Int J Game Theory (2010) 39:89–104
DOI 10.1007/s00182-009-0176-3

# The nucleolus of a standard tree game revisited: a study of its monotonicity and computational properties

**Michael Maschler · Jos Potters · Hans Reijnierse**

Accepted: 25 August 2009 / Published online: 10 September 2009

**Abstract** This paper introduces yet another algorithm to compute the nucleolus of a standard tree game. One advantage of this algorithm is that it provides a very intuitive interpretation of the nucleolus, under which the players participate in a joint enterprize in which each group sends a member to help the community. Another advantage is that it demonstrates monotonicity properties of the nucleolus within this class of games. As a consequence the nucleolus of a tree game can be extended to a population monotonic allocation scheme.

**Keywords** Standard tree games · Nucleolus · Population monotonic allocation scheme

**Mathematics Subject Classification (2000)** 90D30

## 1 Introduction

The computation of a solution concept for a general $n$-person cooperative game is not an easy task. One reason is the fact that one has to enter $\mathcal{O}(2^n)$ pieces of data to

This article is dedicated to the memory of Michael. This paper is a result of a longstanding working together. We had long breaks and short, but intensive periods of interaction, the final one only three weeks before his passing away.

M. Maschler
c/o H. Reijnierse,
P.O. Box 90153, 5000 LE Tilburg, The Netherlands

J. Potters
Radboud University Nijmegen, Nijmegen, The Netherlands

H. Reijnierse (✉)
Center and Department of Economics and OR, Tilburg University,
P.O. Box 90153, 5000 LE Tilburg, The Netherlands
e-mail: J.H.Reijnierse@uvt.nl

the computer, i.e., the worths of the various coalitions, before even starting to apply computation protocols. And even if one finds a solution to a particular game, say, the nucleolus, one obtains a vector of numbers and the question is: what does it signify aside from the fact that some game theorists happen to like it?

For these reasons, a lot of research has been done on *families of games* in the hope that one can use properties of such families to shorten the computation significantly and also be able to capture common properties of the solutions for each particular family. This may provide insight into the nature of the solution and may make it attractive to apply the solution to particular real cases.

One such family is the family of *standard tree games*. Its nucleolus (Schmeidler 1969) was first studied by Megiddo (1978), who produced an algorithm to compute it in $\mathcal{O}(q^3)$ steps, where $q$ is the number of edges of the tree. His method was modified by Galil (1980) who shortened it to $\mathcal{O}(q \log q)$ steps. Granot et al. (1996) produced another algorithm whose complexity was $\mathcal{O}(q)$ for subclasses of trees which includes all binary trees on the one hand and the airport games (when the tree is a chain), on the other hand.

In this paper we produce another algorithm which has in our opinion a natural interpretation. It can be interpreted as a procedure followed by a group of agents to accomplish a task. They work in such a way that various subgroups send one of their members ahead, helping the general community. This insight, aside from being interesting in itself, encourages studying and employing other related solution concepts, in which, say, helping the community takes different directions and still yields an outcome in the core of the game.

The nucleolus is usually not too good at following monotonicity rules. It may happen, for example, that by raising the worth of a single coalition some members of that coalition get less (have to pay more in a cost game). This drawback is a consequence of being a core-related solution. (See Megiddo (1974) and a discussion of this issue in Maschler (1992).) It therefore came somewhat as a surprise when Sönmez (1994) proved that the nucleolus of the airport game is a population monotonic allocation scheme—a concept due to Thomson (1983a,b) for bargaining games and to Sprumont (1990) for general $n$-person TU games. By this, one means that if some of the players leave the cost game, none of the remaining players has to pay less in the nucleolus. This paper extends Sönmez's result to any standard tree game, using the insight gained from the interpretation of the nucleolus.

The paper is organized as follows: Sect. 2 defines the standard tree game and shows that other tree games can be reduced to standard ones without changing the coalition function. Section 3 briefly describes the algorithms mentioned above, introduces our own and compares them with each other. Section 4 interprets the new algorithm in an intuitive way. This interpretation is used in Sect. 5 to prove the two monotonicity properties.

