related by a constant factor). The nucleolus payment x_i can also exceed w_i by an arbitrary factor. For example, take an arbitrarily small $\delta > 0$. Consider the game with $I = \{1, 2\}$, $\mathbf{w} = \{1 - \delta, \delta\}$, and $T = 1 - \delta/2$. By Theorem 2, $x = (0.5, 0.5)$ so $x_2 = 0.5$. In any case, it is clear from Corollary 2 that w_i cannot be a constant-factor approximation to the nucleolus payment x_i of an individual agent i, since that would imply P=NP.

However, if we focus on approximating the payoffs of coalitions rather than individual players, we can obtain positive results for a large class of weighted voting games. Namely, in Theorem 8 we show that, for an appropriate sense of approximation based on *coalitions*, the vector $\mathbf{w}$ can be seen as a good approximation to the nucleolus. While this notion of approximation is quite different from the traditional one, we believe that it may be appropriate in the context of coalitional games: in such games, decisions to deviate are made by coalitions rather than individual players, and it is assumed that the players within a coalition can redistribute the payoffs. Hence, it makes sense to compare the behavior of two payoff vectors from the perspective of coalitions.

We start with a simple lower bound on nucleolus payments.

Lemma 5 *Let $G = (I; \mathbf{w}; T)$ be a normalised weighted voting game. If $w(S) \geq T$ then $x(S) \geq T$.*

Proof A winning coalition S has $e(\mathbf{w}, S) = \mathbf{w}(S) - 1 \geq T - 1$. The nucleolus maximises the minimum payoff to a winning coalition, so $e(\mathbf{x}, S) \geq T - 1$ and $\mathbf{x}(s) \geq T$. □

A *minimal* winning coalition is a coalition S with $\mathbf{w}(S) \geq T$ such that every proper subset $S' \subset S$ satisfies $\mathbf{w}(S') < T$. We will now use Lemma 5 to show that the weight of any minimal winning coalition is at most twice its nucleolus payoff.

Lemma 6 *Let $G = (I; \mathbf{w}; T)$ be a normalised weighted voting game. Suppose that every agent $i \in I$ has $w_i \leq T$. Let $S \subseteq I$ be a minimal winning coalition in G. Then $w(S) < 2x(S)$.*

Proof Let i be an agent in S. Since S is minimal, $w(S \setminus \{i\}) < T$. So $w(S) < T + w_i < 2T$. The result now follows from Lemma 5. □

We do not know whether there is a value α such every minimal winning coalition S of a normalised weighted voting game satisfies $x(S) \leq \alpha w(S)$. However, it is easy to see that this is true with $\alpha = 2$ if $T \geq \frac{1}{2}$ since $x(S) \leq 1$ and, for a winning coalition S, $w(S) \geq T \geq \frac{1}{2}$. So we get the following observation.

Observation 1 Let $G = (I; \mathbf{w}; T)$ be a normalised weighted voting game with $T \geq \frac{1}{2}$. Let $S \subseteq I$ be a winning coalition in G. Then $x(S) \leq 2w(S)$.

If T is less than $\frac{1}{2}$ but is relatively large compared to the individual weights, the vector $\mathbf{w}$ is still a good approximation to the nucleolus.

Lemma 7 *Consider a normalised weighted voting game $G = (I; \mathbf{w}; T)$ that satisfies $w_i \leq \epsilon T$, $T \geq \frac{\epsilon}{1+\epsilon}$ for some $\epsilon \leq 1$. For any such game, any minimal winning coalition $S \subseteq I$ satisfies $x(S) \leq 3w(S)$*

Proof For any minimal winning coalition S, we have $w(S \setminus \{i\}) < T$ for all $i \in S$, so $w(S) < T + w_i < T(1 + \epsilon)$. Now, fix a minimal winning coalition S_0. We have $w(S_0) \geq T$, $w(I \setminus S_0) > 1 - T(1 + \epsilon)$. We can construct a collection of $t = \lfloor \frac{1-T(1+\epsilon)}{T(1+\epsilon)} \rfloor \geq \frac{1}{T(1+\epsilon)} - 2$ disjoint minimal winning coalitions in $I \setminus S_0$. (For example, we can construct these coalitions consecutively by adding agents to a current coalition one by one until the weight of the coalition under construction becomes at least T.) Let these coalitions be $S_1, \ldots, S_t$. Lemma 5 implies $x(S_i) \geq T$ for $i = 1, \ldots, t$. Hence, $x(S_0) \leq 1 - tT \leq 2T - \frac{1}{1+\epsilon} + 1 \leq 2T + \frac{\epsilon}{1+\epsilon} \leq 3T \leq 3w(S_0)$. □

