Int J Game Theory (2001) 30:79–98

International Journal of Game Theory
© Springer Verlag 2001

On the computation of the nucleolus of a cooperative game

Ulrich Faigle[1], **Walter Kern**[2], **Jeroen Kuipers**[3]

[1] Zentrum für Angewandte Informatik, Universität zu Köln, Weyertal 80, 50931 Köln, Germany. E-mail address: faigle@zpr.uni-koeln.de
[2] Department of Applied Mathematics, University of Twente, P.O.Box 217, 7500 AE Enschede, The Netherlands. E-mail address: kern@math.utwente.nl
[3] Department of Mathematics, Maastricht University, P.O.Box 616, 6200 MD Maastricht, The Netherlands. E-mail address: kuipers@math.unimaas.nl

Received February 2000/Final version April 2001

Abstract. We consider classes of cooperative games. We show that we can efficiently compute an allocation in the intersection of the prekernel and the least core of the game if we can efficiently compute the minimum excess for any given allocation. In the case where the prekernel of the game contains exactly one core vector, our algorithm computes the nucleolus of the game. This generalizes both a recent result by Kuipers on the computation of the nucleolus for convex games and a classical result by Megiddo on the nucleolus of standard tree games to classes of more general minimum cost spanning tree games. Our algorithm is based on the ellipsoid method and Maschler's scheme for approximating the prekernel.

1. Introduction

Recent years have seen an increased interest in computational complexity aspects of solution concepts in cooperative game theory (see, *e.g.*, Deng and Papadimitriou [1994] or Deng *et al.* [1997], [2000]). On the positive side, efficient algorithms have been developed, *e.g.*, for the computation of the nucleolus of assignment games (Solymosi and Raghavan [1994)], the nucleon of matching games (Faigle *et al.* [1998a]), the nucleolus of convex games (Kuipers [1996]), and the nucleolus of standard tree games (Megiddo [1987], Granot *et al.* [1996]). On the negative side, several *NP*-hardness results were obtained, *e.g.*, for testing core membership (Faigle *et al.* [1997]) or computing the nucleolus for min cost spanning tree games (MCST-games) (Faigle *et al.*

1991 *Mathematics Subject Classification*. 90C27, 90D12.
Key words and phrases. Core, nucleolus, prekernel, kernel, computational complexity, convex games, MCST-games, ellipsoid method.

[1998b], see also Faigle *et al.* [2000] for related results). It is a challenging problem in mathematical programming to decide what classes of cooperative games permit polynomial time computation of the nucleolus.

The present paper takes a new approach to the problem which is motivated by Schmeidler's [1969] observation that the (pre)nucleolus of a game lies in the (pre)kernel. In fact, we propose an algorithm for the computation of a prekernel element that will actually produce the nucleolus for many classes of games.

Our algorithm uses the ellipsoid method as a subroutine (which implies that the efficiency of our algorithm is of a more theoretical kind). Since the prekernel of a cooperative game is typically non-convex, it might be somewhat surprising that the ellipsoid method should be successful at all. We overcome this difficulty with the help of an approximating scheme that Maschler suggested for finding (pre-)kernel elements: we perform sufficiently many steps of Maschler's scheme in order to find suitable cutting planes for the ellipsoid algorithm.

The presentation of our paper is as follows. In Section 2, we review the basic game theoretic concepts. Our computational model is made precise in Section 3, where we also review the ellipsoid method to the extent we need it. Maschler's approximation scheme is discussed in Section 4. We describe our algorithm in Section 5. Some applications are given afterwards.

2. Basic definitions

For our purposes, a *cooperative game* is described by a pair (N, c), where N is a finite set of n players and $c : 2^N \to \mathbb{R}$ is a *cost function* satisfying $c(\emptyset) = 0$. A *coalition* is a subset $S \subseteq N$ and $c(S)$ is called the *cost* of coalition S with the interpretation that $c(S)$ is the joint cost of the players in S if they decide to cooperate.

A vector $x \in \mathbb{R}^N$ is an *allocation* (or *preimputation*) if $x(N) = c(N)$. (We will use the shorthand notation $x(S) = \sum_{i \in S} x_i$ throughout in this paper). The allocation x is called an *imputation* if $x_i \leq c(\{i\})$ holds for all $i \in N$.

Given an allocation $x \in \mathbb{R}^N$, the *excess* of a coalition S (with respect to x) is defined as the number

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

Setting $e_{min}(x) := \min\{e(S, x) \mid \emptyset \neq S \neq N\}$, we arrive at the *core* of the game (N, c) as the set core(c) of all allocations whose excesses are non-negative, *i.e.*,

$$\text{core}(c) := \{x \in \mathbb{R}^N \mid x(N) = c(N), e_{min}(x) \geq 0\}.$$

Because the core of a game may be empty, it is useful to relax the notion of the core. Let

$$\varepsilon(c) := \max\{e_{min}(x) \mid x \in \mathbb{R}^N, x(N) = c(N)\}$$

and define the *least core* as the set leastcore(c) of those allocations whose non-trivial excesses are at least $\varepsilon(c)$, *i.e.*,

$$\text{leastcore}(c) := \{x \in \mathbb{R}^N \mid x(N) = c(N), e_{min}(x) \geq \varepsilon(c)\}.$$

Note that leastcore(c) is always non-empty. Moreover, leastcore(c) $\subseteq$ core(c) holds whenever core(c) is non-empty.

Another optimality property with respect to the excesses of an allocation gives rise to the nucleolus of a cooperative game in the following way. Given an allocation $x \in \mathbb{R}^N$ for the game (N, c), let $\Theta(x)$ denote the $(2^{|N|} - 2)$-dimensional vector of all non-trivial excesses $e(S, x)$, $\emptyset \neq S \neq N$, arranged in non-decreasing order. The *prenucleolus* $\eta(c)$ is then defined to be the (unique) allocation $x \in \mathbb{R}^N$ that lexicographically maximizes Θ over the set of all allocations. We obtain the *nucleolus* when we compute the lexicographic maximum over the subset of all imputations.

It follows immediately from the definition that $\eta(c)$ always exists and is a member of the least core leastcore(c). Moreover, the prenucleolus and the nucleolus coincide whenever core(c) is non-empty.

The (pre)nucleolus can be computed by solving a sequence of linear programs as follows. We set $\mathscr{S}_0 = \{\emptyset, N\}$ and first solve

$$\begin{aligned} \max \quad & \varepsilon \\ & \sum_{i \in N} x_i = c(N) \\ & \sum_{i \in S} x_i \leq c(S) - \varepsilon \quad \text{for all } S \notin \mathscr{S}_0. \end{aligned} \tag{LP$_1$}$$

If ε_1 is the optimal value of (LP$_1$), let $\mathscr{S}_1$ be the collection of all coalitions that become tight at $\varepsilon = \varepsilon_1$ and solve

$$\begin{aligned} \max \quad & \varepsilon \\ & \sum_{i \in N} x_i = c(N) \\ & \sum_{i \in S} x_i = c(S) - \varepsilon_1 \quad \text{for all } S \in \mathscr{S}_1 \\ & \sum_{i \in S} x_i \leq c(S) - \varepsilon \quad \text{otherwise.} \end{aligned} \tag{LP$_2$}$$

Continuing this way, we obtain a sequence $\varepsilon_1 < \varepsilon_2 \cdots < \varepsilon_k$ until, finally, the optimal solution of (LP$_k$) is unique and hence the prenucleolus $\eta(c)$ of the game. It is not difficult to see that this procedure requires the solution of at most $|N|$ linear programs. Each of these LPs, however, has exponentially many restrictions. Rather than attacking these LPs, we approach the nucleolus indirectly.