## 2 Definitions and notations

A *tree network* $\Gamma$ is given by the following:

(i) $(V, E)$ is a finite tree, i.e., a connected graph without cycles. $V$ is the set of nodes and $E$ is the set of edges. The number of edges is denoted by $q$.

(ii) One node $\mathbf{o} \in V$ has a special meaning and is called the *root* of the tree.
(iii) There is a cost function $a : E \to \Re$ on the edges of the tree.
(iv) There is a cost function $b : V \to \Re$ on the nodes of the tree.
(v) For each node $\mathbf{p} \in V$ there is a (possibly empty) player set $N_\mathbf{p}$, whose members are the *residents* of $\mathbf{p}$. The cardinality of $N_\mathbf{p}$ is denoted by $n_\mathbf{p}$. The union of all players is called $N$.

We denote nodes by boldface characters like $\mathbf{p}$ and $\mathbf{q}$, players are denoted by numbers or italic characters like $i$ and $j$. For every node $\mathbf{p}$ in $V \setminus \{\mathbf{o}\}$ there is a unique path from the root $\mathbf{o}$ to $\mathbf{p}$. The node on this path adjacent to $\mathbf{p}$ is called the *parent* $\pi(\mathbf{p})$ of $\mathbf{p}$. The edge $(\pi(\mathbf{p}), \mathbf{p})$ is denoted by $e_\mathbf{p}$ and its costs by $a_\mathbf{p}$. Furthermore, we write $\mathbf{p} \preceq \mathbf{q}$ if the path from $\mathbf{o}$ to $\mathbf{q}$ contains $\mathbf{p}$. For a node $\mathbf{p} \in V$ we denote by $d_\mathbf{p}$ the number of nodes of which it is the parent. If $d_\mathbf{p} = 0$, $\mathbf{p}$ is called a *leaf*. A *trunk* $T = (V(T), E(T))$ is a subgraph of $(V, E)$, spanned by a set of nodes $V(T) \subseteq V$ that is closed under the precedence relation $\preceq$, i.e., if $\mathbf{p} \in V(T)$ and $\mathbf{q} \preceq \mathbf{p}$, then $\mathbf{q} \in V(T)$. A *branch*, denoted $B_\mathbf{p} = (V(B_\mathbf{p}), E(B_\mathbf{p}))$, is a pair consisting of the set of nodes that succeed a certain node $\mathbf{p}$, i.e., $V(B_\mathbf{p}) = \{\mathbf{q} \in V \mid \mathbf{q} \succeq \mathbf{p}\}$, and the set of edges that enter these nodes, i.e., $E(B_\mathbf{p}) = \{e_\mathbf{q} \mid \mathbf{q} \in V(B_\mathbf{p})\}$. Note that a branch $B_\mathbf{p}$ is not a graph, because $(\pi(\mathbf{p}), \mathbf{p}) \in E(B_\mathbf{p})$, but $\pi(\mathbf{p}) \notin V(B_\mathbf{p})$. The complement of branch $B_\mathbf{p}$ is called $T_\mathbf{p}$. For trunk $T$ we define the total costs $C(T) = b(\mathbf{o}) + \sum_{\mathbf{p} \in V(T) \setminus \{\mathbf{o}\}} (a_\mathbf{p} + b(\mathbf{p}))$ and, similarly, for branch $B_\mathbf{p}$ we define $C(B_\mathbf{p}) = \sum_{\mathbf{q} \in V(B_\mathbf{p})} (a_\mathbf{q} + b(\mathbf{q}))$.

In one possible interpretation of the model the nodes $\mathbf{p}$ are villages that are connected via a road system to a central supplier $\mathbf{o}$. The costs on the edges are the "road maintenance costs" of the connections and the costs in the nodes are "local maintenance costs." The issue is: how to share the maintenance costs among the residents of the villages? In order to give an answer to this question we associate to $\Gamma$ the transferable utility *tree game* $(N; c_\Gamma)$. The costs for a coalition $S \subseteq N$, $S \neq \emptyset$, are defined by the costs of the cheapest trunk that connects all members of $S$ to the root:

$$c_\Gamma(S) = \min\{C(T) \ : \ T \text{ is a trunk and } S \subseteq N(T)\}.$$

Here, $N(T)$ is an abbreviation for $\bigcup_{\mathbf{p} \in V(T)} N_\mathbf{p}$.