Remark 5 Let $G = (I; \mathbf{w}; T)$ be a normalised weighted voting game which satisfies $w_i \leq T^2$ for every agent $i \in I$. Then Lemma 7 applies with $\epsilon = T$.

Remark 6 By setting $\epsilon = 1$ in Lemma 7, we can obtain that $x(S) \leq 3w(S)$ for $T \geq \frac{1}{2}$ with the additional restriction that $w_i \leq T$ for all w_i; considering the case $T \geq \frac{1}{2}$ separately using Observation 1 gives us a stronger result.

Lemma 6, Observation 1 and Lemma 7 give us the following theorem. The theorem shows that, for a wide class of normalised weighted voting games, the weight vector $\mathbf{w}$ approximates the nucleolus $\mathbf{x}$ in the sense that the payoff to a minimal winning coalition only differs by at most a factor of 3 under these two imputations.

Theorem 8 *Let $G = (I; \mathbf{w}; T)$ be a normalised weighted voting game. Suppose that every agent $i \in I$ has $w_i \leq T$. If $T \geq \frac{1}{2}$ then any minimal winning coalition S satisfies $w(S)/2 \leq x(S) \leq 2w(S)$. If there is an $\epsilon \in (0, 1]$ such that $T \geq \frac{\epsilon}{1+\epsilon}$ and every agent satisfies $w_i \leq \epsilon T$ then any minimal winning coalition S satisfies $w(S)/2 \leq x(S) \leq 3w(S)$.*

5 Generalisations to *k*-vector weighted voting games

While weighted voting game represent a wide class of decision-making scenarios, there are also natural classes of problems that cannot be represented in this manner. Consider, for example, a political decision-making procedure in which, to pass a bill, the total weight of a coalition must exceed a certain threshold, and, in addition, the size of the coalition must exceed a certain threshold as well. Similarly, in task execution scenarios, there may be several types of resources needed to achieve a task, and each agent may possess a certain amount of each resource. Such coalitional games can be seen as intersections of weighted voting games, in the sense that a coalition is winning if it is a winning coalition in each of the component games. Formally, such scenarios are modeled as *k-vector weighted voting games*, defined as follows [28, 29].

Definition 2 A *k-vector weighted voting game* is given by a set of players I, $|I| = n$, a list of k n-dimensional vectors $(\mathbf{w}^1, \ldots, \mathbf{w}^k)$, and a list of k thresholds $\mathbf{T} = (T^1, \ldots, T^k)$; we write $G = (I; \mathbf{w}^1, \ldots, \mathbf{w}^k; \mathbf{T})$. Given a k-vector weighted voting game $G = (I; \mathbf{w}^1, \ldots, \mathbf{w}^k; \mathbf{T})$, we say that a coalition $J \subseteq I$ is *winning* if we have $\mathbf{w}^j(J) \geq T^j$ for all $j = 1, \ldots, k$, and losing otherwise. The parameter k is usually referred to as the *dimension* of the game.