We thus come to the central game theoretic solution concept for our current investigation. Given an allocation x and a pair (i, j) of players, we define the *surplus* of player $i \in N$ against player $j \in N$ as the number

$$s_{ij}(x) := \min\{e(S, x) \mid S \subseteq N, i \in S, j \notin S\}.$$

The *prekernel* of the game (N, c) is the set $\mathcal{K}^*(c)$ of allocations x for which the surplusses are symmetric, *i.e.*,

$$\mathcal{K}^*(c) := \{x \in \mathbb{R}^N \mid x(N) = c(N), s_{ij}(x) = s_{ji}(x) \ \forall i, j\}.$$

The *kernel* $\mathcal{K}(c)$ of the game (N, c) is a related solution concept and is defined as the set of all imputations $x \in \mathbb{R}^N$ satisfying the following condition for every pair (i, j) of players:

$$s_{ij}(x) < s_{ji}(x) \text{ implies } x_j = c(\{j\}).$$

It is generally agreed that the notion of (pre-)kernel is not very intuitive in its own right. Yet, it has received considerable attention because it exhibits quite interesting (and useful) relationships with other solution concepts (see also Section 4 below). For example, one can prove that the prenucleolus is always in the prekernel (*cf.* Schmeidler [1969]). On the other hand, we have seen that the (pre)nucleolus is in the core whenever the core is non-empty. Hence we conclude

Theorem 2.1. *If* core$(c) \cap \mathcal{K}^*(c)$ *contains just one single point* $x^* \in \mathbb{R}^N$*, then* x^* *is the nucleolus.* ◇

There are several important classes of games to which Theorem 2.1 applies: for example, the class of *convex games* (*cf.* Maschler *et al.* [1972]) and the class of *minimum cost spanning tree* (MCST) games (*cf.* Granot and Huberman [1984]). (In the case of convex games, even the prekernel consists of only one point; the prekernel of MCST-games may contain several points, but only one of them is in the core).

Therefore, it seems desirable to have efficient algorithms available that compute elements in $\mathcal{K}^*(c)$ or in $\mathcal{K}^*(c) \cap \text{core}(c)$. Our main result states that, provided we can efficiently compute all surplusses $s_{ij}(x)$ for any allocation x, we can also efficiently compute an element $x^* \in \mathcal{K}^*(c)$. Moreover, the allocation x^* we compute will be in core(c) whenever core$(c) \neq \emptyset$. To be more precise: the allocation x^* we compute will always be a member of the least core of the game (N, c).

Intuitively, it seems inevitable that one should at least be able to compute the $s_{ij}(x)$'s if one wants to check whether a given vector $x \in \mathbb{R}^N$ is a member of the prekernel $\mathcal{K}^*(c)$. In that sense, our result would be about the best one can hope for.

Before proceeding further, however, we want to make the notion of (algorithmically) "efficient" ("polynomial time") more precise and describe our computational model in more detail.

3. Computational assumptions

We consider a class $\mathcal{C}$ of cooperative games. Each game $(N, c) \in \mathcal{C}$ is assumed to have a rational cost function c and will be given to us by

(C_1) The finite set N of players.

(C_2) An upper bound $\langle c\rangle$ on the maximum (input) size (*i.e.*, encoding length)

$$\max_{S\subseteq N}\langle c(S)\rangle.$$

(C_3) An algorithm ("oracle"), which, on input $S \subseteq N$, computes and outputs the value $c(S) \in \mathbb{Q}$.

(As is usual, we take the encoding length $\langle q\rangle$ of a rational number $q \in \mathbb{Q}$ as the number of bits needed for its binary representation (see, *e.g.*, Grötschel *et al.* [1993] for more details)).

In the following, we consider *algorithms for the class* $\mathscr{C}$. The input for an algorithm A for $\mathscr{C}$ is a game $(N, c) \in \mathscr{C}$, presented *via* the player set N, the upper bound $\langle c\rangle$ on the encoding length of the c-values, and access to the oracle for the values $c(S)$, and possibly some "additional" input x of encoding length $\langle x\rangle$. The running time of A is measured in terms of the number of "elementary operations" (bit operations) and calls to the oracle for the c-values.

We say that the algorithm A is *polynomial time* or *efficient* if there exists a polynomial p_A such that, on input (N, c) and x, A performs at most $p_A(|N|, \langle c\rangle, \langle x\rangle)$ elementary operations and calls to the oracle.

Let, for example, $\mathscr{C}$ be the class of MCST-games (see also Section 7). Then the cost function c of a game $(N, c) \in \mathscr{C}$ is defined implicitly by a graph on $|N| + 1$ nodes and edge weights w_{ij}. The maximum size of the c-values will be bounded by a polynomial in $|N|$ and $\langle w\rangle := \max\langle w_{ij}\rangle$ (because each $c(S)$ is the sum of the edge weights in the underlying graph).

Thus a polynomial time algorithm for the class of MCST-games, in the sense above, is an algorithm A which, on input (N, c) and x, performs polynomially (in $|N|$, $\langle w\rangle$, and $\langle x\rangle$) many elementary operations and calls to the oracle for (N, c). Note the oracle may in the present example be replaced by any minimum spanning tree algorithm that computes the value $c(S)$ in polynomial time (relative to $|N|$ and $\langle w\rangle$). Hence a polynomial time algorithm for the class of MCST-games in this abstract setting is just a polynomial time algorithm in the usual sense (with running time bounded by $poly(|N|, \langle w\rangle, \langle x\rangle)$.

Let us now return to the extra assumption we need to guarantee that our computations will be polynomial. First of all, recall that without any further assumption the problem of computing an element in $\mathscr{K}^*$, even if this set consists of only one point, is NP-hard. This follows, *e.g.*, from Faigle *et al.* [1998b]. To see what assumption we need, let us consider the seemingly simpler problem of checking whether a given allocation x is an element of $\mathscr{K}^*(c) \cap \mathrm{core}(c)$. This amounts to verifying the following equations and inequalities:

$$x(S) \le c(S) \quad (S \subseteq N) \tag{2.1}$$

$$s_{ij}(x) = s_{ji}(x) \quad (i, j \in N) \tag{2.2}$$

As mentioned above, it seems reasonable that, in order to check (2.2) efficiently, we should at least be able to efficiently compute all $s_{ij}(x)$, given x. This is exactly the computational extra assumption relative to the class $\mathscr{C}$ we make:

Assumption (CCS):

There exists a polynomial algorithm A which, for every given game $(N,c) \in \mathscr{C}$, allocation $x \in \mathbb{R}^N$, and players $i, j \in N$, computes the number $s_{ij}(x)$.

Here we implicitly assume, of course, that the input vectors x in assumption (CCS) are rational and hence $\langle x \rangle$ is well-defined.

Note that (CCS) also allows us to check whether the system of inequalities (2.1) above is satisfied, since

$$\min_{\emptyset \neq S \neq N} e(S,x) = \min_{i,j} s_{ij}(x).$$

Indeed, the "converse" is also true, *i.e.*, (CCS) is equivalent to the following assumption on the computational complexity of the minimal excess:

Assumption (CCM):

There exists a polynomial algorithm A' which, for every given $(N,c) \in \mathscr{C}$ and allocation $x \in \mathbb{R}^N$, computes

$$e_{min}(x) = \min_{\emptyset \neq S \neq N} e(S,x).$$

To prove the equivalence, we have to show that (CCM) implies (CCS).