Let $T_S$ denote the trunk spanned by the root and the nodes in which the members of $S$ reside ($S \subseteq N$). Since the cost functions $a$ and $b$ can have negative values, it may well happen that $c_\Gamma(S) < C(T_S)$.

A *standard tree network* obeys the following additional properties (*cf.* Definition 2.5 of Granot et al. (1996)):

(a) The cost function $a$ on the edges is non-negative.
(b) The cost function $b$ on the nodes is the zero function.
(c) $N_\mathbf{p} \neq \emptyset$ for every leaf $\mathbf{p}$.

Games generated by standard tree networks are called standard tree games. They have the property that for each coalition $S \subseteq N$,

$$c_\Gamma(S) = C(T_S).$$

The following lemma shows that the class of standard tree games coincides with the class of tree games of which all trunks have non-negative costs. This is the class of tree networks that will be considered in this paper.

**Lemma 2.1** *Let $\Gamma$ be a tree network such that all trunks have non-negative costs. Then there is a standard tree network $\bar{\Gamma}$ that generates the same cost game.*

*Proof* We modify the tree network gradually and prove that the generated game does not change under each given action. In the description of an action, we will only state the data that change and indicate the changed data with a bar.

- First, we transfer the costs in the nodes $\mathbf{p} \in V \setminus \{\mathbf{o}\}$ to the edges: $\bar{b}(\mathbf{p}) = 0$ and $\bar{a}_{\mathbf{p}} = a_{\mathbf{p}} + b(\mathbf{p})$.
  This action does not change costs of trunks and, therefore, neither coalitional costs.
- If the costs on an edge $e_{\mathbf{p}}$ are negative and $\pi(\mathbf{p}) \neq \mathbf{o}$, we replace $a_{\mathbf{p}}$ by zero and add $a_{\mathbf{p}}$ to $a_{\pi(\mathbf{p})}$. If $a_{\mathbf{p}} < 0$ and $\pi(\mathbf{p}) = \mathbf{o}$ we replace again $a_{\mathbf{p}}$ by zero but add $a_{\mathbf{p}}$ to the costs of the root.
  This action neither changes the coalitional costs, nor the nonnegativity of the costs of trunks, as, if $a_{\mathbf{p}} < 0$, a trunk containing $\pi(\mathbf{p})$ and not $\mathbf{p}$ cannot be optimal for any coalition; it is always better to add node $\mathbf{p}$. So optimal trunks contain both $\mathbf{p}$ and $\pi(\mathbf{p})$ or none of them. The costs of these trunks are not changed by the proposed action. Repeating this action as long as there are edges with negative costs leads to a nonnegative costs function $a$. The final costs at the root are nonnegative, because they are equal to the initial costs of some trunk, perhaps consisting of the root alone.
- If $b(\mathbf{o}) > 0$, we add a new root $\bullet$ and define $\bar{\pi}(\mathbf{o}) = \bullet$, $\bar{a}_{\mathbf{o}} = b(\mathbf{o})$, and $\bar{b}(\bullet) = 0$.
  Again, coalitional costs remain the same. By now, properties (a) and (b) are met.
- If $\mathbf{p}$ is a leaf and $N_{\mathbf{p}} = \emptyset$, we remove $\mathbf{p}$ and $e_{\mathbf{p}}$ from the network.
  If $\mathbf{p}$ occurs in a trunk optimal for some coalition, node $\mathbf{p}$ and its edge can be omitted from the optimal trunk without losing the optimality of the trunk. Hence, deleting such a leaf $\mathbf{p}$ and its edge $e_{\mathbf{p}}$ does not change the generated game. Repeating this action as long as there are leaves $\mathbf{p}$ with $n_{\mathbf{p}} = 0$, we get a tree network satisfying property (c). □

Granot et al. (1996), Theorem 2.6, show that a standard tree game is convex. Our definition of being standard differs from theirs, but their proof can be adapted in a straightforward manner. A direct consequence of their result and the previous lemma is thereby

**Corollary 2.2** *Let $\Gamma$ be a tree network such that all trunks have non-negative costs. Then the game $(N; c_\Gamma)$ is convex.*

The following two lemmas show that the core and the nucleolus of a standard tree game $(N; c_\Gamma)$ depend on the coalitions $N \backslash \{i\}$, $(i \in N)$, and $N(T_{\mathbf{p}})$, $(\mathbf{p} \in V \setminus \{\mathbf{o}\})$, only. We denote this collection of coalitions by $\mathcal{E}$.

