An Algorithm to Compute the Nucleolus of Shortest Path Games

Mourad Baïou, Francisco Barahona

Noname manuscript No.
(will be inserted by the editor)

An algorithm to compute the Nucleolus of Shortest Path Games

Mourad Baïou · Francisco Barahona

Received: date / Accepted: date

Abstract We study a type of cooperative games introduced in [9] called shortest path games. They arise on a network that has two special nodes s and t. A coalition corresponds to a set of arcs and it receives a reward if it can connect s and t. A coalition also incurs a cost for each arc that it uses to connect s and t, thus the coalition must choose a path of minimum cost among all the arcs that it controls. These games are relevant to logistics, communication, or supply-chain networks. We give a polynomial combinatorial algorithm to compute the nucleolus. This vector reflects the relative importance of each arc to ensure the connectivity between s and t. Our development is done on a directed graph, but it can be extended to undirected graphs and to similar games defined on the nodes of a graph.

Keywords Cooperative games · Shortest path games · Nucleolus

1 Introduction

Shortest path games arise on a network, a coalition corresponds to a set of arcs and it receives a reward if it can connect two fixed nodes s and t. A coalition also incurs a cost for each arc that it uses to connect s and t, thus the coalition must choose a path of minimum cost among all the arcs that it controls. Shortest path games have been introduced in [9], this type of games is useful to determine the most critical links to ensure connectivity between two distinguished nodes s and t in a network. This analysis is relevant to logistics, communication, or supply-chain networks.

Our contribution is a polynomial combinatorial algorithm to compute the nucleolus of a shortest path game. This vector reflects the relative importance of the

A shorter version of this paper was presented at the SAGT 2017 conference [3].

M. Baïou
CNRS and Université Clermont Auvergne, 1 Rue de la Chebarde, Campus des Cézeaux, 1 Rue de la Chebarde, 63178 Aubière cedex, France.

F. Barahona
IBM T. J. Watson research Center, Yorktown Heights, NY 10589, USA.

different arcs in the network that ensure the connectivity between s and t. Our development is done on a directed graph, but it can be extended to undirected graphs and to similar games defined on the nodes of a graph.

Related work on this type of games is the following. The core and the Shapley Value of shortest path games were studied in [9], also the least core was studied in [2]. Flow games were introduced in [15]. Polynomial combinatorial algorithms for computing the nucleolus of simple flow games were given in [5] and [19]. Algorithms for computing the nucleolus of several other combinatorial games have been found, see [8] for path cooperative games, see [18] for cost allocation games in a tree, [21] for assignment games, [16] for cardinality matching games, [6] for weighted voting games. On the other hand computing the nucleolus is NP-Hard for minimum spanning tree games [7], and general flow games [5].

This paper is organized as follows. In Section 2 we give some basic definitions and mention some basic results of Network Flows and Linear Programming. In Section 3 we study the core. Section 4 contains the basics for the computation of the nucleolus. In Section 5 we give an algorithm to compute the nucleolus when the core is non-empty. In Section 6 we extend the algorithm to the case when the core is empty. In Section 7 we study the time complexity of the algorithm.

2 Preliminaries

Here we give some definitions and mention some basic results on Network Flows and Linear Programming. Throughout this paper we assume that we are working with a directed graph $G = (V, A)$. We use n to denote $|V|$, and m to denote $|A|$. Given two distinguished nodes s and t. An *st-path* is a sequence of arcs $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$, where $s = u_1$, $v_k = t$, and $v_i = u_{i+1}$ for $i = 1, \ldots, k-1$. A *cycle* is a sequence of arcs $(u_1, v_1), (u_2, v_2), \ldots, (u_k, v_k)$, where $u_1 = v_k$, and $v_i = u_{i+1}$ for $i = 1, \ldots, k-1$. Consider a partition of V into U and $V \setminus U$, with $s \in U$ and $t \in V \setminus U$. Then the arc-set $\{(u, v) \,|\, u \in U, v \in V \setminus U\}$ is called an *st-cut*. For a function $w : A \to \mathbb{R}$, and $S \subseteq A$, we use $w(S)$ to denote $w(S) = \sum_{a \in S} w(a)$. For $S \subseteq A$, we use $V(S)$ to denote the set of nodes covered by S, i.e., $V(S) = \cup_{(u,v) \in S} \{u, v\}$.

Shortest path games have been introduced in [9], and they are defined as follows. The graph has two distinguished vertices s and t, and there is a cost function $c : A \to \mathbb{R}_+$, that gives the costs for using different arcs. There is also a reward r obtained if s and t are connected by a path, then for a coalition $S \subseteq A$ the value function is

$$\mathbf{v}(S) = \begin{cases} r - m(S) & \text{if } m(S) < \infty, \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

where $m(S) = \min \big\{ c(P) \,|\, P \subseteq S,\ P \text{ is an } st\text{-path and } c(P) \leq r \big\}$. In the definition given in [9] a player could own more than one arc, here we assume that each player owns exactly one arc.

2.1 Shortest Paths

For a cost function $c : A \to \mathbb{R}$, a shortest st-path is an st-path of minimum cost. The cost of a path is the sum of the costs of the arcs in the path. If the costs are

non-negative, a shortest path can be found in $O(m + n \log n)$ time with Dijkstra's algorithm, see [1].

2.2 Minimum Ratio Cycles

Assume that there is a cost function $c : A \to \mathbb{R}$, and a "time" function $\tau : A \to \mathbb{Z}_+$, then the *cost to time ratio* of a cycle D is $c(D)/\tau(D)$. An algorithm to find a cycle that minimizes the cost to time ratio was given in [13]. In our case we require the function τ to take the values 0 or 1, so the algorithm of [13] takes $O(nm+n^2 \log n)$ time. This will be used in sections 5 and 6.