It is well-known [28] that not all k-vector weighted voting games can be represented as weighted voting games. Consider, for example, a 2-dimensional game G with four players $\{1, 2, 3, 4\}$, weight vectors $\mathbf{w}^1 = (9, 1, 5, 5)$ and $\mathbf{w}^2 = (1, 9, 5, 5)$ and thresholds $T^1 = T^2 = 10$. In this game, $\{1, 2\}$ and $\{3, 4\}$ are winning coalitions. However, if we swap the players 2 and 3, i.e., consider the coalitions $\{1, 3\}$ and $\{2, 4\}$, we observe that both of these coalitions are losing: indeed, $w_1^2 + w_3^2 < 10$, $w_2^1 + w_4^1 < 10$. On the other hand, in [28] it is shown that this situation cannot occur in a weighted voting game. More precisely, given a weighted voting game $G' = (I'; \mathbf{w}'; T')$ and two disjoint winning coalitions C_1 and C_2, for all $x \in C_1$, $y \in C_2$ with weights w'_x, w'_y, at least one of the coalitions $C_1 \setminus \{x\} \cup \{y\}$ and $C_2 \setminus \{y\} \cup \{x\}$ must be winning: if $w'_x \geq w'_y$, then $w'(C_2 \setminus \{y\} \cup \{x\}) \geq w'(C_2) \geq T'$, and if $w'_x < w'_y$, then $w'(C_1 \setminus \{x\} \cup \{y\}) \geq w'(C_1) \geq T'$. We conclude that G cannot be represented as a weighted voting game.

Several complexity questions related to k-vector weighted voting games are studied in [11]; however, paper [11] does not consider stability issues in such games.

In this section, we extend some of the results presented in the previous sections to k-vector weighted voting games. First, observe that Theorem 2 applies to all simple games, and, in particular to k-vector weighted voting games. Hence, we get the following generalisation of Corollary 1.

Corollary 3 *There exists a polynomial-time algorithm that checks whether the core of a k-vector weighted voting game is non-empty. Moreover, if the core is non-empty, the nucleolus has polynomially-bounded rational coordinates and can be computed in polynomial time.*

For any positive integer k, we can extend all of our computational problems by allowing the input to be a k-vector weighted voting game (rather than a weighted voting game). We refer to the extended problem by the same name as the original problem, but with the prefix "k-". For example, we define k-EmptyCore as follows.

Name: k-EmptyCore.
Instance: k-vector weighted voting game $G = (I; \mathbf{w}^1, \ldots, \mathbf{w}^k; \mathbf{T})$.
Question: Is the core empty?

Any weighted voting game can be trivially represented as a k-vector weighted voting game by introducing dummy weight vectors $\mathbf{w}^2 = \cdots = \mathbf{w}^k = (0, \ldots, 0)$ and dummy thresholds $T^2 = \cdots = T^k = 0$. Hence, we get the following generalisation of Theorem 3.

Theorem 9 *The decision problems k-EpsilonCore and k-In-EpsilonCore are* coNP*-hard. The decision problem k-In-LeastCore is* NP*-hard. The function problems k-Construct-EpsilonCore and k-Construct-LeastCore are* NP*-hard.*

We also get the following generalisation of Theorem 7.

Theorem 10 *The problem k-IsZero-Nucleolus is* coNP*-hard.*

As for our algorithmic results, the situation is more complicated. We can modify the pseudopolynomial time algorithm presented in the proof of Theorem 5 to work for k-vector weighted voting games. Thus, we have the following result.

Theorem 11 *Given a k-vector weighted voting game $G = (I; \mathbf{w}^1, \ldots, \mathbf{w}^k; \mathbf{T})$, the problems k-EpsilonCore, k-Construct-EpsilonCore, k-In-EpsilonCore, k-Construct-LeastCore, and k-In-LeastCore can be solved in time* poly(n, W^k)*, where*

$$W = \max_{j=1,\ldots,k} \sum_{i=1}^{n} w_i^j$$

In particular, if the weights w_i^j are presented in unary and k is fixed, then these problems can be solved in polynomial time.

Proof The proof is similar to that of Theorem 5. Namely, we consider the linear program

$$\begin{aligned} \max \quad & C; \\ p_1 + \cdots + p_n &= 1 \\ p_i &\geq 0 \text{ for all } i = 1, \ldots, n \\ \sum_{i \in J} p_i &\geq C \text{ for all } J \subseteq I \text{ such that } J \text{ is a winning coalition in } G \end{aligned} \tag{2}$$

and show that it can be solved in polynomial time by constructing a polynomial-time separation oracle for it.