**Lemma 2.3** *Let $\Gamma$ be a standard tree network with player set $N$ and let $x \in \Re^N$. Then*

$$x \in \operatorname{Core}(N; c_\Gamma) \text{ if and only if } x(N) = c_\Gamma(N), \quad x(S) \leq c_\Gamma(S) \text{ for all } S \in \mathcal{E}.$$

*Proof* Clearly, the conditions are necessary. To prove that they are sufficient, let $x$ satisfy the restricted set of constraints above. It is to be showed that $x$ is a core point. For every player $i$ in $N$ we have

$$x_i = c_\Gamma(N) - x(N\backslash\{i\}) \geq c_\Gamma(N) - c_\Gamma(N\backslash\{i\}) \geq 0.$$

Let $S$ be a proper subset of $N$. Denote the set of nodes adjacent to $T_S$ by $I_S$, i.e., $I_S = \{\mathbf{p} \in V\backslash V(T_S) \mid \pi(\mathbf{p}) \in V(T_S)\}$. We have

$$c_\Gamma(S) = C(T_S) = c_\Gamma(N) - \sum_{\mathbf{p}\in I_S} C(B_\mathbf{p}).$$

As

$$C(B_\mathbf{p}) = c_\Gamma(N) - C(T_\mathbf{p}) \leq x(N) - x(N(T_\mathbf{p})) = x(N(B_\mathbf{p})),$$

we find

$$c_\Gamma(S) \geq c_\Gamma(N) - \sum_{\mathbf{p}\in I_S} x(N(B_\mathbf{p})) = x(N(T_S)) \geq x(S).$$

□

The *excess* of coalition $S$ in the game $(N; c_\Gamma)$ with respect to $x \in \Re^N$ is defined by

$$\text{exc}_\Gamma(S, x) = c_\Gamma(S) - x(S).$$

The *nucleolus* $\text{nu}(N; c_\Gamma)$, or simply nu if no confusion can occur, of the game $(N; c_\Gamma)$ is the imputation[1] that maximizes the excesses of all coalitions lexicographically, i.e., it has the largest possible minimum excess, under the imputations with maximal minimum excess, it has the largest possible second minimum excess, and so on. The following lemma shows that it is not necessary to take all coalitions into account. If we consider only the coalitions in $\mathcal{E}$, the nucleolus is found as well.

**Lemma 2.4** *Let $\Gamma$ be a standard tree network and let the imputation $x$ maximize the excesses $\{\text{exc}_\Gamma(S, x) : S \in \mathcal{E}\}$ lexicographically. Then $x$ is the nucleolus of $(N; c_\Gamma)$.*

*Proof* Since the game $(N; c_\Gamma)$ is convex, its kernel is a single point (see Maschler et al. 1972) which is therefore the nucleolus. The kernel intersected with the core is a locus of the core (Maschler et al. 1979), i.e., any two games having the same core have the same intersection of core and kernel. Thus, any game with the same core as $(N; c_\Gamma)$ has the same nucleolus as $(N; c_\Gamma)$ as well.

By Lemma 2.3, we get a game with the same core as $(N; c_\Gamma)$ if we increase the costs of all coalitions not in $\mathcal{E}$. Let $(N, \bar{c})$ be a game obtained in this way such that for any element of the core, the excess of any coalition outside $\mathcal{E}$ exceeds the excess

[1] A vector $x \in \Re^N$ is an *imputation* of $(N; c_\Gamma)$ if $x(N) = c_\Gamma(N)$ and $x_i \leq c_\Gamma(\{i\})$ for all $i \in N$.

