

Implementing the simplex method for computing the prenucleolus of transferable utility games

Jean Derks Jeroen Kuipers*

April, 1997

Abstract

The prenucleolus of a transferable utility n - persons game is usually computed by solving a sequence of linear programs. In this paper we describe how the simplex method can be implemented for this type of linear programs in order to reduce the amount of work in each pivoting step by a factor n compared to the standard implementation. Theoretically, it is impossible to improve upon the complexity of our implementation of a pivoting step.

1 Introduction

In the past years several algorithms have been proposed for computing the nucleolus or prenucleolus of transferable utility games. These algorithms have in common that they compute the (pre)nucleolus by solving one or more linear programs. For instance, Kohlberg (1972) and Owen (1974) give one linear program of which the outcome determines the nucleolus. For an n -person game, Kohlberg uses $\mathcal{O}(n)$ variables and $(2^n)!$ constraints in his linear program, and Owen needs $\mathcal{O}(2^n)$ variables and $\mathcal{O}(4^n)$ constraints. These linear programs are interesting from a theoretical viewpoint, but because of their size they are not used in practice for computing the nucleolus.

The fastest algorithms nowadays compute the (pre)nucleolus by solving a sequence of smaller linear programs. The first algorithm of this type was already described in an unpublished paper by Kopelowitz (1967). Other algorithms are by Dragan (1981), Sankaran (1991), Solymosi (1993), Zumsteg

*Department of Mathematics, University of Maastricht, the Netherlands.

(1995), and Potters et al. (1996). Not all of the papers above mention explicitly how these linear programs should be solved, but if they do mention some method, it is always a version of the well-known simplex method. The linear programs involved are still large, and in all cases a simplex tableau with at least $\mathcal{O}(n2^n)$ entries is required. Performing one pivoting operation in such a tableau requires $\mathcal{O}(n2^n)$ elementary operations if the simplex method is implemented in the standard way. In this paper we exploit the special structure of the linear programs involved in order to reduce the complexity of one pivoting operation to $\mathcal{O}(2^n)$. Since the number of restrictions in each linear program is $\mathcal{O}(2^n)$ and since at each pivoting step it is necessary to inspect all restrictions, it is theoretically impossible to improve upon the complexity of our implementation of a pivoting step.

Our algorithm arrives at the prenucleolus after solving at most $n - 1$ linear programs. It has this important feature in common with the algorithms of Solymosi (1993) and Potters et al. (1996). In the algorithm of Potters et al. this feature comes automatically with their implementation of the simplex method. The algorithm of Solymosi obtains this feature by a subroutine that is executed between each two linear programs. Also our algorithm uses such a subroutine. It is shown that our version of the subroutine is of the same complexity as one pivoting step, hence this subroutine does not disturb the good performance of our algorithm.

Another advantage of our algorithm is that it is unnecessary to reserve memory space for a simplex tableau with $\mathcal{O}(n2^n)$ coefficients. Our implementation requires memory space for $\mathcal{O}(n^2)$ variables and for the $2^n - 1$ coefficients of the game. Since these $2^n - 1$ coefficients are accessed in a sequential way and require no adaptations during the calculations, it is possible to store them on disk, thus enabling us to compute the prenucleolus for games with a relatively large number of players.

In section 2 of this paper we recall how the prenucleolus can be computed by solving a sequence of linear programs. In section 3 we describe an implementation of the simplex method specialized for the type of linear programs we have encountered in section 2. In section 4 we show how one pivoting operation can be speeded up by a factor n . Some empirical findings are presented in section 5.