Thus assume A' is an algorithm as implied by (CCM). Let $x \in \mathbb{R}^N$ be an allocation and $i, j \in N$. Let $K := 2^{1+\langle c \rangle + \langle x \rangle}$ (so that $K \geq 2(|c(S)| + |x(S)|)$ for all $S \subseteq N$), and define the allocation $\bar{x}$ by

$$\bar{x}_m = \begin{cases} x_i + K & \text{if } m = i \\ x_j - K & \text{if } m = j \\ x_m & \text{else.} \end{cases}$$

Use A' to compute $e_{min}(\bar{x})$. If $S \subset N$ is a set whose excess attains this minimum, then, obviously, $i \in S$ and $j \notin S$. Hence $e_{min}(\bar{x}) = s_{ij}(\bar{x})$. But $s_{ij}(x) = s_{ij}(\bar{x}) + K$, which establishes the claim.

Still assuming (CCS) or (CCM), it is important to see that a similar construction allows us to actually compute in polynomial time coalitions $S \subset N$ with $e_{min}(x) = e(S,x)$ and $S_{ij} \subset N$ with $i \in S_{ij}$, $j \notin S_{ij}$, and $s_{ij}(x) = e(S_{ij}, x)$. We illustrate the algorithmic idea for $e_{min}(x)$.

For all $i, j \in N$, $i \neq j$, define the allocation $x^{(i,j)}$ by

$$x^{(i,j)}_m = \begin{cases} x_i + 1 & \text{if } m = i \\ x_j - 1 & \text{if } m = j \\ x_m & \text{else.} \end{cases}$$

Then we obtain for each $S \subseteq N$,

$$e(S, x^{(i,j)}) = \begin{cases} e(S,x) - 1 & \text{if } i \in S, j \notin S \\ e(S,x) + 1 & \text{if } j \in S, i \notin S \\ e(S,x) & \text{else.} \end{cases}$$

Let $i_1, j_1 \in N$ now be such that $e_{min}(x^{(i_1,j_1)}) = e_{min}(x) - 1$. Then each $\emptyset \neq S \neq N$ with $e_{min}(x^{(i_1,j_1)}) = e(S, x^{(i_1,j_1)})$ necessarily satisfies $i_1 \in S$, $j_1 \notin S$, and $e_{min}(x) = e(S,x)$. Assume, we have found such a pair (i_1, j_1) of players and set $x_1 := x^{(i_1,j_1)}$.

We try to find a pair (i_2, j_2) of players with $j_2 \in N \backslash \{j_1\}$ and $e_{min}(x_1^{(i_2,j_2)}) = e_{min}(x_1) - 1$. If this turns out to be impossible, then $N\backslash\{j_1\}$ apparently must satisfy $e_{min}(x_1) = e(N\backslash\{j_1\}, x_1)$ and hence

$$e_{min}(x) = e(N\backslash\{j_1\}, x).$$

Otherwise, each $\emptyset \neq S \neq N$ with $e_{min}(x_1^{(i_2,j_2)}) = e(S, x_1^{(i_2,j_2)})$ necessarily satisfies $S \subseteq N\backslash\{j_1, j_2\}$. So we can repeat the procedure with $x_2 := x_1^{(i_2,j_2)}$ in order to find a pair (i_3, j_3) with $j_3 \in N\backslash\{j_1, j_2\}$ *etc.*. It is clear that this procedure halts after $k < |N|$ iterations, and we will have found a nonempty coalition $S = N\backslash\{j_1, \ldots, j_k\}$ such that

$$e_{min}(x) = e(S, x).$$

We have seen that if we assume (CCS) – or, equivalently, (CCM) – we can efficiently check whether a given allocation x is in $\mathcal{K}^*(c) \cap \mathrm{core}(c)$. The task of actually computing such an element x in $\mathcal{K}^*(c) \cap \mathrm{core}(c)$, however, appears to be more difficult. Our approach to such a computation will be based on a scheme for approximating the prekernel, which was proposed by Maschler at the 1965 Jerusalem Conference on Game Theory, convergence of which was proved later by Stearns (*cf.* Stearns [1968]).

However, as Stearns points out, "there are some undesirable properties of this method regarding the speed of convergence and knowing when one is close to $\mathcal{K}^*(c)$". We try to overcome this difficulty by performing only a few steps of Maschler's algorithmic scheme and use the outcome to generate separating hyperplanes for the ellipsoid method, which we now briefly review.

3.1. The ellipsoid method

The ellipsoid method, as presented by Khachiyan [1979], was a major breakthrough in the theory of computational complexity, implying that linear programming problems can be solved in polynomial time. Subsequently, this algorithmic tool was sharpened by Grötschel, Lovász and Schrijver to show that, roughly speaking, the problems of separation and optimization over a polyhedron K are computationally equivalent in the sense that one can optimize a linear function over K efficiently, provided one can efficiently find, for any given $x \notin K$, a *separating hyperplane, i.e.*, a vector h and a rational number $\gamma \in \mathbb{Q}$ such that $h^T x \leq \gamma$ but $h^T y > \gamma$ holds for all $y \in K$. For our purposes, we can restrict ourselves to rational polyhedra. In order to describe the necessary background, we need some definitions.

A *rational polyhedron* is a subset $P \subseteq \mathbb{R}^n$ which can be described as

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

with a rational matrix A and a rational vector b.

We say that P has *facet complexity at most* $\varphi \in \mathbb{N}$, if there exists a system $Ax \leq b$ of linear inequalities with rational coefficients that has P as its solution set and has encoding length (*i.e.*, the number of bits necessary to represent each individual inequality of the system $Ax \leq b$) at most φ.

Example 3.1:

(1) Consider $P = \mathrm{core}(c) \subseteq \mathbb{R}^N$. P can be described via the inequalities

$$-x(N) \leq -c(N)$$

$$x(S) \leq c(S) \qquad (S \subseteq N).$$

Each of these inequalities has encoding length at most $|N| + \langle c \rangle$. Therefore, P has facet complexity at most $\varphi = |N| + \langle c \rangle$.

(2) Consider a game (N, c) where $P = \mathscr{K}^*(c) \cap \mathrm{core}(c)$ consists of a unique point x^*. By definition, then, there exist sets $S_{ij} \subseteq N$ with $i \in S_{ij}$, $j \notin S_{ij}$ such that $P = \{x^*\}$ is described by the following system

$$x(S_{ij}) - x(S_{ji}) = c(S_{ij}) - c(S_{ji}) \quad (i, j \in N)$$

$$x(S_{ij}) - x(S) \geq c(S_{ij}) - c(S) \quad (S \subseteq N, i \in S, j \notin S)$$

$$x(S) \leq c(S) \quad (S \subseteq N)$$

$$x(N) = c(N)$$

We conclude that P has facet complexity at most $\varphi = n + 2\langle c \rangle$. (Note that adding or subtracting two rational numbers $c = p/q$ and $c' = p'/q'$ of encoding length at most $\langle c \rangle$ each, results in a rational number $(pq' \pm p'q)/(qq')$ of encoding length at most $2\langle c \rangle$.) ◇

A *separation algorithm* for P is an algorithm *SEP* solving the *separation problem*: "Given a vector $x \in \mathbb{Q}^n$, decide whether $x \in P$ or not, and, in the latter case, find a vector $a \in \mathbb{Q}^n$ such that $a^T x < a^T y$ for all $y \in P$".