of any coalition inside $\mathcal{E}$. By lexicographically maximizing the excesses with respect to the game $(N, \bar{c})$, we reach a single point at the moment when the excesses of all coalitions in $\mathcal{E}$ have become constant, because the excesses of the coalitions of type $N \backslash \{i\}$ determine this point. Hence, the nucleolus of $(N, \bar{c})$, which as we have seen is also the nucleolus of $(N; c_\Gamma)$, maximizes the excesses $\{\text{exc}_\Gamma(S, x) : S \in \mathcal{E}\}$ lexicographically. $\square$

Let us take another look at the proof above to obtain a useful corollary. Since the games $(N; c_\Gamma)$ and $(N, \bar{c})$ are balanced, their prenucleoli[2] coincide with their nucleoli and hence, with each other. A collection of coalitions $\mathcal{B}$ is called *balanced* if there exist positive reals $\lambda_S$, $S \in \mathcal{B}$, such that $\sum_{S \in \mathcal{B}} \lambda_S e_S = e_N$, in which $e_S$ denotes the indicator vector of coalition $S$. Recall Sobolev's criterion (cf. Kohlberg (1971)) for the prenucleolus (in terms of cost games):

**Theorem 2.5** (Sobolev (1975)) *A preimputation of a cost game $(N, c)$ equals the prenucleolus of this game if and only if for all $t \in \Re$ the set $\{S \subset N | \, c(S) - x(S) \leq t\}$ is balanced or empty.*

Because the nucleolus of $(N; c_\Gamma)$ coincides with the prenucleolus of $(N, \bar{c})$, we obtain

**Corollary 2.6** *Let $x$ be a (pre-)imputation of the game $(N; c_\Gamma)$. Then $x = \text{nu}(N; c_\Gamma)$ if and only if for all $t \in \Re$ the collection $\{S \in \mathcal{E} | \, \text{exc}_\Gamma(S, x) \leq t\}$ is balanced or empty.*

## 3 An algorithm for the nucleolus

In this section we present an algorithm for the nucleolus of games associated with standard tree networks. In the literature one finds two different approaches to the computation of the nucleolus of standard tree games. The first algorithm is due to Megiddo (1978). It runs as follows. For every trunk $T$ in $\Gamma$ one can define its *weight* $w(T)$ to be $\frac{C(T)}{D(T)}$, where $C(T)$ denotes the total costs on the edges of $T$ and $D(T)$ denotes the sum of the number of players in $T$ and the number of edges outgoing from $T$. To compute the nucleolus one has to find the largest trunk $T^*$ with minimal weight. The players in $T^*$ pay each $w(T^*)$ under the nucleolus. The trunk $T^*$ is then absorbed to the root and the costs of the outgoing arcs of $T^*$ are increased by $w(T^*)$. In the contracted tree we repeat these steps. Megiddo proved that this algorithm yields the nucleolus and that the complexity of the algorithm is $\mathcal{O}(q^3)$. Galil (1980) improved the performance of the algorithm mainly by shortening the method to find $T^*$. The complexity of his algorithm is $\mathcal{O}(q \log q)$. A different algorithm for the nucleolus of standard tree games can be found in Granot et al. (1996). It assumes that all nodes are inhabited. This algorithm starts with the computation of—what they call—the *proto-nucleolus*. The proto-nucleolus $\nu^*$ is computed recursively, starting from the root: put $\nu^*_{\mathbf{o}} = 0$. If the component $\nu^*_{\pi(\mathbf{p})}$ of the proto-nucleolus has been computed, define

[2] The *prenucleolus* of a cost game $(N, c)$ is the preimputation that maximizes the excesses of all coalitions lexicographically. A vector $x \in \Re^N$ is said to be a *preimputation* if $x(N) = c(N)$.