2.3 Network Flows and Linear Programming

A reference for Minimum Cost Network Flows is [1], Maximum Cost Circulations are also treated there. We just mention that the algorithm of [11] runs in $O(n^2m^3 \log n)$ time. This background is needed in all following sections.

Consider the following network flow problem.

$$\begin{aligned}
&\max \sum_{(u,v)\in A} c(u,v)x(u,v) \\
&\sum_{(u,v)\in A} x(u,v) - \sum_{(v,u)\in A} x(v,u) = 0, \quad \text{for } v \in V, \\
&x(u,v) \geq 0, \quad \text{for all } (u,v) \in A.
\end{aligned}$$

The dual problem is

$$\min \sum_{v\in V} 0 \cdot \pi(v); \quad \pi(v) - \pi(u) \geq c(u,v) \quad \text{for all } (u,v) \in A.$$

Here we have a variable π for each node in V. Notice that there are no capacities for the arcs, so either the optimal value is zero, or the problem is unbounded. The problem is unbounded if and only if there is a cycle of positive cost. Thus the dual problem has a solution if and only if there is no cycle with positive cost. The cost of a cycle is the sum of the costs of its arcs.

In the following sections we use the following basic result on linear programming. Let $g(\epsilon)$ be the value of the linear program $\min\{cx \mid Ax = b + \epsilon d,\ x \geq 0\}$, then g is a convex piecewise linear function of ϵ, cf. [4].

3 The core

Here we study some basic properties of the core. A vector (or a function) $x : A \to \mathbb{R}$ is called an *allocation* if $x(A) = \mathbf{v}(A)$. Here $x(a)$ represents the amount paid to the player a. The core is a concept introduced in [10], it is based on the following stability condition: No subgroup of players does better if they break away from the joint decision of all players to form their own coalition. Thus an allocation x is in the core if $x(S) \geq \mathbf{v}(S)$ for each coalition $S \subseteq A$. Therefore the core is the polytope below.

$$\{x \in \mathbb{R}^A \,|\, x(A) = \mathbf{v}(A),\, x(S) \geq \mathbf{v}(S),\ \text{for } S \subseteq A\}. \tag{2}$$

Let λ be the cost of a shortest st-path in G. If $\lambda \geq r$ then the core consists of only the vector $x = 0$. If $\lambda < r$, the following lemma gives a description of the core that is useful for our purposes.

Lemma 1 *If $\lambda < r$, the core is also defined by*

$$x(A) = r - \lambda, \tag{3}$$
$$x(P) \geq r - c(P) \quad \textit{for each } st\textit{-path } P \textit{ with } c(P) \leq r, \tag{4}$$
$$x \geq 0. \tag{5}$$

Proof Consider an arc a, if a does not go from s to t, we have $x(a) \geq 0$. If a goes from s to t then we have $x(a) \geq \max\{0, r - c(a)\}$. Thus either we have an inequality (4) or an inequality (5).

If P is an st-path with $c(P) \leq r$, then its associated inequality (4) is among the inequalities in (2).

Now assume that $S \subseteq A$ is not an st-path, but it contains an st-path with cost at most r. We have $m(S) = c(\bar{P})$ where $\bar{P} \subseteq S$ is an st-path. Then $x(S) \geq \mathbf{v}(S)$ can be written as $x(S \setminus \bar{P}) + x(\bar{P}) \geq r - c(\bar{P})$. This last inequality is implied by $x(a) \geq 0$ for $a \in S \setminus \bar{P}$, and $x(\bar{P}) \geq r - c(\bar{P})$ that is one of the inequalities (4).

Finally if $S \subseteq A$ does not contain an st-path with cost at most r, then $x(S) \geq 0$ is implied by $x(a) \geq 0$ for $a \in S$. □

The coalitions characterized in Lemma 1 fall into the category of *dually saturated coalitions* introduced in [22]. See also [14], that introduced the notion of *essential coalitions*.

To understand the core, we look at the set of solutions of the linear program below.

$$\min x(A), \quad x(P) \geq r - c(P),\ \text{for each } st\text{-path } P, \quad x \geq 0. \tag{6}$$

Notice that the minimum above is at least $\max\{0, r-\lambda\}$. If it is exactly this value, then the core is nonempty, and it corresponds to the set of optimal solutions of this linear program. If we look at r as a parameter we have that the optimal value is a convex piecewise linear function of r, see Figure 1. Then there is a value $\bar{r}$ such that the core is non-empty if and only if $r \leq \bar{r}$, see Figure 1. The value $\bar{r}$ is discussed in Remark 2 below.

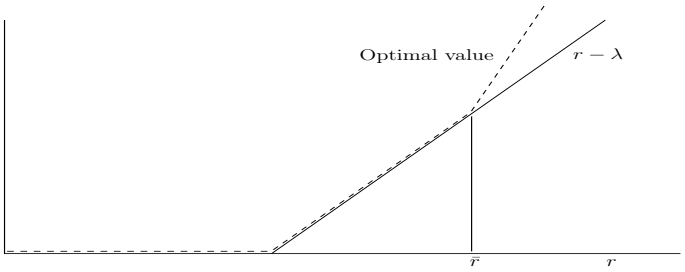

Fig. 1 Optimal value and the value $\bar{r}$. The core is non-empty for $r \leq \bar{r}$.

Consider now the dual of (6), this is

$$\begin{aligned}
&\max \sum_P (r - c(P))y_P \\
&\sum \{y_P \,|\, a \in P\} \leq 1, \quad \text{for each arc } a, \\
&y \geq 0.
\end{aligned}$$

There is a variable y_P for each st-path P, and its associated cost is $r - c(P)$. Each arc has capacity one, and the sum of the variables y for the paths using a particular arc is at most one. We can reduce this to a network flow problem as follows. We add an artificial arc from t to s with cost coefficient r and infinite capacity. To every other arc a we give cost $-c(a)$ and capacity one. Then we look for a circulation of maximum cost.