As in the proof of Theorem 5, the separation oracle is an algorithm that takes a vector $(p_1, \ldots, p_n, C)$ as an input and checks if there is a winning coalition J such that $\sum_{i \in J} p_i < C$. Our algorithm is based on dynamic programming. For $j = 1, \ldots, n$ and any integer vector $\mathbf{v} = (v^1, \ldots, v^k) \in [0, W]^k$, let

$$x(j, \mathbf{v}) = \min \left\{ p(J) \mid J \subseteq \{1, \ldots, j\}, w^1(J) = v^1, \ldots, w^k(J) = v^k \right\}.$$

It is straightforward to compute $x(1, \mathbf{v})$: we have $x(1, \mathbf{v}) = p_1$ if $w_1^1 = v^1, w_1^2 = v^2, \ldots, w_1^k = v^k$ and $x(1, \mathbf{v}) = +\infty$ otherwise. Now, suppose we have computed $x(j, \mathbf{v})$ for all $\mathbf{v} \in [0, W]^k$. Then we can compute $x(j+1, \mathbf{v})$ as $\min\{x(j, \mathbf{v}), p_{j+1} + x(j, \mathbf{v}')\}$, where $\mathbf{v}'$ is given by $(v^1 - w_{j+1}^1, \ldots, v^k - w_{j+1}^k)$. Now, suppose that we have computed $x(n, \mathbf{v})$ for all $\mathbf{v} \in [0, W]^k$. We can now check all vectors $\mathbf{v}$ that correspond to winning coalitions of G, i.e., the ones that satisfy $v^j \geq T^j$ for $j = 1, \ldots, k$. If any of the corresponding values of $x(n, \mathbf{v})$ is less than C, then there is a violated constraint. The running time of this algorithm is polynomial in n and W^k, hence we have successfully constructed an efficient separation oracle for our linear program.

As in the proof of Theorem 5, a solution to this linear program can be converted to solutions to k-EpsilonCore, k-In-LeastCore, k-Construct-LeastCore, k-In-LeastCore, and k-Construct-LeastCore. □

On the other hand, we do not know how to adapt our approximation algorithm to run in polynomial time for k-vector weighted voting games. Intuitively, the reason for this is that in this setting we cannot use dynamic programming to find a winning coalition that is paid less than a certain amount. We propose finding an approximation algorithm for the least core in k-vector weighted voting games as an open problem.

6 Related work

Cooperative game theory was introduced into to the artificial intelligence community largely through the work of Shehory and Kraus [27], who emphasised the *coalition formation* problem from a computational point of view. They developed algorithms for coalition structure formation in which agents were modelled as having different capabilities, and were assumed to benevolently desire some overall task to

be accomplished, where this task had some complex (plan-like) structure [25–27]. Sandholm and colleagues developed algorithms to find optimal coalition structures (i.e., partitions of agents) with worst case guarantees (that is, ones whose performance is within some given ratio bound k of optimal) [22].

More recently, the general issue of finding representations for coalitional games that strike a useful balance between tractability and expressive power has received some attention. Conitzer and Sandholm consider a *modular* representation of coalitional games, where a characteristic function is represented as a collection of subgames [5]; under this representation, they showed that checking non-emptiness of the core is coNP-complete. In related work, Ieong and Shoham propose a representation of coalitional games called *marginal contribution nets* [15]. In this representation, a characteristic function over a set I of agents is represented as a set of rules, with the structure

$$\text{pattern} \longrightarrow \text{value}.$$

The pattern is a propositional formula whose variables correspond to the elements of I, and such a rule is said to *apply* to a group of agents S if the truth assignment that sets the variables corresponding to the elements of S to $\top$ while setting all other variables to $\bot$ satisfies this formula. The value of a coalition in the marginal contribution network representation is then the sum over the values of all the rules that apply to the coalition. Ieong and Shoham show that, under this representation, checking whether an imputation is in the core is coNP-complete, while checking whether the core is non-empty is coNP-hard. They also show that their representation can capture that of Conitzer and Sandholm [5].