$\nu^*_{\mathbf{p}} = \frac{a_{\mathbf{p}}+\nu^*_{\pi(\mathbf{p})}}{n_{\mathbf{p}}+d_{\mathbf{p}}}$. Thus, each player in $N_{\mathbf{p}}$ pays $\nu^*_{\mathbf{p}}$. This computation requires $\mathcal{O}(q)$ steps. If the proto-nucleolus $\nu^*$ turns out to satisfy $\nu^*_{\mathbf{p}} \leq \nu^*_{\mathbf{q}}$ whenever $\mathbf{p} \preceq \mathbf{q}$, the resulting payoff vector is the nucleolus of the game. If not, then there are "bad" edges, i.e., edges $e_{\mathbf{p}}$ with $\nu^*_{\mathbf{p}} < \nu^*_{\pi(\mathbf{p})}$. By contracting bad edges in an *appropriate order*, one finds, finally, a contracted tree whose proto-nucleolus is the nucleolus of the original game. Contracting means eliminating the edge, adding the residents of $\mathbf{p}$ to $\pi(\mathbf{p})$ and increasing the costs of $e_{\pi(\mathbf{p})}$ by the costs of the eliminated edge. The subtlety of the algorithm lies in the order in which the edges must be contracted. For each bad edge $e_{\mathbf{p}}$, compute $\frac{a_{\mathbf{p}}}{d_{\mathbf{p}}-1}$ and eliminate that edge for which this expression is minimal. The algorithm has a complexity of $\mathcal{O}(q^2)$, but for important classes of trees, such as airport games and binary trees, the algorithm is even linear.

The main advantage of Megiddo's algorithm is the fact that its description is the shortest. Also, it sheds some light on the nature of the nucleolus for standard trees. However, we find the intuitive meaning of $C(T)/D(T)$ hard to explain. Granot et al.'s algorithm has the advantage that quite often it provides the shortest procedure to reach the nucleolus. The proto-nucleolus is quite intuitive in economic terms and so is the contraction procedure. However, we do not know of any intuitive explanation that justifies the required order of contractions.

In this section we present yet another algorithm which is, in our opinion, the most intuitive of the three. The algorithm runs as follows. Let $\Gamma$ be a standard tree network. The heart of the algorithm consists of $q$ iterations in which the costs on the edges are partly allocated. Each iteration allocates the (remaining) costs of at least one edge completely, resulting in an edge with zero costs. This edge is contracted.

To keep track of how much costs players are allocated, a variable $x$ is introduced. Since players with the same residence are treated equally, it is not necessary to store this amount for each player individually. So $x$ is an element of $\Re^V$ and $x_{\mathbf{p}}$ denotes the amount that every player residing in $\mathbf{p}$ has been allocated so far.

We define a "shadow tree network" $\Gamma^0$. It has the same tree and the same values $n_{\mathbf{p}}$ as $\Gamma$. The costs of the edges, however, are different. Let for each node $\mathbf{p} \in V \setminus \{\mathbf{o}\}$ its *grade* $g_{\mathbf{p}}$ be defined by $g_{\mathbf{p}} = n_{\mathbf{p}} + d_{\mathbf{p}} - i_{\mathbf{p}}$. The number $i_{\mathbf{p}}$ equals 1 if $\pi(\mathbf{p}) \neq \mathbf{o}$ and equals 0 if $\pi(\mathbf{p}) = \mathbf{o}$. The costs of an edge of the shadow tree are equal to its grade.

The number $y \in \Re_+$ is computed satisfying $a_{\mathbf{p}} - y g_{\mathbf{p}} \geq 0$ for all $\mathbf{p} \in V \setminus \{\mathbf{o}\}$ with at least one equality. Since $n_{\mathbf{p}} + d_{\mathbf{p}} > 0$ by property (c), we have $g_{\mathbf{p}} \geq 0$ with a strict inequality if $i_{\mathbf{p}} = 0$. Hence, the number $y$ is well defined. We denote by $\Gamma - y\,\Gamma^0$ the tree network having the same tree and values $n_{\mathbf{p}}$ as $\Gamma$, and with costs $a_{\mathbf{p}} - y\,g_{\mathbf{p}}$ ($\mathbf{p} \in V \setminus \{\mathbf{o}\}$). The tree $\Gamma$ is replaced by $\Gamma - y\,\Gamma^0$. In order to finance this costs reduction, all players not residing in the root are allocated an amount of $y$, i.e., $x_{\mathbf{p}}$ increases by $y$ for all nodes $\mathbf{p}$ except the root.