The following results are due to Grötschel, Lovász and Schrijver (*cf.* Schrijver [1986], Theorem 14.1 and Corollary 14.1a):

Theorem 3.1. *There exists an algorithm FEAS and a polynomial p in two variables n, φ such that the following holds:*

For each polyhedron $P \subseteq \mathbb{R}^n$ with facet complexity at most φ for which there exists a separation algorithm SEP, FEAS computes, on input n and φ, a vector in P or concludes that $P = \emptyset$ in time bounded by $T \cdot p(n, \varphi)$, where T is the maximum time required by SEP for inputs $x \in \mathbb{R}^n$ of size $p(n, \varphi)$. ◇

Corollary 3.1. *There exists an algorithm ELL such that ELL solves, on input (n, φ, SEP, d), the optimization problem*

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

in time polynomially bounded by n, φ, the size of d, and T, the maximum time required by SEP for inputs $x \in \mathbb{R}^n$ of size $p(n, \varphi)$.

To get an idea how the algorithm *FEAS* of Theorem 3.1 works, recall first the standard iteration of the ellipsoid method: Given a polyhedron $P \subseteq \mathbb{R}^n$ and an ellipsoid $E^{(i)} \supseteq P$ with center $x^{(i)}$, one computes an approximation $\tilde{x}^{(i)}$ of $x^{(i)}$ of size $\langle \tilde{x}^{(i)} \rangle \leq p(n, \varphi)$ and uses *SEP* in order to decide whether $\tilde{x}^{(i)} \in P$ (in which case the algorithm stops with output $\tilde{x}^{(i)}$) or not. In the latter case, *SEP* computes a separating hyperplane which cuts off one half of $E^{(i)}$. One then determines an ellipsoid $E^{(i+1)}$ containing the other half (and, in particular, P) with center $x^{(i+1)}$ and repeats the procedure. If P is full-dimensional, this algorithm produces some $x^{(k)} \in P$ after polynomially many steps.

The algorithm *FEAS* of Theorem 3.1 is somewhat more sophisticated so that it can also deal with lower dimensional polyhedra. If P is contained in some hyperplane H, it can happen that *SEP* returns separating hyperplanes that are parallel (or almost parallel) with H. *FEAS* will detect this situation and, using certain approximation techniques, will compute a hyperplane containing P. The algorithm then continues within this hyperplane as before. Details about *FEAS* can be found, *e.g.*, in the book of Schrijver [1986].

We conclude this section by illustrating the power of Corollary 3.1 and, assuming (CCM), derive a polynomial algorithm for the computation of an element in the least core of the game $(N, c) \in \mathscr{C}$.

Consider the linear program (LC):

$$\begin{aligned} \max \quad & \varepsilon \\ s.t \quad & x(N) = c(N) \\ & x(S) \leq c(S) - \varepsilon \quad \text{for all } \emptyset \neq S \neq N \end{aligned}$$

(LC) is clearly feasible. We next claim that the linear program (LC) is bounded (and hence has an optimal solution, which, by definition, will be in leastcore(c)). Indeed, adding all the inequalities

$$x_i \leq c(\{i\}) - \varepsilon,$$

and using $x(N) = c(N)$, we obtain the bound

$$\varepsilon \leq \frac{1}{|N|}\left(\sum_{i \in N} c(\{i\}) - c(N)\right).$$

Let P denote the polytope of feasible solutions of (LC). From Corollary 3.1, we know that we can solve (LC) in polynomial time if we can solve the separation problem for P in polynomial time. The latter can be done as follows. Having checked in polynomial time whether $x(N) = c(N)$ is satisfied, we turn to the other restrictions of (LC).

So let (x, ε) be given. Invoking (CCM), we either find out that

$$e_{min}(x) \geq \varepsilon$$

holds and (x, ε) is indeed feasible, or we can compute a coalition $\emptyset \neq S \neq N$ in polynomial time such that

$$c(S) - x(S) = e_{min}(x) < \varepsilon.$$

So $y(S) + \varepsilon \leq c(S)$ yields the desired separating inequality.

4. Approximating the prekernel

Suppose $x \in \mathbb{R}^N$ is an allocation for which we can find players $i, j \in N$ such that $s_{ij}(x) < s_{ji}(x)$, say $s_{ij}(x) = s_{ji}(x) - 2\alpha$, $\alpha > 0$. We then say that the allocation x' defined by

$$x'_k = \begin{cases} x_i - \alpha & \text{if } k = i \\ x_j + \alpha & \text{if } k = j \\ x_k & \text{else} \end{cases}$$

arises from x by a *transfer from i to j (of size α)* (or simply (i, j)-transfer). Note that α is not chosen arbitrarily here but depends on the surplusses. One can show that the vector of non-decreasingly ordered excesses at x' will be lexicographically greater than at x, *i.e.*, x' will be "closer" to the prenucleolus $\eta(c)$ than x. If no pair of players admits a transfer, then $x \in \mathscr{K}^*(c)$ holds by definition.

The idea to try to approximate the prekernel by iteratively carrying out transfers between pairs of players is due to Maschler. But it is not clear whether an arbitrary sequence of transfers necessarily converges to an allocation in the prekernel. Stearns [1968] proposes a version of such a transfer scheme for which convergence to the prekernel can be proved. In this section, we will take a closer look at the transfer scheme (and provide a short proof of Stearns' main result).

We say that the coalition $S \subseteq N$ *separates* the (ordered) pair (i, j) of players if $i \in S$ and $j \in N \backslash S$. Given the allocation $x \in \mathbb{R}^N$, the coalition $S \subseteq N$ is called $(i, j; x)$-*minimal* if S separates (i, j) and

$$s_{ij}(x) = c(S) - x(S) \quad (= e(S, x)).$$

Lemma 4.1. *Let the allocation $x' \in \mathbb{R}^N$ arise from x by a transfer from i to j of size $\alpha > 0$. Then*

(i) $s_{kl}(x) - \alpha \leq s_{kl}(x') \leq s_{kl}(x) + \alpha$ *for all $k, l \in N$.*
(ii) $s_{ij}(x') = s_{ji}(x') = s_{ij}(x) + \alpha$.

Proof: Consider an arbitrary coalition $S \subseteq N$. Then

$$x'(S) = \begin{cases} x(S) - \alpha & \text{if } S (i, j)\text{-separating} \\ x(S) + \alpha & \text{if } S (j, i)\text{-separating} \\ x(S) & \text{otherwise.} \end{cases}$$

Assume now that S is $(k, l; x)$-minimal. Then

$$s_{kl}(x) = e(S, x) = c(S) - x(S) \geq c(S) - x'(S) - \alpha$$
$$= e(S, x') - \alpha \geq s_{kl}(x') - \alpha.$$

The other inequality is proved similarly relative to a $(k, l; x')$-minimal coalition S'. ◇

Lemma 4.2. *Assume that the allocation $x' \in \mathbb{R}^N$ arises from the allocation $x \in \mathbb{R}^N$ by an (i, j)-transfer of size α and let (k, l) be a pair of players such that $s_{kl}(x) < s_{ij}(x)$. Then*

$$s_{kl}(x') = e(S, x') = e(S, x) = s_{kl}(x)$$

for every $(k, l; x)$-minimal coalition $S \subseteq N$.

Proof: Let $S \subseteq N$ be $(k, l; x)$-minimal. Then the hypothesis

$$e(S, x) = s_{kl}(x) < s_{ij}(x) < s_{ji}(x)$$

implies in particular that S separates neither (i, j) nor (j, i). Hence we have $x'(S) = x(S)$, which yields $s_{kl}(x') \leq s_{kl}(x)$.