Recall that λ is the value of a shortest st-path. We need the optimal value to be $r - \lambda$, for this we should have one optimal solution consisting of exactly one shortest path. This leads to the following remark.

Remark 2 *There is a value $\bar{r}$ so that the core is non-empty if and only if $r \leq \bar{r}$. The value $\bar{r}$ is the maximum value of r so that there is an optimal solution consisting of exactly one path.*

Notice that the graph could contain more than one st-path with cost equal to λ. Let A' be the set of arcs that belong to a shortest path, and assume that the core is non-empty. Then the graph should not contain two arc-disjoint shortest paths. Thus the subgraph $G' = (V, A')$ contains an st-cut consisting of exactly one arc. Such an arc is called a *veto-player*, because every shortest path contains it. Let P_1 and P_2 be two shortest st-paths. If x is an element of the core, we have $x(P_1) = x(P_2) = r - \lambda$, and since $x(A) = r - \lambda$, we have $x(P_1 \setminus P_2) = x(P_2 \setminus P_1) = 0$. Thus if A'' is the set of veto-players, we have $x(a) = 0$, for $a \in A \setminus A''$. This had already been established in [9], using different techniques.

4 The Nucleolus

For a coalition S and a vector $x \in \mathbb{R}^A$, their *excess* is $e(x, S) = x(S) - \mathbf{v}(S)$. The nucleolus has been introduced in [20] trying to minimize dissatisfaction of players, more precisely, the nucleolus is the allocation that lexicographically maximizes the vector of non-decreasingly ordered excesses, cf. [20]. Thus in some sense, it is the fairest allocation. The nucleolus can be computed with a sequence of linear programs as follows, cf. [17]. First solve

$$\begin{aligned}
&\max \epsilon \\
&x(S) \geq \mathbf{v}(S) + \epsilon, \quad \forall S \subset A \\
&x(A) = \mathbf{v}(A).
\end{aligned}$$

Let ϵ_1 be the optimal value of this, and $P_1(\epsilon_1)$ be the polytope defined above, with $\epsilon = \epsilon_1$, i.e., $P_1(\epsilon_1)$ is the set of optimal solutions of the linear program above. For a polytope $P \subset \mathbb{R}^A$ let

$$\mathcal{F}(P) = \{S \subseteq A \,|\, x(S) = k_S, \forall x \in P\}. \tag{7}$$

This is the set of coalitions S such $x(S)$ takes a constant value k_S, for all $x \in P$. We called these coalitions *fixed*. In general given ϵ_{r-1} we have to solve

$$\max \epsilon \tag{8}$$
$$x(S) \geq \mathbf{v}(S) + \epsilon, \forall S \notin \mathcal{F}(P_{r-1}(\epsilon_{r-1})) \tag{9}$$
$$x \in P_{r-1}(\epsilon_{r-1}). \tag{10}$$

We denote by ϵ_r the optimal value of this, and $P_r(\epsilon_r)$ the polytope above with $\epsilon = \epsilon_r$. We continue for $r = 2, ..., |A|$, or until $P_r(\epsilon_r)$ is a singleton. Notice that each time the dimension of $P_r(\epsilon_r)$ decreases by at least one, so it takes at most $|A|$ steps for $P_r(\epsilon_r)$ to be a singleton.

In general the difficulty in computing the nucleolus resides in having to solve a sequence of linear programs with an exponential number of inequalities. In our case we shall see that most of the inequalities are redundant, and only a polynomial number of them is needed. Moreover these linear programs can be solved in a combinatorial way.

5 The nucleolus when the core is non-empty

In this section we study the nucleolus under the assumption that the core is non-empty. In that case the dual of (6) should have an optimal solution consisting of exactly one path. Thus if there are several shortest st-paths, no two of them should be arc-disjoint. The only arcs a for which $x(a)$ can take a non-zero value, are the arcs that belong to the intersection of all shortest paths. To compute the nucleolus, we have to solve the sequence of linear programs described in Section 4, for that we need some changes of variables as follows.

5.1 An alternative description of the core

For x in the core, define $z = x + c$, where c is the vectors of costs. We fix one shortest st-path $\mathcal{P}$, then the following should be satisfied: $z(a) = c(a)$, if $a \notin \mathcal{P}$; $z(a) \geq c(a)$, if $a \in \mathcal{P}$; $z(\mathcal{P}) = r$; $z(P) \geq r$, if P is an st-path; $z(P) = c(P) \geq r$, if P is an st-path not containing arcs in $\mathcal{P}$.

Now we use a change of variables similar to the one used in [19]. For two nodes a and b in $\mathcal{P}$ we denote by $\mathcal{P}_{ab}$ the sub-path of $\mathcal{P}$ going from a to b. Given z as defined above, for each node u in $\mathcal{P}$, let $p_z(u) = z(\mathcal{P}_{su})$. Therefore we have

$$p_z(s) = 0, \quad p_z(t) = r, \tag{11}$$
$$p_z(v) - p_z(u) \geq c(u, v), \text{ for } (u, v) \in \mathcal{P}. \tag{12}$$

For two nodes u and v in $\mathcal{P}$, a *jump* J_{uv} is a path from u to v, such that all nodes in J_{uv} different from u and v are not in $\mathcal{P}$, this notion was used in [19]. If there is an arc $a = (u, v)$ with $a \notin \mathcal{P}$, then we have a jump consisting of one arc. Consider an st-path P consisting of the sub-path $\mathcal{P}_{su}$ from s to u, then a jump J_{uv} from u to v, and a sub-path $\mathcal{P}_{vt}$ from v to t. Then the inequality $z(P) \geq r$ can be written as $p_z(u) + c(J_{uv}) + r - p_z(v) \geq r$, or

$$p_z(u) - p_z(v) \geq -c(J_{uv}). \tag{13}$$

Now we show that (11), (12) and (13) are sufficient to describe the core. We need the lemma below.