After this partial allocation of the costs, one[3] edge, say $e_{\mathbf{p}}$, with, by now, zero costs is contracted as follows. Let $V_a \subset V \setminus \{\mathbf{o}\}$ denote the set of nodes that are still *active* (of which the edges have not been contracted yet). $\mathbf{p}$ is removed from $V_a$, the residents of $\mathbf{p}$ move to $\pi(\mathbf{p})$, ($n_{\pi(\mathbf{p})} = n_{\pi(\mathbf{p})} + n_{\mathbf{p}}$), and the parent map $\pi$ is adapted by

[3] If there are multiple edges with zero costs, an arbitrary one is chosen. In the next iteration, $y$ will equal zero.

$\pi(\mathbf{q}) = \pi(\mathbf{p})$ for all $\mathbf{q} \in V_a$ with $\pi(\mathbf{q}) = \mathbf{p}$. Note that after contraction, the resulting tree network is still standard. In order to keep track of to which vertices the original nodes have been contracted, we introduce another function. $r : V \setminus \{\mathbf{o}\} \longrightarrow V_a \cup \{\mathbf{o}\}$ denotes for each original node $\mathbf{p} \in V \setminus \{\mathbf{o}\}$ the node to which it is contracted. So, initially $r(\mathbf{p}) = \mathbf{p}$ and when edge $e_\mathbf{p}$ is contracted, we set $r(\mathbf{q}) = \pi(\mathbf{p})$ for all nodes $\mathbf{q}$ with $r(\mathbf{q}) = \mathbf{p}$. After the contraction another iteration starts, until all edges have been contracted and all costs have been divided.

Let us here present a scheme of the algorithm.

**Scheme of the algorithm**

**Input**

A standard tree network $\Gamma$ with data $V$ (of size $q+1$), $\mathbf{o}$, $\pi$, $(n_\mathbf{p})_{\mathbf{p}\in V}$, and $(a_\mathbf{p})_{\mathbf{p}\in V\setminus\{\mathbf{o}\}}$.

**Initialization**

Define $x = 0 \in \Re^V$, $q_a = q$, $V_a = V \setminus \{\mathbf{o}\}$, and let for all $\mathbf{p} \in V_a$:

$$d_\mathbf{p} = |\{\mathbf{q} \in V \mid \pi(\mathbf{q}) = \mathbf{p}\}|, \quad i_\mathbf{p} = \begin{cases} 1 & \text{if } \pi(\mathbf{p}) = \mathbf{o}, \\ 0 & \text{if } \pi(\mathbf{p}) \neq \mathbf{o}, \end{cases} \text{ and } \mathrm{r}(\mathbf{p}) = \mathbf{p}.$$

**Iterations**

```
As long as $q_a > 0$, perform the following steps:
    let $g_\mathbf{p} = n_\mathbf{p} + d_\mathbf{p} - i_\mathbf{p}$ for all $\mathbf{p} \in V_a$,
    take $y$ as the largest number satisfying $a_\mathbf{p} - y\,g_\mathbf{p} \geq 0$ for all $\mathbf{p} \in V_a$,
    for each $\mathbf{p} \in V \setminus \{\mathbf{o}\}$ with $r(\mathbf{p}) \neq \mathbf{o}$, increase $x_\mathbf{p}$ by $y$,
    set $a_\mathbf{p} = a_\mathbf{p} - y g_\mathbf{p}$ for all $\mathbf{p} \in V_a$,
    choose a node $\mathbf{p} \in V_a$ with $a_\mathbf{p} = 0$,
    let $V_a = V_a \setminus \{\mathbf{p}\}$ and $q_a = q_a - 1$,
    let $n_{\pi(\mathbf{p})} = n_{\pi(\mathbf{p})} + n_\mathbf{p}$ and $d_{\pi(\mathbf{p})} = d_{\pi(\mathbf{p})} + d_\mathbf{p} - 1$,
    let $\pi(\mathbf{q}) = \pi(\mathbf{p})$ for all $\mathbf{q} \in V_a$ with $\pi(\mathbf{q}) = \mathbf{p}$,
    let $r(\mathbf{q}) = \pi(\mathbf{p})$ for all $\mathbf{q} \in V_a$ with $r(\mathbf{q}) = \mathbf{p}$,
    if $\pi(\mathbf{p}) = \mathbf{o}$, let $i_\mathbf{q} = 0$ for all $\mathbf{q} \in V_a$ with $\pi(\mathbf{q}) = \mathbf{o}$.
```

**Output**

$\mathrm{Nu}(\Gamma) = x$.