The proof of $s_{kl}(x) \leq s_{kl}(x')$ is similar. ◇

For the allocation $x \in \mathbb{R}^N$, let us set

$$s(x) := \begin{cases} \infty & \text{if } x \in \mathscr{K}^*(c) \\ \min\{s_{ij} \mid s_{ij}(x) < s_{ji}(x)\} & \text{else.} \end{cases}$$

Hence we have $x \notin \mathscr{K}^*(c)$ if and only if $s(x) < \infty$. Moreover, $s_{kl}(x) < s(x)$ implies $s_{kl}(x) = s_{lk}(x)$ and, by Lemma 4.2, $s_{kl}(x') = s_{kl}(x)$ whenever x' arises from x by some (i, j)-transfer.

Lemma 4.3. *Assume that the allocation x' arises from the allocation x by an (i, j)-transfer of size α. Then*

(i) $s_{kl}(x') < s_{kl}(x)$ *implies* $s_{kl}(x') \geq s_{ij}(x') = s_{ij}(x) + \alpha$.
(ii) $s(x') \geq s(x)$.

Proof: Assume $s_{kl}(x') < s_{kl}(x)$ and let $S \subset N$ be $(k, l; x')$-minimal. Then the assumption implies that S is (j, i)-separating, and we deduce

$$s_{kl}(x') = e(S, x') \geq s_{ji}(x') = s_{ji}(x) - \alpha = s_{ij}(x) + \alpha,$$

which implies (i). (ii) is a direct consequence of (i). ◇

We aim at sequences of transfers that eventually yield a *strict* increase of $s(x)$. We therefore restrict our attention to a special type of transfer and call the (i, j)-transfer *canonical* if

$$s_{ij}(x) = s(x).$$

Proposition 4.1. *Given any allocation $x \notin \mathscr{K}^*(c)$, then one obtains an allocation $x^{(t)}$ satisfying $s(x^{(t)}) > s(x)$ after $t < |N|^2$ canonical transfers.*

Proof: Consider the set $I(x) := \{(k,l) \mid s_{kl}(x) = s(x)\}$ of pairs of players. Let x' be obtained via a canonical (i,j)-transfer of size α.

If $s(x') = s(x)$, then $(i,j) \notin I(x')$ while no new pair (k,l) enters $I(x')$. Indeed, if $s_{kl}(x) < s(x)$, then $s_{kl}(x') = s_{kl}(x) < s(x)$ by Lemma 4.2. If $s_{kl}(x) > s_{ij}(x) = s(x)$, then $s_{kl}(x') < s_{kl}(x)$ yields

$$s_{kl}(x') \geq s_{ij}(x) + \alpha > s(x)$$

by Lemma 4.3. So $s(x') = s(x)$ implies $|I(x')| \leq |I(x)| - 1$, *i.e.*, after at most $|I(x)| < |N|^2$ canonical transfers the desired allocation will be found. $\diamond$

If the new allocation $x^{(t)}$ from Proposition 4.1 is not in $\mathscr{K}^*(c)$, we could continue to compute a sequence of transfers which further increases $s(x^{(t)})$ *etc.*. Would such a procedure necessarily converge to an element in $\mathscr{K}^*(c)$? We do not know, but we suspect that the answer is negative. In order to ensure convergence to $\mathscr{K}^*(c)$, Stearns [1968] requires the sequence of transfers to contain an infinite number of maximal transfers. Here, a *maximal* transfer is, by definition, a transfer from i to j such that

$$s_{ji}(x) - s_{ij}(x) = \max_{k,l}\{s_{kl}(x) - s_{lk}(x)\}.$$

In order to make our paper selfcontained, we present a short alternative proof for Stearns' result. Given an allocation x, we order the surplusses non-decreasingly:

$$s_{i_1 j_1}(x) \leq \cdots \leq s_{i_m j_m}(x) \quad (m = |N|(|N| - 1)).$$

Define

$$\theta(x) := \sum_{r=1}^{m} 2^{m-r} s_{i_r j_r}(x).$$

Note that θ is well defined (even though the ordering is not unique).

Suppose that x' arises from x by an (i,j)-transfer. Assume $(i,j) = (i_p, j_p)$ and $(j,i) = (i_q, j_q)$ for some $q > p$ and $s_{ji}(x) - s_{ij}(x) = 2\alpha$. Let r be maximal such that $s_{i_r j_r}(x) \leq \beta := s_{i_p j_p}(x) + \alpha$, *i.e.*,

$$s_{i_1 j_1}(x) \leq \cdots \leq s_{i_r j_r}(x) \leq \beta < s_{i_{r+1} j_{r+1}}(x) \leq \cdots \leq \cdots \leq s_{i_m j_m}.$$

From Lemma 4.3, we know that no surplus is decreased below level β when passing from x to x'. In particular, the surplusses

$$s_{i_1 j_1}(x), \ldots, s_{i_r j_r}(x)$$

are not decreased at all and at least one of them is increased by α. The surplusses of value strictly larger than β are decreased by at most α (see Lemma 4.1). So the new ranking of this surplus that was increased will be at most r, and the new rankings of the surplusses that were decreased stay strictly greater than r. This implies

$$\theta(x') - \theta(x) \geq 2^{m-r}\alpha - 2^{m-r-1}\alpha - \cdots - 2^0\alpha = \alpha.$$

Lemma 4.4. *Assume that $x^{(0)}, x^{(1)}, \ldots$ is a sequence of allocations such that each $x^{(t)}$ arises from $x^{(t-1)}$ by an (i_t, j_t)-transfer of size $\alpha^{(t)}$. Then $(x^{(t)})$ converges to some allocation $x^* \in \mathbb{R}^N$.*

Proof: Let $s_{min}^{(t)} := \min\{s_{ij}(x^{(t)})\}$. Then

$$\theta(x^{(0)}) \geq 2^{m-1}s_{min}^{(0)} + \cdots + 2^0 s_{min}^{(0)} = (2^m - 1)s_{min}^{(0)}.$$

The proof of Proposition 4.1 shows that a transfer never decreases s_{min}, *i.e.*, $s_{min}^{(t)} \geq s_{min}^{(0)}$. Now consider any pair (i, j) and let

$$c_{max} := \max_{S \subseteq N} |c(S)|.$$

Choose any subset S with $i \in S$, $j \notin S$. Then we obtain the inequalities

$$s_{ij}(x^{(t)}) \leq c(S) - x^{(t)}(S)$$

$$s_{ji}(x^{(t)}) \leq c(N \backslash S) - x^{(t)}(N \backslash S).$$

Since $x^{(t)}$ is an allocation, we conclude

$$s_{ij}(x^{(t)}) + s_{ji}(x^{(t)}) \leq c(S) + c(N \backslash S) - c(N) \leq 3c_{max}.$$

Hence $s_{ij}(x^{(t)}) \leq 3c_{max} - s_{min}^{(0)}$ and, consequently,

$$\theta(x^{(t)}) \leq (2^m - 1)(3c_{max} - s_{min}^{(0)}).$$

This shows

$$\sum_{t \geq 1} \alpha^{(t)} \leq \lim_{t \to \infty} \theta(x^{(t)}) - \theta(x^{(0)}) \leq (2^m - 1)(3c_{max} - 2s_{min}^{(0)}) < \infty$$

and the claim of the Lemma follows. ◇

Theorem 4.1 (Stearns (1968)). *If $x^{(0)}, x^{(1)}, \ldots$ is a sequence of allocations such that each $x^{(t)}$ arises from $x^{(t-1)}$ by a transfer and if an infinite number of these transfers are maximal, then $(x^{(t)})$ converges to an element $x^* \in \mathscr{K}^*(c)$.*

Proof: Suppose that $x^{(t)}$ arises from $x^{(t-1)}$ by a transfer of size $\alpha^{(t)}$. If $\lim x^{(t)} =: x^* \notin \mathscr{K}^*(c)$, then there exists a pair (i, j) of players with

$$s_{ij}(x^*) = s_{ji}(x^*) - 2\alpha \quad (\alpha > 0).$$

Choose t_0 large enough so that

$$\sum_{t \geq t_0} \alpha^{(t)} < \alpha/4.$$

Then

$$\|x^{(t)} - x^*\|_1 = \sum_{i \in N} |x_i^{(t)} - x_i^*| < \alpha/2$$

for all $t \geq t_0$. In particular, $s_{ij}(x^{(t)}) < s_{ji}(x^{(t)}) - \alpha$ holds for all $t \geq t_0$, implying that none of the $\alpha^{(t)}$-transfers, $t \geq t_0$, can be maximal, a contradiction. $\diamond$

Although we have not been able to derive the analogue of Theorem 4.1 when restricting ourselves to canonical transfers only, it turns out that canonical transfers can be used in the design of polynomial algorithms for the computation of allocations in the prekernel. We will discuss such an algorithm in the next section.