Lemma 3 *Inequalities associated with paths with more than one jump are implied by inequalities associated with paths with one jump.*

Proof Consider an st-path P with two jumps. Thus assume that P consists of the following segments: a sub-path $\mathcal{P}_{sa}$ from s to a, a jump J_{ab} from a to b, a sub-path $\mathcal{P}_{bd}$ from b to d, a jump J_{de} from d to e, and a sub-path of $\mathcal{P}_{et}$ from e to t. Then the inequality $z(P) \geq r$, can be written as $z(P) = p_z(a) + c(J_{ab}) + p_z(d) - p_z(b) + c(J_{de}) + r - p_z(e) \geq r$, or $p_z(a) - p_z(b) + c(J_{ab}) + p_z(d) - p_z(e) + c(J_{de}) \geq 0$. This is implied by $p_z(a) - p_z(b) + c(J_{ab}) \geq 0$ and $p_z(d) - p_z(e) + c(J_{de}) \geq 0$. These two inequalities correspond to st-paths with one jump. Paths with more than two jumps can be treated in a similar way. □

On the other other hand, let $\bar{V} = V(\mathcal{P})$, the set of nodes spanned by $\mathcal{P}$. Consider any function $p : \bar{V} \to \mathbb{R}$ satisfying (11), (12) and (13), we can define $\bar{z}(u, v) = p(v) - p(u)$ for each arc $(u, v) \in \mathcal{P}$, $\bar{z}(u, v) = c(u, v)$ for each arc $(u, v) \in A \setminus \mathcal{P}$, and $\bar{x} = \bar{z} - c$. It is easy to see that $\bar{x}$ is an element of the core. Thus there is a bijection between the vectors in the core and the functions $p : \bar{V} \to \mathbb{R}$ satisfying (11), (12) and (13).

Paths with one jump correspond to *dually essential coalitions* in the terminology of [22]. This structure resembles to "trunks with one missing subbranch" used in [23]. At this point we have a polynomial number of inequalities that define the core. Based on that, it has been shown in [12] that the computation of the nucleolus when the core is nonempty, reduces to a sequence of combinatorial linear programs, that can be solved in strongly polynomial time with the algorithm of [24]. In our case, we show that the computation of the nucleolus reduces to a sequence of minimum "cost to time ratio cycle" problems that can be solved with the algorithm of [13]. This is the subject of the next sub-section.

5.2 The nucleolus

To compute the nucleolus we have to solve the sequence of linear programs defined in Section 4. The development above suggests the following procedure.

We create an auxiliary graph $G' = (\bar{V}, A')$ as follows. First we include in A' each arc $(u, v) \in \mathcal{P}$ with cost $d(u, v) = c(u, v)$. Then for every pair of nodes u and v in $\bar{V}$, we find the cost of a shortest path in G from u to v using only arcs not in $\mathcal{P}$ and going only through nodes in $V \setminus \bar{V}$. Let ρ be the value of this shortest path, we add an arc (v, u) to A' with cost $d(v, u) = -\rho$. Finally we add the arcs (s, t) and (t, s) to A', with costs $d(s, t) = r$ and $d(t, s) = -r$. Then we impose the inequalities

$$p(v) - p(u) \geq d(u, v) \quad \text{for all } (u, v) \in A'. \tag{14}$$

These inequalities come from the following conditions.

- For an arc $(u, v) \in \mathcal{P}$ they correspond to $x(u, v) \geq 0$.
- For an arc $(u, v) \notin \mathcal{P}$, $(u, v) \neq (s, t), (t, s)$ they correspond to $x(P) \geq r - c(P)$, where P is the path consisting of $\mathcal{P}_{sv}$, a jump J_{vu}, and $\mathcal{P}_{ut}$.
- The inequalities for (s, t) and (t, s) imply $x(\mathcal{P}) + c(\mathcal{P}) = p(t) - p(s) = r$.

Thus from a vector p satisfying (14) we can derive a vector in the core. Moreover, as mentioned in Subsection 2.3, the system (14) has a solution if and only if the graph G' has no cycle of positive weight.

Now we can describe the computation of the nucleolus. We set $k = 0$, $\bar{\epsilon} = 0$, and we call the arcs (s,t) and (t,s) *fixed*. We have to solve

$$\max \mu \tag{15}$$
$$p(v) - p(u) \geq d(u,v) \quad \text{if } (u,v) \in A' \text{ is fixed,} \tag{16}$$
$$p(v) - p(u) \geq d(u,v) + \mu \quad \text{if } (u,v) \in A' \text{ is not fixed,} \tag{17}$$
$$\mu \geq 0. \tag{18}$$

Inequalities (17) come from $x(u,v) \geq \mu$ for $(u,v) \in \mathcal{P}$, or $x(P) \geq r - c(P) + \mu$ if P is an st-path with one jump. To prove that these inequalities are sufficient we need the following lemma, its proof is similar to the one of Lemma 1.