2 Computing the prenucleolus of a game

A transferable utility game is a pair (N, v) , where $N = \{1, 2, \dots, n\}$ is a finite set and v is a real valued function on the subsets of N , assigning 0 to the empty set. The subsets of N are called coalitions. For a vector $x \in \mathbb{R}^N$ and a coalition $S \subseteq N$ we denote $\sum_{i \in S} x_i$ by $x(S)$. The value $v(S) - x(S)$ is called the excess of coalition S at x . Let $\theta(x)$ be the vector of all excesses at x ($S \subseteq N$) arranged in order of non-increasing magnitude. We say that a vector θ is lexicographically greater than ϕ if $\theta \neq \phi$ and if the first non-zero coordinate of the vector $\theta - \phi$ is positive. Now, the prenucleolus is defined as the set of vectors in \mathbb{R}^N which lexicographically minimize the vector $\theta(x)$ over the set $\{x \in \mathbb{R}^N \mid x(N) = v(N)\}$ of efficient vectors. It was proved by Schmeidler (1969) that the prenucleolus consists of one point. We denote the prenucleolus of (N, v) by η_v .

For a game (N, w) and a collection of coalitions $\mathcal{F} \subseteq 2^N$ we define the linear program $\mathcal{P}(\mathcal{F}, w)$ by

$$\begin{aligned} \mathcal{P}(\mathcal{F}, w) : \quad & \text{minimize} \quad t \\ & \text{subject to} \quad x(S) = w(S) \quad S \in \mathcal{F} \\ & \quad \quad \quad x(S) + t \geq w(S) \quad S \notin \mathcal{F}. \end{aligned}$$

The prenucleolus of a game can be computed by solving a sequence of at most $n - 1$ linear programs of the above type. Before we present this procedure, we have to introduce some terminology. For a coalition $S \subseteq N$ we define the *incidence* vector 1_S by

$$1_{S,i} = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S. \end{cases}$$

A collection $\mathcal{F} \subseteq 2^N$ is called *saturated* if every coalition whose incidence vector can be written as a linear combination of incidence vectors of coalitions in \mathcal{F} is a member of \mathcal{F} itself. We define the *saturation* $\overline{\mathcal{F}}$ of \mathcal{F} as the intersection of all saturated collections that contain \mathcal{F} . Observe that $\overline{\mathcal{F}}$ itself is saturated, i.e. $\overline{\mathcal{F}}$ is the smallest saturated collection that contains \mathcal{F} .

Now, we wish to compute the prenucleolus of the game (N, v) . Initialize $\mathcal{F}^0 = \{N\}$ and $w^0 = v$. Observe that $\mathcal{P}(\mathcal{F}^0, w^0)$ is feasible and bounded. Let t^0 be the optimal value of problem $\mathcal{P}(\mathcal{F}^0, w^0)$ and let \mathcal{A}^0 be a non-empty collection such that

$$\mathcal{A}^0 \subseteq \{S \notin \mathcal{F}^0 \mid x(S) = y(S) \text{ for all optimal } (x, t^0) \text{ and } (y, t^0)\}.$$

Later in this section we show how such a collection can be found by inspection of the optimal dual solution of problem $\mathcal{P}(\mathcal{F}^0, w^0)$. The collection \mathcal{F} is updated according to $\mathcal{F}^1 = \overline{\mathcal{F}^0 \cup \mathcal{A}^0}$, and the game w is updated according to

$$w^1(S) = \begin{cases} x(S) & \text{if } S \in \mathcal{F}^1 \\ w^0(S) - t^0 & \text{otherwise.} \end{cases}$$

Observe that problem $\mathcal{P}(\mathcal{F}^1, w^1)$ is again feasible and bounded. Next, this problem is solved, etc. It was shown in Maschler et al. (1979) that (η_v, t^k) is an optimal solution for every problem $\mathcal{P}(\mathcal{F}^k, w^k)$. Since in each iteration the collection \mathcal{F}^k is growing with at least one coalition, the solution space of $\mathcal{P}(\mathcal{F}^k, w^k)$ will finally consist of a single point, and we have found the prenucleolus.

Our claim is that at most $n - 1$ linear programs are needed to arrive at the prenucleolus. That this is possible was first noted by Dragan (1981), and also the algorithms by Solymosi (1993) and Potters et al. (1996) have this property. That it is true for our algorithm is seen as follows. Since \mathcal{F}^k is a saturated collection for all k , it follows that there exists a $(p^k \times n)$ -matrix F^k of rank $p^k \leq n$ such that the incidence vectors of the coalitions in \mathcal{F}^k are precisely the incidence vectors that can be written as a linear combination of the rows of F^k . Since \mathcal{F}^{k+1} contains \mathcal{F}^k as a proper subset, we must have $p^{k+1} > p^k$. The process starts with $p^0 = 1$, hence we have $p^k \geq k + 1$, and as a consequence $\mathcal{F}^{n-1} = 2^N$.