5. Computing an element in $\mathscr{K}^*(c)$

Let $y \in \mathbb{R}^N$ be an allocation with $y \in \text{leastcore}(c)$ but $y \notin \mathscr{K}^*(c)$. If $\varepsilon^* = e_{min}(y) = s(y)$, we perform $t = O(|N|^2)$ canonical transfers to obtain some allocation $y^{(t)}$ with $s(y^{(t)}) > s(y) = \varepsilon^*$ (*cf.* Proposition 4.1). It follows from Lemma 4.2 and Lemma 4.3 that $y^{(t)}$ will also lie in leastcore(c). Hence there is no loss of generality when we assume from the outset that our $y \in \text{leastcore}(c)$ satisfies $s(y) > e_{min}(y)$.

We now assume $y \notin \mathscr{K}^*(c)$ and consequently order the surplusses of y so that

$$s_{i_1 j_1}(y) \leq \cdots \leq s_{i_r j_r}(y) < s(y) = s_{i_{r+1} j_{r+1}}(y) \leq \cdots$$

We call the (non-empty) ordered set of pairs

$$\mathscr{S}(y) = \{(i_1, j_1), \ldots, (i_r, j_r)\}$$

a *feasible collection of pairs* for y. Note that, by definition, the pairs in $\mathscr{S}(y)$ are symmetric in the sense that $(i,j) \in \mathscr{S}(y)$ if and only if $(j,i) \in \mathscr{S}(y)$.

For each pair $(i,j) \in \mathscr{S}(y)$, let S_{ij} be an $(i,j;y)$-minimal coalition. The ordered collection

$$\sigma(y) = \{S_{i_1 j_1}, \ldots, S_{i_r j_r}\}$$

is called a *feasible collection of sets* for y.

It will be convenient to introduce the following notation for all $i,j \in N$:

$$\mathscr{N}_{ij} := \{S \subseteq N \mid S \text{ separates}(i,j)\}.$$

Recalling the parameter $\varepsilon^* = \varepsilon(c) = \max\{e_{min}(x) \mid x \text{ allocation}\}$, which defines the least core of the game (N, c), consider the linear program

$$
\begin{array}{lll}
\mathbf{LP}(\sigma(y))): & \max \delta & \\
& s.t. & \\
(5.0) & x(N) = c(N) & \\
(5.1) & e(S_{ij}, x) = e(S_{ji}, x) & \text{for all } (i,j) \in S(y) \\
(5.2) & e(S_{ij}, x) \leq e(S, x) & \text{for all } (i,j) \in S(y), S \in \mathscr{N}_{ij} \\
(5.3) & e(S_{i_k j_k}, x) \leq e(S_{i_{k+1} j_{k+1}}, x) & \text{for } k = 1, \dots, r-1 \\
(5.4) & e(S_{i_r j_r}, x) \leq e(S, x) - \delta & \text{for all } (i,j) \notin S(y), S \in \mathscr{N}_{ij} \\
(5.5) & e(S_{i_1 j_1}, x) \geq \varepsilon^* &
\end{array}
$$

Note that all inequalities describing the feasible region of $\mathbf{LP}(\sigma(y))$ have size polynomially bounded in $|N|$ and $\langle c \rangle$, *i.e.*, the facet complexity of the feasible region is polynomially bounded (independent of the vector y).

By our assumption on y, $\mathbf{LP}(\sigma(y))$ is feasible (for example, y itself is a feasible solution with $\delta = s(y) - s_{i_r j_r}(y) > 0$). We claim that $\mathbf{LP}(\sigma(y))$ is bounded (and hence has an optimal solution (y^*, δ^*). Indeed, because $y \notin \mathscr{K}^*(c)$, there exists some $i, j \in N$ such that $(i,j) \notin \mathscr{S}(y)$. Let S be (i,j)-separating. Then the restrictions (5.4) relative to S and to its complement $N \backslash S$ imply in view of $x(N) = c(N)$ (condition (5.0)) and $\varepsilon^* \leq e(S_{i_r j_r}, x) - \delta$ (condition (5.3) and (5.5)):

$$2\delta \leq e(S,x) + e(N \backslash S, x) - 2\varepsilon^* \leq c(S) + c(N \backslash S) - c(N) - 2\varepsilon^* \leq 3c_{max} - 2\varepsilon^*.$$

Hence we conclude that $\mathbf{LP}(\sigma(y))$ has a finite optimal solution (y^*, δ^*). If (y^*, δ^*) is an optimal solution for $\mathbf{LP}(\sigma(y))$, then $y^* \in \text{leastcore}(c)$ (condition (5.5)) and $\mathscr{S}(y^*) \supseteq \mathscr{S}(y)$ (because $\delta^* > 0$). For our algorithm, the following property of y^* is crucial.

Lemma 5.1. *Assume that the allocation y' arises from y^* after a sequence of canonical transfers. Then $s(y') > s(y^*)$ implies $\mathscr{S}(y') \supset \mathscr{S}(y^*)$.*

Proof: It follows from Lemma 4.2 that each canonical transfer leaves any surplus of value less than $s(y^*)$ unchanged. Lemma 4.3 tells us that a canonical (i,j)-transfer from the allocation z to the allocation z' yields $s(z') \geq s(z)$. So we conclude $\mathscr{S}(z') \supseteq \mathscr{S}(z)$.

Hence, if $s(z') > s(z)$ and $\mathscr{S}(z') = \mathscr{S}(z)$ is satisfied, the canonical transfer must have raised *every* surplus $s_{kl}(z)$ satisfying $s_{kl}(z) = s(z)$. So we observe $s(z') - s_{i_p j_p}(z') > s(z) - s_{i_p j_p}(z)$ for all $(i_p, j_p) \in \mathscr{S}(z)$.

Since y^* optimizes δ in $\mathbf{LP}(\sigma(y))$, $s(y') > s(y^*)$ implies that $\mathscr{S}(y')$ contains $\mathscr{S}(y^*)$ strictly. ◇

Lemma 5.1 suggests the following algorithm for the computation of an allocation in the prekernel $\mathcal{K}^*(c)$:

Algorithm PREKER:

```
(0) (Initialization):
    (0.0) Compute an allocation $y^* \in$ leastcore$(c)$
    (0.1) Starting with $y^*$, perform canonical transfers until an allocation $y'$
          with $s(y') > s(y^*)$ is found;
(1) (Iteration):
    (1.0) IF $s(y') = \infty$ THEN output $y'$ and STOP;
    (1.1) Compute an optimal solution $y^*$ for the linear program $\mathbf{LP}(\sigma(y'))$;
    (1.2) Starting with $y^*$, perform canonical transfers until an allocation $y'$
          with $s(y') > s(y^*)$ is found;
    (1.3) GOTO (1.0);
```

Because $\mathcal{S}(y^*) \subset \mathcal{S}(y')$ holds in every iteration, Lemma 5.1, together with the observation $|\mathcal{S}(y)| < |N|^2$, implies that algorithm *PREKER* will stop after less than $|N|^2$ iterations and output some $y \in$ leastcore$(c) \cap \mathcal{K}^*(c)$. Moreover, *PREKER* will be a polynomial algorithm if the initialization and each iteration can be implemented to run in polynomial time. Assuming (CCM) or (CCS), we will show that the latter can indeed be achieved.