Lemma 4 *The inequalities $x(S) \geq \mathbf{v}(S) + \epsilon$, for $S \subseteq A$, are implied by*

$$x(P) \geq r - c(P) + \epsilon, \quad \textit{for each } st\textit{-path } P;\ x(a) \geq \epsilon;\ \epsilon \geq 0.$$

As seen in sub-section 2.3, when solving (15)-(18) we are looking for the maximum value of μ so that when we increase the costs of the non-fixed arcs by this amount, the graph has no positive cycle. Thus we need $d(C) + \mu\, n(C) \leq 0$, for each cycle C, where $n(C)$ is the number of non-fixed arcs in the cycle C. Then we have to compute

$$\bar{\mu} = \min_C \frac{-d(C)}{n(C)}, \tag{19}$$

where the minimum is taken over all cycles in G'. Here we are looking for a cycle that minimizes a "cost to time" ratio. As mentioned in Subsection 2.2, this can be solved with the algorithm of [13]. Once the value $\bar{\mu}$ is obtained in (19), we update the arcs costs as $d(u,v) \leftarrow d(u,v) + \bar{\mu}$, for each non-fixed arc $(u,v) \in A'$. Let $\bar{C}$ be a cycle giving the minimum in (19), we declare *fixed* all arcs in $\bar{C}$. We update $\bar{\epsilon} \leftarrow \bar{\epsilon} + \bar{\mu}$, $k \leftarrow k+1$, and set $\epsilon_k = \bar{\epsilon}$. As long as there is an arc in A' that is not fixed we solve (15)-(18) and continue. Since at each iteration at least one new arc in A' becomes fixed, this procedure takes at most $|A'|$ iterations.

6 The nucleolus when the core is empty

Here we assume that the core is empty, thus for the first linear program defined in Section 4, we have $\epsilon_1 < 0$. First we have to prove that its set of optimal solutions $P_1(\epsilon_1)$, is in the non-negative orthant.

Lemma 5 $P_1(\epsilon_1) \subset \mathbb{R}^A_+$.

Proof Assume that $(\bar{x}, \epsilon_1)$ is a solution of

$$\begin{aligned} &\max \epsilon \\ &x(A) = \mathbf{v}(A) \\ &x(S) \geq \mathbf{v}(S) + \epsilon, \text{ for } S \subset A, \end{aligned}$$

and $\bar{x}(a_0) < 0$ for some arc $a_0 \in A$.

First we prove that if $\bar{x}(S) = \mathbf{v}(S) + \epsilon_1$, then $a_0 \in S$. Suppose $a_0 \notin S$. We have two cases:

- $S \cup \{a_0\} \neq A$. Then $\bar{x}(S \cup \{a_0\}) < \bar{x}(S) = \mathbf{v}(S) + \epsilon_1 \leq \mathbf{v}(S \cup \{a_0\}) + \epsilon_1$. This contradicts the feasibility of $(\bar{x}, \epsilon_1)$.
- $S \cup \{a_0\} = A$. Then $\bar{x}(A) = \bar{x}(S) + \bar{x}(a_0) = \mathbf{v}(S) + \epsilon_1 + \bar{x}(a_0) < \mathbf{v}(S) \leq \mathbf{v}(A)$. Again this contradicts the feasibility of $(\bar{x}, \epsilon_1)$.

Then we can define $x'(a_0) = \bar{x}(a_0) + \beta$, and $x'(a) = \bar{x}(a) - \beta/(m-1)$, for $a \in A \setminus \{a_0\}$, for a small number $\beta > 0$, $m = |A|$. Since x' is a better solution we have a contradiction. □

As in the previous section, we have to see that when computing the nucleolus most inequalities are redundant. This is in the lemma below, its proof is similar to the one of Lemma 1.

Lemma 6 *Inequalities $x(S) \geq \mathbf{v}(S) + \epsilon$, for $S \subset A$, are implied by $x(P) \geq r - c(P) + \epsilon$, for each st-path P, and $x \geq 0$.*

We give an algorithm for the first linear program in the sub-section below.

6.1 Computation of ϵ_1

Here we deal with the first linear program. We are going to use the structure of the solutions of this to derive a change of variables similar to the one in Section 5.

To compute ϵ_1 we do not treat the first linear program directly. Instead we use parametric linear programming and look for the maximum value of the parameter $\epsilon < 0$, so that the value of the parametric linear program below is $r - \lambda$.

$$\min x(A) \tag{20}$$

$$x(P) \geq r + \epsilon - c(P), \text{ for each } st\text{-path } P, \tag{21}$$

$$x \geq 0. \tag{22}$$

Lemma 6 justifies the use of inequalities (21). Lemma 5 justifies inequalities (22). The maximum value of the parameter ϵ is exactly the value ϵ_1 that we need.

The dual of (20)-(22) is

$$\max \sum_P (r + \epsilon - c(P)) y_P \tag{23}$$

$$\sum \{y_P \mid a \in P\} \leq 1, \text{ for each arc } a, \tag{24}$$

$$y \geq 0. \tag{25}$$

Here we have a variable y_P for each st-path P. For each arc a we have a constraint (24). It says that the sum of the variables y_P for all paths P that contain the arc a, should be at most 1.

Notice that there might be an exponential number of st-paths, so (23)-(25) might have an exponential number of variables. However we reduce this to a network flow problem as follows. We add an artificial arc from t to s with cost coefficient $r + \epsilon$ and infinite capacity. To every other arc a we give cost $-c(a)$ and

capacity one. Then we look for a circulation of maximum cost. If a circulation has positive flow in the arc from t to s, then there is at least one path from s to t whose arcs have positive flow. Since each original arc has capacity one, then there is an optimal circulation that corresponds to a set of arc-disjoint st-paths of minimum cost.

Recall that $m = |A|$. For a non-negative integer k, $0 \leq k \leq m$, let $f(k)$ be the value of a minimum cost set of k arc-disjoint st-paths. This is an optimal solution of a minimum cost flow problem sending k units of flow from s to t, and with capacities equal to one on every arc. Denote by $g(\epsilon)$ the optimal value of (23)-(25), then

$$g(\epsilon) = \max_k \{k(r+\epsilon) - f(k)\}. \tag{26}$$

Here the maximum is taken over all possible values of k so that G has k arc-disjoint st-paths. Thus the function g is the maximum of a set of linear functions, see Figure 2. This is a convex piece-wise linear function. One evaluation of $g(\cdot)$ is done by solving a network flow problem. In what follows we discuss how to find the value ϵ_1 so that $g(\epsilon_1) = r - \lambda$. This is done with the algorithm below that is an adaptation of Newton's method for finding a root of a real-valued function. It relies on the convexity of g.