The representation of the collection \mathcal{F}^k by means of the matrix F^k also allows for a more compact description of problem $\mathcal{P}(\mathcal{F}^k, w^k)$. Note that (x^{k-1}, t^{k-1}) is a feasible solution of problem $\mathcal{P}(\mathcal{F}^k, w^k)$. Problem $\mathcal{P}(\mathcal{F}^k, w^k)$ may now be written as

$$\mathcal{Q}(F^k, x^{k-1}, w^k) := \min\{t \mid F^k x = F^k x^{k-1}, A^k x + 1t \geq w^k\}.$$

Here, A^k is the $(0, 1)$ -matrix whose rows are precisely the $(0, 1)$ -row vectors of length n that cannot be written as a linear combination of the rows of F^k , and 1 is the all-one column vector of appropriate length. A further simplification is possible. Observe that for coalitions S which are not in \mathcal{F}^k we have $w^k(S) = v(S) - \sum_{l=0}^{k-1} t^l$. Hence, problem $\mathcal{Q}(F^k, x^{k-1}, w^k)$ is equivalent to problem

$$\mathcal{Q}(F^k, x^{k-1}, v) = \min\{t \mid F^k x = F^k x^{k-1}, A^k x + 1t \geq v\},$$

since (x, t) is an optimal solution for $\mathcal{Q}(F^k, x^{k-1}, w^k)$ if and only if $(x, t + \Delta^k)$ is an optimal solution for $\mathcal{Q}(F^k, x^{k-1}, v)$, where $\Delta^k := \sum_{l=0}^{k-1} t^l$. Hence,

during the computations it is unnecessary to update the coalitional values of the game.

Let us now concentrate on the updating of the collection \mathcal{F}^k after having solved problem $\mathcal{Q}(F^k, x^{k-1}, v)$. Since the collection \mathcal{F}^k is completely determined by the matrix F^k , it is sufficient to explain how this matrix is updated. The dual of problem $\mathcal{Q}(F^k, x^{k-1}, v)$ is given by

$$\max\{zF^k x^{k-1} + uv \mid zF^k + uA^k = 0, u1 = 1, u \geq 0\}.$$

Since each problem $\mathcal{Q}(F^k, x^{k-1}, v)$ is feasible and bounded, it follows that solving this problem with the simplex method provides us with both a primal optimal solution (\bar{x}, \bar{t}) and a dual optimal solution (ν, μ) (see section 3).

Clearly, we have that $\mu_i > 0$ for some i . Let a_i^k be the corresponding row of the matrix A^k . Then $a_i^k \bar{x} + \bar{t} = v_i$, since otherwise

$$\begin{aligned} \nu F^k x^{k-1} + \mu v &< \nu F^k x^{k-1} + \mu(A^k \bar{x} + 1\bar{t}) \\ &= (\nu F^k + \mu A^k) \bar{x} + \mu 1\bar{t} \\ &= \bar{t}, \end{aligned}$$

contradicting the fact that the values of the primal and dual solution are the same (this phenomenon is known as the complementary slackness principle). It follows that $a_i^k x$ is constant for all optimal solutions (x, \bar{t}) , while the vector a_i^k is not in the row space of F^k . Hence, the matrix F^k can be updated by adding the row a_i^k to it. If the vector μ has more than one coordinate which is positive, then it is possible to add more than one row to F^k . However, simply adding all rows corresponding to positive coordinates of μ may cause dependency between the rows of the matrix F^{k+1} . If this happens, then dependent rows have to be removed afterwards.