In Section 3, we have seen that the ellipsoid method allows us to find an allocation in the least core in polynomial time. Moreover, Proposition 4.1 shows that steps (0.1) and (1.2) require not more than $|N|^2$ transfers and hence can be carried out in polynomial time. It suffices, therefore, to prove that the linear program $\mathbf{LP}(\sigma(y))$ is solvable in polynomial time. As in the case of least core computations, we recall the polynomial equivalence of linear optimization and separation and approach the problem via Corollary 3.1: We show that the associated separation problem is polynomially solvable.

Let (x_0, δ_0) be given and suppose, for example, that $s_{ij}(x_0) - \delta_0 < e(S_{i_r j_r}, x_0)$ holds for some $(i,j) \notin \mathcal{S}(y)$, *i.e.*, that condition (5.4) is violated. Then we compute some $(i,j;x_0)$-minimal coalition $S \subseteq N$ and obtain

$$e(S,x) - \delta \geq e(S_{i_r j_r}, x)$$

as a separating inequality. From Corollary 3.1, we therefore deduce the existence of a polynomial algorithm that computes the desired feasible solution (y^*, δ^*).

We have therefore proved our main result:

Theorem 5.1. *Assume that (CCS) or, equivalently, (CCM) holds for the class $\mathcal{C}$ of cooperative games. Then there exists a polynomial algorithm that computes an allocation $x \in$ leastcore$(c) \cap \mathcal{K}^*(c)$ for every game $(N,c) \in \mathcal{C}$.*

6. Convex games

Since we have chosen the "cost" model of cooperative games, let us define "convex" games as those games (N,c) whose characteristic function $c : 2^N \to \mathbb{Q}$ is *submodular*, *i.e.*,

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

We assume that the cost function c is given by an *oracle* which, on input $S \subseteq N$, outputs $c(S)$ in time $T_c = \langle c \rangle$.

It is well-known that for convex games the (pre)kernel consists only of a single point and that the core is nonempty. Hence $\{x^*\} = \mathscr{K}^*(c) \cap \text{core}(c)$ yields the nucleolus (cf. Maschler *et al.* [1972]).

The fact that assumption (CCM) holds for this class of games is a standard result in combinatorial optimization (cf. Grötschel *et al.* [1993]).

A collection $\mathscr{M} \subseteq 2^N$ of subsets of N is called a *crossing family* if

$$S, T \in \mathscr{M}, \quad S \cap T \neq \emptyset, \quad S \cup T \neq N \Rightarrow S \cap T \in \mathscr{M}, \quad S \cup T \in \mathscr{M}.$$

Note that $\mathscr{M} = 2^N \backslash \{\emptyset, N\}$, for example, is a crossing family.

Given an allocation $x \in \mathbb{R}^N$, it is easy to see that the excess function $e(\cdot, x) : \mathscr{M} \to \mathbb{Q}$ is also submodular on the crossing family $\mathscr{M}$ whenever c is submodular. Therefore the following holds (cf. Grötschel *et al.* [1993] or, in particular, Iwata *et al.* [2000] and Schrijver [2000] for stronger results that avoid the ellipsoid method):

(CCM) for submodular games (N, c):
Given a submodular game (N, c) and an allocation $x \in \mathbb{R}^N$, one can compute

$$\min\{e(S, x) \mid \emptyset \neq S \neq N\}$$

in time polynomial in $|N|$ and $\langle c \rangle$. ◇

Thus we obtain the following result of Kuipers [1997] as a special case of our algorithm *PREKER*:

Theorem 6.1. *If $c : 2^N \to \mathbb{Q}$ is submodular, the nucleolus $\eta(c)$ can be computed in time polynomial in $|N|$ and $\langle c \rangle$.* ◇

7. MCST-games

In this section, we will apply our main result to the class of minimum cost spanning tree games (MCST-games). This class of games has been widely studied in the literature. After its introduction by Bird [1976], various results about the core and nucleolus were established (see, *e.g.*, Granot and Huberman [1981, 1984]). Megiddo [1987] and Granot *et al.* [1996] describe an $O(n^3)$ algorithm for computing the nucleolus in the special case where the underlying graph is a tree. Galil [1980] subsequently reduced the number of operations to $O(n \log n)$ and Granot and Granot [1992] consider an extended model in which they compute the nucleolus in strongly polynomial time.

In contrast with the positive results for some special cases, Faigle *et al.* [1998b] show that the problem of computing the nucleolus for an MCST-game is *NP*-hard if there are no restrictions on the underlying graph.

Of interest for us is a result of Granot and Huberman [1984] who show that the intersection of core and prekernel of an MCST-game consists of precisely the nucleolus. According to Theorem 5.1, polynomial solvability of the minimum excess problem for MCST-games would imply that the nucleolus of an MCST-game could be computed in polynomial time. The result of Faigle *et al.* shows that this is very unlikely for the general MCST-game. However, we will indicate a subclass of MCST-games for which the minimum excess problem is solvable in polynomial time. For this subclass we can therefore conclude that the nucleolus is computable in polynomial time.

First, however, let us rigorously define what we understand by an MCST-game. Denote by $N = \{1, \dots, n\}$ a set of customers who all need to be connected to some common supplier denoted by 0. Given is an undirected connected graph $G = (N \cup \{0\}, E)$, together with a non-negative weight function $w : E \to \mathbb{R}_+$. The weighted graph determines the cost of establishing a link between a pair $i, j \in N \cup \{0\}$: it is the minimal weight sum on a path between i and j in G. The cost $c(S)$ of coalition $S \subseteq N$ is defined as the cost of a minimum spanning tree on $S \cup \{0\}$. The game (N, c) defined in this way is called a *minimum cost spanning tree game*. The graph G is called the *underlying graph* of the game.

The minimal excess problem is closely related to the so-called vertex weighted Steiner tree problem. Let us give a description of this problem. Let (V, E) be an undirected graph, let $w : E \to \mathbb{R}_+$ be a non-negative weight function on the edges, and let $r : V \to \mathbb{R}$ be a reward function on the vertices. For any set $U \subseteq V$ the *cost* is defined as

$$k(U) - r(U),$$

where $k(U)$ is the weight of a minimum spanning tree on U. The *vertex weighted Steiner tree* (VWST) problem is the problem of determining a set $U \supseteq T$ of minimal cost, where $T \subseteq V$ is a specified non-empty set of so-called *terminal nodes*. The problem was first treated by Segev [1987].

Observe that minimizing $c(S) - x(S)$ for an MCST-game (N, c) and some input vector x is precisely the VWST problem on the underlying graph with root 0 as the only terminal node, and reward x_i on vertex $i \in N$ (reward 0 on node 0). Applying our main Theorem 5.1, we obtain the following corollary.