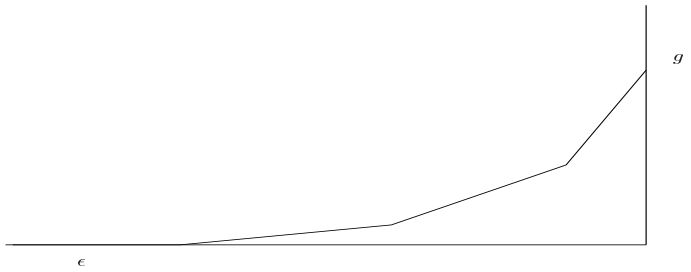

Fig. 2 Function g.

Step 0. We set $\epsilon^- = -r$ and $\epsilon^+ = 0$, thus $g(\epsilon^-) = 0 < r - \lambda$ and $g(\epsilon^+) > r - \lambda$. We denote by $k(\epsilon)$ a value of k giving the maximum in (26).

Step 1. Let $k^- = k(\epsilon^-)$, $k^+ = k(\epsilon^+)$. If $k^- = k^+$, then ϵ_1 is the solution of $g(\epsilon^-) + k^-(\epsilon - \epsilon^-) = r - \lambda$, and we Stop. Otherwise let $\tilde{\epsilon}$ be the solution of $g(\epsilon^-) + k^-(\epsilon - \epsilon^-) = g(\epsilon^+) + k^+(\epsilon - \epsilon^+)$. See Figure 3.

Step 2. If $g(\tilde{\epsilon}) < r - \lambda$ set $\epsilon^- = \tilde{\epsilon}$. Otherwise set $\epsilon^+ = \tilde{\epsilon}$ and go to Step 1.

Notice that at each iteration, either k^- increases or k^+ decreases, and since $0 \leq k \leq m$, this algorithm takes at most $2m$ iterations.

6.2 A change of variables

Now we assume that the value ϵ_1 has been found. Based on the structure of the solutions in the previous sub-section, we can derive a change of variables similar to the one in Section 5. This is below. There is an optimal solution of (23)-(25) that corresponds to a set of k arc-disjoint st-paths $\mathcal{S} = \{\mathcal{P}^1, \ldots, \mathcal{P}^k\}$. We set

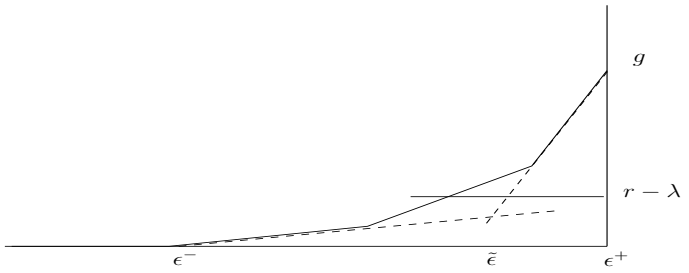

Fig. 3 Finding $\tilde{\epsilon}$.

$\mathcal{P} = \cup_i \mathcal{P}^i$. Let $\bar{x}$ be a solution of (20)-(22), then the complementary slackness conditions imply $\bar{x}(a) = 0$ if $a \in A \setminus \mathcal{P}$, and $\bar{x}(\mathcal{P}^i) = r + \epsilon_1 - c(\mathcal{P}^i)$, $i = 1, \ldots, k$.

Let $\bar{z}(a) = \bar{x}(a) + c(a)$, for each $a \in A$, then $\bar{z}(a) \geq c(a)$, for $a \in \mathcal{P}$; $\bar{z}(a) = c(a)$, if $a \in A \setminus \mathcal{P}$; $\bar{z}(\mathcal{P}^i) = r + \epsilon_1$, $i = 1, \ldots, k$; $\bar{z}(P) \geq r + \epsilon_1$, for every st-path P. Let $\bar{V} = V(\mathcal{P})$. Before the next change of variables, we need the lemma below.

Lemma 7 *Let $v \in \bar{V}$, and assume that there are two paths $\mathcal{P}^i$ and $\mathcal{P}^j$ going through v. Then $\bar{z}(\mathcal{P}^i_{sv}) = \bar{z}(\mathcal{P}^j_{sv})$.*

Proof We have $\bar{z}(\mathcal{P}^i_{sv}) + \bar{z}(\mathcal{P}^i_{vt}) = \bar{z}(\mathcal{P}^j_{sv}) + \bar{z}(\mathcal{P}^j_{vt}) = r + \epsilon_1$. If $\bar{z}(\mathcal{P}^i_{sv}) < \bar{z}(\mathcal{P}^j_{sv})$, then $\bar{z}(\mathcal{P}^i_{sv}) + \bar{z}(\mathcal{P}^j_{vt}) < r + \epsilon_1$. This leads to a contradiction because for the st-path $\mathcal{P}^i_{sv} \cup \mathcal{P}^j_{vt}$ we should have $\bar{z}(\mathcal{P}^i_{sv}) + \bar{z}(\mathcal{P}^j_{vt}) \geq r + \epsilon_1$. □

Based on Lemma 7, for any $u \in \bar{V}$ we can define $p_z(u) = z(\mathcal{P}^i_{su})$, where z is any vector in $P_1(\epsilon_1)$ and $\mathcal{P}^i$ is any path in $\mathcal{S}$ going through u. Recall that $P_i(\epsilon_i) \subset P_1(\epsilon_1)$ for $i > 1$. We have

$$p_z(s) = 0, \quad p_z(t) = r + \epsilon_1, \tag{27}$$

$$p_z(v) - p_z(u) \geq c(u, v), \text{ for } (u, v) \in \mathcal{P}. \tag{28}$$

Now we use the same notion of a jump used in the last section. Consider an st-path P consisting of the sub-path $\mathcal{P}^i_{su}$ from s to u, then a jump J_{uv} from u to v, and a sub-path $\mathcal{P}^j_{vt}$ from v to t. Then the inequality $z(P) \geq r + \epsilon_1$ can be written as $p_z(u) + c(J_{uv}) + r + \epsilon_1 - p_z(v) \geq r + \epsilon_1$, or $p_z(u) - p_z(v) \geq -c(J_{uv})$.