In section 3 we describe the simplex method for linear programs of the type

$$\min\{t \mid Fx = c, Ax + 1t \geq b\},$$

where F is a matrix whose rows are linearly independent, and where the system $Fx = c$ has at least one solution. Problem $\mathcal{Q}(F^k, x^{k-1}, v)$ is a linear program of this type with the extra property that the matrix A^k is implicitly given by F^k . We show in section 4 that this latter property enables us to perform each pivoting step in $\mathcal{O}(2^n)$ operations.

3 An implementation of the simplex method

Suppose that we wish to solve the problem

$$\mathcal{P} := \min\{t \mid Fx = c, Ax + 1t \geq b\},$$

where A is an $m \times n$ -matrix, F is a $p \times n$ matrix of rank p , 1 is the all-one column vector of length m , and b is a column vector of length m . We assume that the system $Fx = c$ has at least one solution, since otherwise the problem is infeasible.

Choose a feasible solution (x^0, t^0) and a subsystem $A^0x + 1t \geq b^0$ of $Ax + 1t \geq b$ such that $A^0x^0 + 1t^0 = b^0$ and such that the rows of the matrix $\begin{pmatrix} F & 0 \\ A^0 & 1 \end{pmatrix}$ are linearly independent. That such a choice is possible is seen as follows. Let x^0 be a solution of the system $Fx = c$ and define $t^0 = \max\{b_i - a_i x^0 \mid i = 1, \dots, m\}$. Then (x^0, t^0) is a feasible solution. Moreover, there exists a row a_i of the matrix A such that $a_i x^0 + t^0 = b_i$. Hence, if we define A^0 as the matrix consisting of the single row a_i , then x^0 , t^0 and A^0 satisfy the above requirements. Observe that the system

$$zF + uA^0 = 0, u1 = 1$$

has at most one solution. Consider three cases.

Case 1. The system $zF + uA^0 = 0, u1 = 1$ has a (unique) solution $(z, u) = (\nu, \mu)$ with $\mu \geq 0$. Then (x^0, t^0) is an optimal solution of \mathcal{P} , since

$$\begin{aligned} t^0 &= (\nu F + \mu A^0)x^0 + \mu 1t^0 \\ &= \nu Fx^0 + \mu(A^0x^0 + 1t^0) \\ &= \nu c + \mu b \\ &\leq \max\{zc + ub \mid zF + uA = 0, u1 = 1, u \geq 0\} \\ &= \min\{t \mid Fx = c, Ax + 1t \geq b\}. \end{aligned}$$

Case 2. The system $zF + uA^0 = 0, u1 = 1$ has a solution $(z, u) = (\nu, \mu)$ for which at least one coordinate of μ is negative. Let \hat{i} be the smallest index for which $\mu_{\hat{i}} < 0$ and let B^0 be the matrix that results from A^0 by deleting the row corresponding to \hat{i} , and let $\bar{\mu}$ be the vector that results from μ by deleting the \hat{i} -th coordinate. Since the solution for the system $zF + uA^0 = 0, u1 = 1$ is unique, it follows that it has no solution with $u_{\hat{i}} = 0$. Equivalently, the system $zF + uB^0 = 0, u1 = 1$ has no solution. Applying Farkas' lemma it

follows that the system $Fy = 0, B^0y = 1$ has a solution. Let y^0 be a solution of this system. Let a_i denote the row that is deleted from A^0 . Observe that $a_i y^0 > 1$, since $\mu_i a_i y^0 = -(\nu F + \bar{\mu} B^0)y^0 = -\bar{\mu}1 = \mu_i - 1$.

Case 3. The system $zF + uA^0 = 0, u1 = 1$ has no solution. Similar to case 2, it then follows that $Fy = 0, A^0y = 1$ has a solution. Let y^0 be such a solution, and define $B^0 = A^0$ in this case.