Corollary 7.1. *Let $\mathcal{G}$ be a class of graphs, $\mathcal{C}(\mathcal{G})$ be the class of MCST-games whose underlying graph is in $\mathcal{G}$, and $\mathcal{V}(\mathcal{G})$ the class of VWST problems whose underlying graph is in $\mathcal{G}$. Then a polynomial algorithm for $\mathcal{V}(\mathcal{G})$ implies a polynomial algorithm for computing the nucleolus for $\mathcal{C}(\mathcal{G})$.* $\diamond$

The general VWST problem is *NP*-hard. This is well-known, but may also be deduced from the fact that it is 'harder' than the problem of computing the nucleolus for a MCST-game, which is already an *NP*-hard problem (*cf.* Faigle *et al.* [1998b]).

Borie *et al.* [1992], however, introduce a relatively large class of so-called *recursively constructed* graphs, for which the VWST-problem (and many others) can be solved even in linear time. Basically the idea is to construct graphs from a finite set of *base graphs*.

Roughly, a recursively constructed graph is then a graph which is either one of those base graphs or can be obtained by merging (identifying specified nodes) of a number of already constructed graphs. Well-known examples of recursively constructed graphs are trees, outer-planar graphs, series-parallel graphs, Halin graphs, bandwidth-k graphs, and partial k-trees (graphs with treewidth $\leq k$ for some fixed $k \in \mathbb{N}$). Theorem 7 of Borie *et al.* [1992] states that, *e.g.*, the VWST-problem is linear time solvable for such graphs. Combining this fact with Corollary 7.1, we conclude that for all those classes of graphs, the nucleolus of the corresponding MCST-game can be computed in polynomial time.

8. Open problems and remarks

Our algorithm has some similarities with the standard prodecure for computing the nucleolus via a sequence of linear programs of decreasing dimension. We suspect that this standard procedure will, in general, increase the facet complexity in each step. In particular, we surmise that the size of the nucleolus is not polynomially bounded in general. This is the reason why we work with the concept of "feasible collections" of sets in our linear programs $\mathbf{LP}(\sigma(y))$.

The complexity result we have derived is a theoretical one and gives little insight in how an element in leastcore$(c) \cap \mathscr{K}^*(c)$ should best be calculated in practice. In Kuipers *et al.* [2000], an $O(n^3|\mathscr{E}|)$ algorithm is described for computing the prenucleolus of games whose collection of 'essential' coalitions $\mathscr{E}$ possess a certain combinatorial structure. The class of MCST-games is one of the examples to which the algorithm can be applied. Interesting here is the structure of this algorithm: it finds the prenucleolus by computing $O(n^2)$ 'approximations' of the nucleolus. The computation of one such approximation vector requires the solution of $O(n)$ minimum excess problems.

The minimum excess problems do not correspond directly to the game for which the prenucleolus is computed, since this game is adapted during the computations, and the minimum excess problem has to be solved with respect to the adapted game. However, a close examination of the way the original game is adapted shows that one can solve the minimum excess problem for the adapted game by solving the problem for the original game plus an extra amount of work that is linear in the number of players. For an MCST-game this means that its nucleolus can be computed in $O(n^3m)$ time if the underlying graph is a recursively constructible graph, where n and m denote the number of vertices and edges in the graph. In view of our results, it would be interesting to look for other classes of games for which core$(c) \cap \mathscr{K}^*(c)$ is a singleton.

Finally, it would be interesting to know whether our intuitive reasoning that (CCS) is the "minimal" assumption we need for computing an element in the prekernel $\mathscr{K}^*$ efficiently can be made hard in the sense that one can prove that an efficient algorithm for the computation of an element in $\mathscr{K}^*$ implies (CCS).

Acknowledgement. We thank the anonymous referees for their careful reading of the manuscript, which helped to improve the presentation.

References

Bird CG (1976) On cost allocation for a spanning tree. A game theoretic approach. Networks 6:335–350

Borie RB, Parker RG, Tovey CA (1992) Automatic generation of linear time algorithms from predictable calculus descriptions of problems on recursively constructed families of graphs. Algorithmica 7:555–581

Deng X, Papadimitriou CH (1994) On the complexity of cooperative game solution concepts. Mathematics of Operations Research 19:257–266

Deng X, Ibaraki T, Nagamochi H (1997) Combinatorial optimization games. Proc. 8th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, 720–729

Deng X, Ibaraki T, Nagamochi H, Zang W (2000) Totally balanced combinatorial optimization games. Mathematical Programming 87:441–452

Faigle U, Fekete S, Hochstättler W, Kern W (1997) On the complexity of testing membership in the core of min cost spanning tree games. Int. Journal of Game Theory 26:361–366

Faigle U, Fekete S, Hochstättler W, Kern W (1998a) The nucleon of cooperative games and an algorithm for matching games. Mathematical Programming 83:195–211

Faigle U, Kern W, Kuipers J (1998b) Computing the nucleolus of min-cost spanning tree games is *NP*-hard. International Journal of Game Theory 27:443–450

Faigle U, Kern W, Paulusma D (2000) Note on the computational complexity of least core concepts for min-cost spanning tree games. To appear: Math. Methods of Operations Research

Galil Z (1980) Applications of efficiently mergeable heaps for optimization problems on trees. Acta Informatica 13:53–58

Granot D, Granot F (1992) Computational complexity of a cost allocation approach to a fixed cost spanning forest problem. Mathematics of Operations Research 17:765–780

Granot D, Huberman G (1981) Minimum cost spanning tree games. Mathematical Programming 21:1–18

Granot D, Huberman G (1984) On the core and nucleolus of minimum cost spanning tree games. Mathematical Programming 29:323–347

Granot D, Maschler M, Owen G, Zhu WR (1996) The kernel/nucleolus of a standard tree game. Int. Journal of Game Theory 25:219–244

Grötschel M, Lovász L, Schrijver A (1981) The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 5:169–179

Grötschel M, Lovász L, Schrijver A (1993) Geometric algorithms and combinatorial optimization. Springer-Verlag, Berlin

Iwata S, Fleischer L, Fujishige S (2000) A combionatorial strongly polynomial-time algorithm for minimizing submodular functions. 32. STOC 2000:97–106

Khachiyan LG (1979) A polynomial time algorithm in linear programming (in Russian). Doklady Akademii Nauk SSSR 244:1093–1096 (English translation: Soviet Mathematics Doklady 20:191–194)

Kuipers J (1996) A polynomial time algorithm for computing the nucleolus of convex games. Report M 96–12, Maastricht University

Kuipers J, Solymosi T, Aarts H (2000) Computing the nucleolus of some combinatorially structured games. Mathematical Programming 88:541–563

Maschler M, Peleg B, Shapley LS (1972) The kernel and the bargaining set for convex games. Int. Journal of Game Theory 1:73–93

Maschler M, Peleg B, Shapley LS (1979) Geometric properties of the kernel, nucleolus, and related solution concepts. Mathematics of Operations Research 4:303–338

Megiddo N (1987) Computational complexity of the game theory approach to cost allocation for a tree. Mathematics of Operations Research 3:189–196

Schrijver A (2000) A combinatorial algorithm minimizing submodular functions in strongly polynomial time. J. Comb. Theroy B 80 (2000):346–355

Schmeidler D (1969) The nucleolus of a characteristic function game. SIAM J. of Applied Mathematics 17:1163–1170

Segev A (1987) The node-weighted Steiner tree problem. Networks 17:1–18

Solymosi T, Raghavan TES (1994) An algorithm for finding the nucleolus of assignment games. International Journal of Game Theory 23:119–143

Stearns RE (1968) Convergent transfer schemes for *n*-person games Trans. Amer. Math. Soc. 134:449–459