Below we have the analogue of Lemma 3, its proof is similar.

Lemma 8 *Inequalities associated with paths with more than one jump are implied by inequalities associated with paths with one jump.*

As in Section 5, we create an auxiliary graph $G' = (\bar{V}, A')$ as follows. First we include in A' each arc $(u, v) \in \mathcal{P}$ with cost $d(u, v) = c(u, v)$. Then for every pair of nodes u and v in $\bar{V}$, we find the cost of a shortest path in G from u to v using arcs not in $\mathcal{P}$ and going only through nodes in $V \setminus \bar{V}$. Let ρ be the value of this shortest path, we add an arc (v, u) to A' with cost $d(v, u) = -\rho$. Finally we add the arcs (s, t) and (t, s) to A', with costs $d(s, t) = r + \epsilon_1$ and $d(t, s) = -r - \epsilon_1$. Then we impose the inequalities

$$p(v) - p(u) \geq d(u, v) \quad \text{for all } (u, v) \in A'. \tag{29}$$

For a vector p satisfying (29) we can define $z(u,v) = p(v) - p(u)$ for $(u,v) \in \mathcal{P}$, and $z(u,v) = c(u,v)$ for $(u,v) \in A \setminus \mathcal{P}$. Then $x = z - c$ satisfies $x(A) = r - \lambda$ and the inequalities (21)-(22). As before, there is a bijection between vectors x satisfying $x(A) = r - \lambda$ and the inequalities (21)-(22), and vectors p satisfying $p(s) = 0$ and (29). Now we can proceed to the computation of the nucleolus.

6.3 computation of the nucleolus

Recall that $\epsilon_1 < 0$. As seen in Section 4, we have to solve a sequence of linear programs where we maximize a parameter ϵ. We divide this into two phases. The case when $\epsilon \leq 0$ is treated first, and then we continue with the case when $\epsilon > 0$.

6.3.1 The case when $\epsilon \leq 0$.

We set $\bar{\epsilon} = \epsilon_1$, $k = 0$, we call *fixed* the arcs (s,t) and (t,s), and we have to solve

$$\max \mu \tag{30}$$
$$p(v) - p(u) \geq d(u,v) \quad \text{if } (u,v) \in A' \text{ is fixed or } (u,v) \in \mathcal{P} \tag{31}$$
$$p(v) - p(u) \geq d(u,v) + \mu \quad \text{if } (u,v) \in A' \setminus \mathcal{P}, \text{ and } (u,v) \text{ is not fixed,} \tag{32}$$
$$0 \leq \mu \leq -\bar{\epsilon}. \tag{33}$$

After solving this, the new value for ϵ will be $\bar{\epsilon} + \mu$. For $(u,v) \in \mathcal{P}$ inequalities (31) correspond to $x(u,v) \geq 0$. Inequalities (32) correspond to $x(P) \geq r - c(P) + \bar{\epsilon} + \mu$, if P is an st-path with one jump. We need the inequality $\mu \leq -\bar{\epsilon}$ in (33) because the new value of ϵ should be non-positive.

When solving (30)-(33) we are looking for the maximum value of μ so that the graph has no positive cycle, if we increase the costs of the non-fixed arcs in $A' \setminus \mathcal{P}$ by this amount. Thus we need $d(C) + \mu\, n(C) \leq 0$ for each cycle C. Here $n(C)$ is the number of arcs in $A' \setminus \mathcal{P}$ that are non-fixed, in the cycle C. Then we have to compute

$$\alpha = \min_C \frac{-d(C)}{n(C)}. \tag{34}$$

Here the minimum is taken over all cycles in G'. As before this can be found with the algorithm of [13].

Once the value α is obtained in (34), we set $\mu = \min\{\alpha, -\bar{\epsilon}\}$. Then we update the arcs costs as

$$d(u,v) \leftarrow d(u,v) + \mu,$$

for each non-fixed arc $(u,v) \in A' \setminus \mathcal{P}$. If $\mu < -\bar{\epsilon}$, let $\bar{C}$ be a cycle giving the minimum in (34). We declare *fixed* all arcs in $\bar{C} \cap (A' \setminus \mathcal{P})$. We also update $\bar{\epsilon} \leftarrow \bar{\epsilon} + \mu$, $k \leftarrow k + 1$, $\epsilon_k = \bar{\epsilon}$. Notice that at the first iteration we obtain $\mu = 0$, because at this point $\bar{\epsilon} = \epsilon_1$. So this iteration just gives a set of arcs that should be fixed.

If $\bar{\epsilon} < 0$ and there is an arc in $A' \setminus \mathcal{P}$ that is not fixed we solve (30)-(33) and continue. Otherwise $\bar{\epsilon} = 0$, or all arcs in $A' \setminus \mathcal{P}$ are fixed, in this case we should have $\epsilon \geq 0$, this is treated below.