In case 1. we are finished. In case 2. and 3. we have found a vector y^0 , such that $Fy^0 = 0$ and $A^0y^0 \geq 1$. If $Ay^0 \geq 1$, then the vector $(x^0 + \lambda y^0, t^0 - \lambda)$ is feasible for all $\lambda \geq 0$, and hence the problem is unbounded. Also in this situation we are finished. Finally, consider the situation that $ay^0 < 1$ for some row a of A . Then let λ^0 be the largest λ such that $(x^0 + \lambda y^0, t^0 - \lambda^0)$ is feasible, i.e.

$$\lambda^0 = \min\left\{\frac{a_j x^0 + t^0 - b_j}{1 - a_j y^0} \mid a_j y^0 < 1\right\},$$

and let \hat{j} be the smallest index attaining this minimum.

Let $(x^1, t^1) = (x^0 + \lambda^0 y^0, t^0 - \lambda^0)$ and let A^1 arise from B^0 by adding the row $a_{\hat{j}}$ to it. Observe that the rows of $\begin{pmatrix} F & 0 \\ A^1 & 1 \end{pmatrix}$ are linearly independent.

Moreover, we have $A^1 x^1 + 1t^1 = b^1$, where b^1 is the part of b corresponding to A^1 . Hence, we may repeat the procedure with A^1, x^1 , etc.

The rule for updating the matrix A^i is known as Bland's pivoting rule. Bland (1977) showed that the process terminates if this pivoting rule is applied, and when the process terminates we have either found an optimal solution or we have detected that the problem is unbounded.

The above description of the simplex method suggests a straightforward implementation, which involves the following amount of work in each iteration.

- Solve the system $zF + uA^i = 0, u1 = 1$.
- Solve the system $Fy = 0, B^i y = 1$.
- Compute the optimal stepsize λ^i .

Both systems of linear equations can be solved in $\mathcal{O}(n^3)$ operations and the optimal λ^i can be computed in $\mathcal{O}(mn)$ operations. For the linear programs

that need to be solved in order to find the prenucleolus, we have $m = \mathcal{O}(2^n)$, so that the computational complexity of performing one pivoting step is $\mathcal{O}(n2^n)$. In the following section we will see that this can be improved to $\mathcal{O}(2^n)$.

4 Solving problem $\mathcal{Q}(F, d, b)$

We wish to solve problem $\mathcal{Q}(F, d, b)$ which we defined in section 1, where F is a $p \times n$ matrix of rank p , d is a column vector of length n , and b is a column vector of appropriate length. Problem $\mathcal{Q}(F, d, b)$ is a problem of the type $\min\{t \mid Fx = c, Ax + 1t \geq b\}$, hence the simplex method as described in section 2 can be applied to this problem. We will exploit the fact that the matrix A is given implicitly by the matrix F : it is the matrix consisting of all possible $(0, 1)$ -rows of length n that cannot be written as a linear combination of the rows of F . We assume that F has less than n rows, since otherwise solving the linear program is trivial. One can easily prove by induction that in this case the matrix A has at least 2^{n-1} rows. Hence, if the linear program is solved by transforming a simplex tableau, a tableau with $\mathcal{O}(n2^n)$ entries is needed, and each pivoting step requires $\mathcal{O}(n2^n)$ elementary operations. We want to do better.

Let us arrange all $(0, 1)$ -vectors of length n (except the null- vector) in lexicographic increasing order. Denote the i -th vector by a_i . Define the vector f of length $2^n - 1$ by $f_i = 1$ if a_i is a row of A , and $f_i = 0$ otherwise. For the moment we will assume that the vector f is given. Later we will show that it can be computed in $\mathcal{O}(2^n)$ operations.

Recall that in one pivoting step we have to solve (at most) two systems of linear equations, after which the optimal stepsize λ is computed defined by

$$\lambda = \min\left\{\frac{a_i x + t - b_i}{1 - a_i y} \mid f_i = 1 \text{ and } a_i y < 1\right\}.$$

Here (x, t) is the current feasible solution, and y is the improving direction. Solving the two systems of linear equations requires only $\mathcal{O}(n^3)$ operations, and hence the computational complexity of performing one pivoting step is determined by the computation of the stepsize λ .