The first systematic investigation of the computational complexity of solution concepts in coalitional games was undertaken by Deng and Papadimitriou [8]. They used a representation based on weighted graphs. To represent a coalitional game with agents I, they used an undirected graph on I, with integer weights $w_{i,j}$ between nodes $i, j \in I$. The value $v(C)$ of a coalition $C \subseteq I$ was then defined to be $\sum_{\{i,j\} \subseteq I} w_{i,j}$, i.e., the value of a coalition $C \subseteq I$ is the weight of the subgraph induced by C. Given this representation, Deng and Papadimitriou showed that the problem of determining emptiness of the core was NP-complete, while the problem of checking whether a specific imputation was in the core of such a game was coNP-complete [8, p. 260]; they also showed that these problems could be solved in polynomial time for graphs with non-negative weights [8, p. 261], and gave the first complexity results for weighted voting games. Subsequently, computational complexity issues have been studied for many other compactly representable coalitional games. Bilbao and colleagues [3] surveyed a representative sample of papers in this area. They focussed on settings in which the characteristic function was given by various combinatorial structures. Given such representations, they showed that (1) establishing membership of the core ranges from polynomial-time computable to coNP complete; (2) determining whether the core is empty is in general NP-complete; (3) computing the Shapley value is in general #P-complete; (4) computing the nucleolus is in general NP-hard. However, there exist natural games in which some or all of these problems are polynomial-time solvable. For example, Deng et al. [7] show that the nucleolus of a network flow game can be computed in polynomial time.

With respect to computational aspects of weighted voting games, the focus of the existing research has been on fairness-related solution concepts. In particular, it has

Table 1 Complexity results for binary weights

Problem name	Lower bound	Upper bound	Reference
EmptyCore	P	P	Corollary 1
EpsilonCore	coNP	Σ_2^p	Theorems 3, 4
In-EpsilonCore	coNP	coNP	Theorem 4
In-LeastCore	NP	Π_2^p	Theorems 3, 4
IsZero-Nucleolus	coNP	?	Theorem 7

been shown that, for weighted voting games, computing the Shapley value of a given player is #P-complete [13], and that it is coNP-hard to determine whether this value is zero [17, 21]. In fact, even approximating the Shapley value within a constant factor is intractable unless P=NP—see Remark 2. On the other hand, there exist presudopolynomial algorithms for computing the Shapley value that are based on dynamic programming [13, 16]. Many of these results apply to the Banzhaf power index [2], another popular solution concept for this class of games.

7 Conclusions and future work

We have systematically investigated the complexity of solution concepts for weighted voting games, a compact class of simple coalitional games that are widely used in practice. We have shown that many stability-related solution concepts in weighted voting games are hard to compute when the players' weights are given in binary. The complexity results for the decision problems considered in the paper are summarised in Table 1. On the other hand, for the least core-related problems, we provided both pseudopolynomial algorithms (i.e., algorithms that run in time polynomial in the maximum weight—see Theorem 5) and a fully polynomial time approximation scheme (FPTAS—see Theorem 6). We also generalised many of our results to k-vector weighted voting games (see Section 5). One of the main contributions of the present paper is to consider the least core and the nucleolus: while other solution concepts such as the core and Shapley value have been thoroughly studied from a computational point of view, the least core and the nucleolus has attracted little attention in this respect.

Perhaps the most obvious and exciting directions for future work involve considering *combinations* of weighted voting games. Our first results in this direction are presented in Section 5, and relate to *conjunctions* of weighted voting games, in which a coalition wins overall if it wins in a collection of k distinct weighted voting games. Such multi-cameral decision making bodies are common in the political world: examples include the US federal legislature and the voting system of the European Union [29]. One can imagine other, richer voting systems, for example involving *Boolean combinations* of weighted voting games.

Acknowledgements This research was supported by the EPSRC under the "Market Based Control" project, by the ESRC EUROCORES LogiCCC project "Computational Foundations of Social Choice", and by NRF under grant NRF-RF2009-08 "Algorithmic Aspects of Coalitional Games". A preliminary version of this paper appeared in AAAI'07 [10].