6.3.2 The case when $\epsilon \geq 0$.

Here the arcs that have been fixed remain fixed. We have to impose $x(a) \geq \epsilon$ for $a \in \mathcal{P}$, this corresponds to inequalities (37) for $a \in \mathcal{P}$. Thus we have to solve

$$\max \mu \tag{35}$$
$$p(v) - p(u) \geq d(u,v) \quad \text{if } (u,v) \in A' \text{ is fixed,} \tag{36}$$
$$p(v) - p(u) \geq d(u,v) + \mu \quad \text{if } (u,v) \in A' \text{ is not fixed,} \tag{37}$$
$$\mu \geq 0. \tag{38}$$

Then we proceed as in Sub-section 5.2.

7 Complexity

Now we discuss the time complexity of all the procedures above.

In Section 3 we need the value of a shortest path, this takes $O(m+n\log n)$ time. Then to decide whether the core is empty we look for a circulation of maximum cost, this takes $O(n^2m^3\log n)$ time.

If the core is non-empty, we use the procedure in Sub-section 5.2. At each iteration we have to find a cycle that minimizes a cost to time ratio. This done with the algorithm of [13] that takes $O(nm+n^2\log n)$ time. Each time at least one arc becomes fixed, therefore this is done at most m times. Thus the complexity of this procedure is $O(nm^2+n^2\log n)$.

If the core is empty, we use the procedure in Section 6. To compute ϵ_1 we need to solve at most $2m$ maximum cost circulation problems. So the complexity of this is $O(n^2m^4\log n)$. Then we have to find at most m cycles that minimize the cost to time ratio. Therefore the complexity of this part is $O(nm^2+n^2\log n)$.

Thus the time complexity is dominated by the computation of ϵ_1 in Sub-section 6.1. Then we have the theorem below.

Theorem 9 *The nucleolus of a shortest path game can be computed in $O(n^2m^4\log n)$ time.*

Acknowledgements We are grateful to an anonymous referee for his careful reading. His comments helped us to improve the presentation.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network flows: theory, algorithms, and applications. Prentice hall (1993)
2. Aziz, H., Sørensen, T.B.: Path coalitional games. arXiv preprint arXiv:1103.3310 (2011)
3. Baïou, M., Barahona, F.: On the nucleolus of shortest path games. In: V. Bilò, M. Flammini (eds.) Algorithmic Game Theory, pp. 55–66. Springer International Publishing, Cham (2017)
4. Chvatal, V.: Linear programming. Macmillan (1983)
5. Deng, X., Fang, Q., Sun, X.: Finding nucleolus of flow game. Journal of combinatorial optimization **18**(1), 64–86 (2009)
6. Elkind, E., Pasechnik, D.: Computing the nucleolus of weighted voting games. In: Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 327–335. Society for Industrial and Applied Mathematics (2009)

7. Faigle, U., Kern, W., Kuipers, J.: Note computing the nucleolus of min-cost spanning tree games is np-hard. International Journal of Game Theory **27**(3), 443–450 (1998)
8. Fang, Q., Li, B., Shan, X., Sun, X.: The least-core and nucleolus of path cooperative games. In: International Computing and Combinatorics Conference, pp. 70–82. Springer (2015)
9. Fragnelli, V., Garcia-Jurado, I., Mendez-Naya, L.: On shortest path games. Mathematical Methods of Operations Research **52**(2), 251–264 (2000)
10. Gillies, D.B.: Solutions to general non-zero-sum games. Contributions to the Theory of Games **4**(40), 47–85 (1959)
11. Goldberg, A.V., Tarjan, R.E.: Finding minimum-cost circulations by canceling negative cycles. Journal of the ACM (JACM) **36**(4), 873–886 (1989)
12. Granot, D., Granot, F., Zhu, W.R.: Characterization sets for the nucleolus. International Journal of Game Theory **27**(3), 359–374 (1998). DOI 10.1007/s001820050078. URL https://doi.org/10.1007/s001820050078
13. Hartmann, M., Orlin, J.B.: Finding minimum cost to time ratio cycles with small integral transit times. Networks **23**(6), 567–574 (1993)
14. Huberman, G.: The nucleolus and the essential coalitions. In: A. Bensoussan, J.L. Lions (eds.) Analysis and Optimization of Systems, pp. 416–422. Springer Berlin Heidelberg, Berlin, Heidelberg (1980)
15. Kalai, E., Zemel, E.: Generalized network problems yielding totally balanced games. Operations Research **30**(5), 998–1008 (1982)
16. Kern, W., Paulusma, D.: Matching games: the least core and the nucleolus. Mathematics of operations research **28**(2), 294–308 (2003)
17. Kopelowitz, A.: Computation of the kernels of simple games and the nucleolus of n-person games. Tech. rep., DTIC Document (1967)
18. Megiddo, N.: Computational complexity of the game theory approach to cost allocation for a tree. Mathematics of Operations Research **3**(3), 189–196 (1978)
19. Potters, J., Reijnierse, H., Biswas, A.: The nucleolus of balanced simple flow networks. Games and Economic Behavior **54**(1), 205–225 (2006)
20. Schmeidler, D.: The nucleolus of a characteristic function game. SIAM Journal on applied mathematics **17**(6), 1163–1170 (1969)
21. Solymosi, T., Raghavan, T.E.: An algorithm for finding the nucleolus of assignment games. International Journal of Game Theory **23**(2), 119–143 (1994)
22. Solymosi, T., Sziklai, B.: Characterization sets for the nucleolus in balanced games. Operations Research Letters **44**(4), 520–524 (2016)
23. Sziklai, B., Fleiner, T., Solymosi, T.: On the core and nucleolus of directed acyclic graph games. Mathematical Programming **163**(1), 243–271 (2017). DOI 10.1007/s10107-016-1062-y. URL https://doi.org/10.1007/s10107-016-1062-y
24. Éva Tardos: A strongly polynomial algorithm to solve combinatorial linear programs. Operations Research **34**(2), 250–256 (1986). URL http://www.jstor.org/stable/170819