Let a and b be two consecutive $(0, 1)$ -rows and let j be the rightmost position of a 0 in a . Let u^j denote the vector with zeros at positions 1 to $j-1$, a 1 at position j and -1 at positions $j+1$ to n . Observe that $b = a + u^j$. Consider the following procedure which runs through all possible $(0, 1)$ -row

vectors of length n , starting with the lexicographic smallest vector e^n , and computes λ .

```

 $a := e^n; ax = x_n; ay = y_n;$ 
if ( $ay < 1$  and  $f_1 = 1$ ) then  $\lambda := \frac{ax+t-b_1}{1-ay}$ ; else  $\lambda := \infty$ ; fi;
for  $i := 2$  to  $2^n$ 
begin
   $j := n$ ; while ( $j > 0$  and  $a_j = 1$ ) do  $j := j - 1$ ; od;
   $a := a + u^j$ ;
   $ax := ax + u^j x$ ;
   $ay := ay + u^j y$ ;
  if ( $ay < 1$  and  $f_i = 1$ ) then  $\lambda := \min(\lambda, \frac{ax+t-b_i}{1-ay})$ ; fi;
end;

```

The first line in the loop is simply meant to compute the rightmost position of a 0 in the vector a . Clearly, the number of elementary operations within each loop is bounded by $\mathcal{O}(n)$. Hence, the complexity of one pivoting step is surely bounded by $\mathcal{O}(n2^n)$, but we wish to derive a sharper complexity bound. Observe that the first $j-1$ positions of the vector u^j are zero. Hence, if j is the rightmost position of a 0 of a in some loop, then the number of elementary operations in this loop is a (constant) multiple of $n-j+1$. The number of $(0, 1)$ -vectors for which the rightmost position of a 0 is j equals 2^{j-1} . Hence, the complexity of entering the loop $2^n - 1$ times is

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

The performance of the procedure can be improved somewhat if all inner products $u^j x$ and $u^j y$ ($j = 1, \dots, n$) are computed and stored beforehand.

Now, let us turn to the problem of constructing the vector f . We may assume that the matrix F is upper triangular and that the first non-zero entry in each row of F is a 1. If this is not the case, it can be achieved in $\mathcal{O}(n^3)$ elementary operations without changing the rowspace of F . Now define the matrix H and the vector h as follows. If F contains a row whose leftmost non-zero position is at coordinate i , then this row will be the i -th row of the matrix H , and we define $h_i := 0$. Otherwise, the i -th row of H is defined to be the i -th unit vector and $h_i := 1$. The matrix H is a square upper triangular matrix with ones on the diagonal. Hence, for each

row vector a the equation $\alpha H = a$ has exactly one solution. Observe that a $(0, 1)$ -vector a is not a row of A if and only if $\alpha_i h_i = 0$ for all $i = 1, \dots, n$.

Let u^j denote again the vector with zeros at coordinates 1 to $j - 1$, a 1 at position j and -1 at positions $j + 1$ to n . For each j we compute the solution μ^j of the equation $\alpha H = u^j$ in a preprocessing procedure. This requires $\mathcal{O}(n^3)$ operations. From the fact that H is an upper triangular matrix and that the first $j - 1$ positions of the vector u_j are zero, it follows that the first $j - 1$ positions of the vector μ_j are also zero. Now, consider the following procedure which runs through all $(0, 1)$ -vectors of length n , and verifies for each vector whether it is a row of A or not. The variable k in the procedure below is defined as $k = \min\{i \mid \alpha_i h_i \neq 0\}$ if $\alpha_i h_i \neq 0$ for some i and $k = n + 1$ otherwise.

```

i := 1; a :=  $e^n$ ;  $\alpha$  :=  $e^n$ ;
if ( $h_n = 0$ ) then k :=  $n + 1$ ; else k :=  $n$ ;
if ( $k \leq n$ ) then fi := 1; else fi := 0; fi;
for i := 2 to  $2^n$ 
begin
  j :=  $n$ ; while ( $j > 0$  and  $a_j = 1$ ) do j :=  $j - 1$ ; od;
  a := a +  $u^j$ ;
   $\alpha$  :=  $\alpha$  +  $\mu^j$ ;
  if ( $k > j$ ) then
    while ( $\alpha_j h_j = 0$  and  $j \leq n$ ) j :=  $j + 1$ ;
    k := j;
  fi;
  if ( $k \leq n$ ) then fi := 1; else fi := 0; fi;
end;

```

By similar arguments as for the earlier procedure in this section one shows that also this procedure has complexity $\mathcal{O}(2^n)$.