References

1. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation. Springer, Berlin (1999)
2. Banzhaf, J.F.: Weighted voting doesn't work: a mathematical analysis. Rutgers Law Rev. **19**, 317–343 (1965)
3. Bilbao, J., Fernández, J., López, J.: Complexity in cooperative game theory (manuscript)
4. Bilbao, J.M., Fernández, J.R., Jiminéz, N., López, J.J.: Voting power in the European union enlargement. Eur. J. Oper. Res. **143**, 181–196 (2002)
5. Conitzer, V., Sandholm, T.: Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In: Proceedings of the Ninteenth National Conference on Artificial Intelligence (AAAI-2004), pp. 219–225, San Jose, CA (2004)
6. Conitzer, V., Sandholm, T.: Complexity of constructing solutions in the core based on synergies among coalitions. Artif. Intell. **170**, 607–619 (2006)
7. Deng, X., Fang, Q., Sun, X.: Finding nucleolus of flow game. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 124–131. ACM, New York (2006)
8. Deng, X., Papadimitriou, C.H.: On the complexity of cooperative solution concepts. Math. Oper. Res. **19**(2), 257–266 (1994)
9. Elkind, E., Chalkiadakis, G., Jennings, N.R.: Coalition structures in weighted voting games. In: Proc. 18th European Conference on Artificial Intelligence (ECAI'08) (2008)
10. Elkind, E., Goldberg, L.A., Goldberg, P.W., Wooldridge, M.: Computational complexity of weighted threshold games. In: Proc. 22nd Conference on Artificial Intelligence (AAAI'07) (2007)
11. Elkind, E., Goldberg, L.A., Goldberg, P.W., Wooldridge, M.: On the dimensionality of voting games. In: Proc. 23rd Conference on Artificial Intelligence (AAAI'08) (2008)
12. Elkind, E., Pasechnik, D.: Computing the nucleolus of weighted voting games. In: Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (SODA'09) (2009)
13. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W.H. Freeman, New York (1990)
14. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, Algorithms and Combinatorics, vol. 2, 2nd edn. Springer, Berlin (1993)
15. Ieong, S., Shoham, Y.: Marginal contribution nets: a compact representation scheme for coalitional games. In: Proceedings of the Sixth ACM Conference on Electronic Commerce (EC'05), Vancouver, Canada (2005)
16. Matsui, T., Matsui, Y.: A survey of algorithms for calculating power indices of weighted majority games. J. Oper. Res. Soc. Jpn **43**(1), 71–86 (2000)
17. Matsui, Y., Matsui, T.: NP-completeness for calculating power indices of weighted majority games. Theor. Comp. Sci. **263**(1–2), 305–310 (2001)
18. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT, Cambridge (1994)
19. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading (1994)
20. Peleg, B.: On weights of constant-sum majority games. SIAM J. Appl. Math. **16**, 527–532 (1968)
21. Prasad, K., Kelly, J.S.: NP-completeness of some problems concerning voting games. Int. J. Game Theory **19**(1), 1–9 (1990)
22. Sandholm, T., Larson, K., Andersson, M., Shehory, O., Tohmé, F.: Coalition structure generation with worst case guarantees. Artif. Intell. **111**(1–2), 209–238 (1999)
23. Schmeidler, D.: The nucleolus of a characteristic function game. SIAM J. Appl. Math. **17**, 1163–1170 (1969)
24. Schrijver, A.: Combinatorial Optimization: Polyhedra and Efficiency. Springer, New York (2003)
25. Shehory, O., Kraus, S.: Coalition formation among autonomous agents: strategies and complexity. In: Castelfranchim, C., Müller, J.-P. (eds.) From Reaction to Cognition—Fifth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW-93. LNAI, vol. 957, pp. 56–72. Springer, Berlin (1995)
26. Shehory, O., Kraus, S.: Task allocation via coalition formation among autonomous agents. In: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), pp. 655–661, Montréal, Québec, Canada (1995)
27. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artif. Intell. **101**(1–2), 165–200 (1998)

28. Taylor, A., Zwicker, W.: Weighted voting, multicameral representation, and power. Games Econom. Behav. **5**, 170–181 (1993)
29. Taylor, A.D., Zwicker, W.S.: Simple Games: Desirability Relations, Trading, Pseudoweightings. Princeton University Press, Princeton (1999)
30. Wolsey, L.A.: The nucleolus and kernel for simple games or special valid inequalities for $0-1$ linear integer programs. Int. J. Game Theory **5**(4), 227–238 (1976)