5 Empirical results

Based on the ideas we have developed in the previous sections, we wrote a computer program in C-code for computing the prenucleolus of a game, and we ran it on a Pentium-120 machine. In the literature we could find only one other paper that gives some empirical results of the proposed algorithm, namely the paper by Potters, Reijniere and Ansing (1996). Their algorithm

is also described in the Ph.D. thesis by Reijnierse (1995). His thesis contains some more empirical results. We compared our algorithm with these results.

Two types of test-games were chosen by Potters et al. for the performance evaluation of their algorithm. For both types of games the worth of all coalitions is a natural number. The first type of games (N, v) has the following characteristics.

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

The characteristics of the second type are:

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

The results of Potters et al. for the first type of games are reported both in the paper by Potters et al. and in the thesis by Reijnierse. The results for the second type of games are reported only in the thesis by Reijnierse. We ran our algorithm on the same test-bed of games. The results are summarized in tables 1 and 2. For our algorithm the total number of pivots, the number of linear programs and the computation time is an average taken over 100 randomly generated test-games. For the algorithm of Potters, Reijnierse and Ansing this is an average over 10 randomly generated test-games (PRA stands for Potters, Reijnierse and Ansing, and DK stands for Derks and Kuipers). An interesting phenomenon here is that the average number of linear programs is not growing as the number of players increases. The fact that the average lies around 2 indicates that in a high percentage of the cases only 1 linear program is necessary to reach the prenucleolus. This is caused by the fact that small coalitions often determine the prenucleolus for these types of games. Therefore we also did some experiments with randomly generated simple games with the following characteristics: coalitions with less than $n - 2$ players get worth 0, the grand coalition gets worth 1, and coalitions of cardinality $n - 2$ and $n - 1$ get worth 1 with probability 0.9. For this type of games the number of linear programs grows more or less linearly as n increases. Since the computation times are much higher than for the other two types, the results in table 3 are averages over only 10 randomly generated games.

Comparing the two algorithms we see that the computation time of our algorithm is always considerably lower than the one of Potters et al. We are

Table 1: type I games.

$ N $	# pivots		# programs		comp. time	
	PRA	DK	PRA	DK	PRA	DK
3	4.0	3.9	1.4	1.6	0.04	0.001
4	6.8	6.2	2.3	2.1	0.05	0.001
5	9.0	8.3	2.3	2.3	0.06	0.003
6	11.7	9.9	2.2	2.4	0.10	0.006
7	16.6	11.2	2.8	2.4	0.21	0.012
8	20.9	13.1	1.9	2.3	0.56	0.025
9	25.8	14.8	1.9	2.1	1.67	0.053
10	32.9	16.8	1.9	2.3	5.27	0.117
11	-	18.1	-	2.1	-	0.242
12	-	21.2	-	1.9	-	0.556
13	-	23.1	-	1.9	-	1.210
14	-	26.4	-	1.8	-	2.770
15	-	29.8	-	1.7	-	6.261
16	-	31.7	-	1.6	-	13.45
17	-	35.6	-	1.5	-	30.55
18	-	42.6	-	1.5	-	74.09
19	-	47.4	-	1.7	-	168.1
20	-	54.5	-	1.7	-	389.8

Table 2: type II games.

$ N $	# pivots		# programs		comp. time	
	PRA	DK	PRA	DK	PRA	DK
3	3.7	3.6	1.6	1.4	0.05	0.001
4	4.5	6.3	2.2	2.2	0.06	0.002
5	5.5	8.2	2.5	2.2	0.08	0.003
6	7.7	9.6	2.6	2.3	0.10	0.006
7	11.4	11.5	3.5	2.3	0.24	0.012
8	10.9	12.7	2.6	2.4	0.52	0.022
9	18.6	14.1	2.5	2.2	1.62	0.045
10	22.9	16.4	2.7	2.3	5.54	0.097
11	-	19.2	-	2.5	-	0.220
12	-	21.6	-	2.4	-	0.481
13	-	24.7	-	2.4	-	1.091
14	-	26.5	-	2.5	-	2.303
15	-	29.7	-	2.7	-	5.241
16	-	31.5	-	2.5	-	10.93
17	-	36.0	-	2.5	-	25.18
18	-	38.4	-	2.6	-	53.90
19	-	41.8	-	2.7	-	118.2
20	-	44.5	-	2.7	-	252.9

Table 3: type III games.

$ N $	# pivots	# programs	comp. time
3	3.2	1.1	0.000
4	8.6	3.0	0.000
5	8.4	1.7	0.000
6	19.9	3.8	0.011
7	18.7	2.7	0.017
8	39.5	5.0	0.066
9	37.7	3.7	0.110
10	62.3	5.9	0.379
11	66.3	5.0	0.763
12	92.6	7.0	2.121
13	98.0	5.6	4.198
14	124.1	7.7	11.30
15	127.0	6.8	22.34
16	182.3	8.8	66.20
17	172.0	7.8	123.4
18	250.9	9.7	366.6
19	306.6	8.6	833.4
20	261.7	10.9	1550

aware that a good comparison of computation times is not possible here, since the algorithms ran on different machines: we used a Pentium 120 PC, while Potters et al. used a SPARC/SUN/10/41 station. However, a Pentium 120 PC is less than 5 times as fast as a SPARC/SUN station, so we expect that our algorithm is also considerably faster if the two algorithms run on the same type of machine.

Another important observation is that the computation time of our algorithm increases at a slower rate as the number of players increases. Theoretically, we expect the following. The complexity of performing one pivoting step is $\mathcal{O}(2^n)$ for our algorithm, and $\mathcal{O}(n2^n)$ for the algorithm of Potters et al. Hence, if the number of players is doubled from n to $2n$, then the time required for one pivoting operation increases with a factor 2^n for our algorithm, and with a factor 2^{n+1} for the algorithm of Potters et al. Assuming that the number of pivoting operations for both algorithms is comparable, this means that every time the number of players is doubled, our algorithm should gain about a factor 2 on the algorithm by Potters et al.

References

- Bland RG (1977), New finite pivoting rules for the simplex method. *Mathematics of Operations Research*, 2:103–107.
- Dragan I (1981), A procedure for finding the nucleolus of a cooperative n -person game. *Zeitschrift für Operations Research*, 25:119–131.
- Kohlberg E (1972), The nucleolus as solution of a minimization problem. *SIAM Journal of Applied Mathematics*, 23:34–39.
- Kopelowitz A (1967), Computation of the kernel of simple games and the nucleolus of n -person games. Technical Report RM 31, The Hebrew University of Jerusalem.
- Maschler M, Peleg B, and Shapley LS (1979), Geometric properties of the kernel, nucleolus, and related solution concepts. *Mathematics of Operations Research*, 4:303–338.
- Owen G (1974), A note on the nucleolus. *International Journal of Game Theory*, 3:101–103.
- Potters J, Reijnierse J, and Ansing M (1996), Computing the nucleolus by solving a prolonged simplex algorithm. *Mathematical Programming*, 21:757–768.
- Reijnierse J (1995), *Games, Graphs and Algorithms*. Ph.D. Thesis, Nijmegen.

- Sankaran J (1991), On finding the nucleolus of an n -person cooperative game. *International Journal of Game Theory*, 19:329–338.
- Schmeidler D (1969), The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 17:1163–1170.
- Solymosi T (1993), On computing the nucleolus of cooperative games. Ph.D thesis, University of Illinois at Chicago.
- Zumsteg SM (1995), Non-cooperative aspects of cooperative game theory and related computational problems. Ph.D thesis, Eidgenössischen Technische Hochschule Zürich.