Let us illustrate the algorithm with an example.

*Example 3.1* Consider the standard tree network $\Gamma$ depicted in Fig. 1a. It has four villages $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{e}$. The number inside a village equals the number of inhabitants. Node $\mathbf{d}$ is uninhabited, say, a storage center. The number alongside an edge $e_\mathbf{p}$ denotes its costs $a_\mathbf{p}$.

First, the grades $g_\mathbf{p}$ of the nodes are determined to find the costs of the shadow tree $\Gamma^0$ (Fig. 1b). By comparing $\Gamma$ and $\Gamma^0$, we find that in the first iteration $y$ equals 3. Since no nodes are contracted to the root yet, $x$ is set to (3, 3, 3, 3, 3). The costs of each edge are reduced by 3 times its grade. By now, edge $a_\mathbf{a}$ has zero costs and

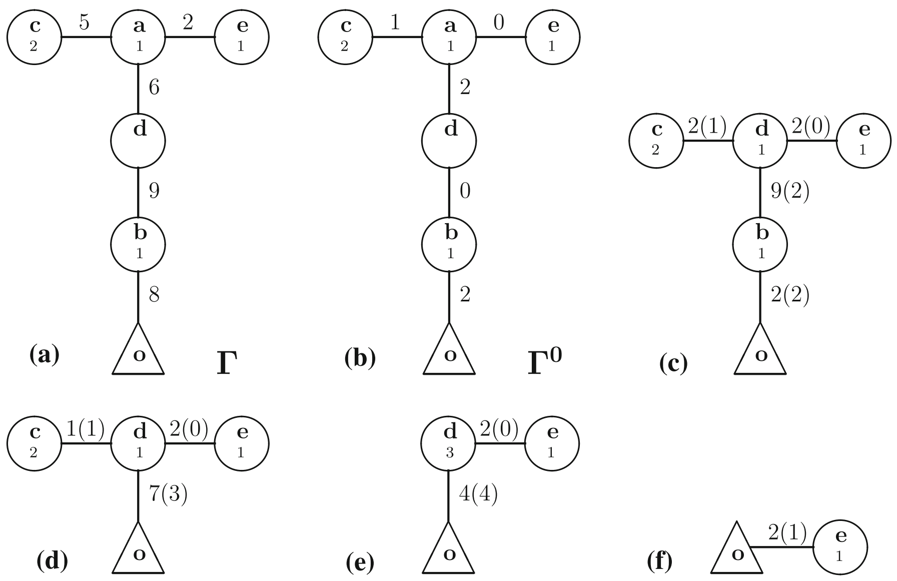

**Fig. 1** The standard tree network $\Gamma$ (**a**), its shadow tree $\Gamma^0$ (**b**), and the resulting trees after four (**c**–**f**) consecutive contractions of edges (costs between brackets are of the corresponding shadow trees)

is contracted. Node **a** is no longer active (i.e., $V_a = V \backslash \{\mathbf{a}\}$). The inhabitants of node **a** move to **d** ($n_{\pi(\mathbf{a})} = n_{\pi(\mathbf{a})} + n_{\mathbf{a}}$). Nodes **c** and **e** get **d** as their parent. The number $d_{\mathbf{d}}$ is set to 2, since after contraction there are two *active* nodes that have **d** as their parent (**c** and **e**). $r(\mathbf{a})$, denoting the current active node with which **a** is contracted, is set to **d**.

The second iteration starts with the tree as depicted in Fig. 1c. The grades are updated (and depicted between brackets). This time $y$ equals 1 and $x$ becomes $(4, 4, 4, 4, 4)$. **b** is contracted to the root. Other variables are updated similar as in the first iteration.

In the third iteration $y$ equals 1 again. Since **b** is contracted to the root, i.e., $r(\mathbf{b}) = \mathbf{o}$, $x_{\mathbf{b}}$ does not change anymore and $x$ is set to $(5, 4, 5, 5, 5)$. After the fourth iteration $x$ equals $(6, 4, 6, 6, 6)$. In the final iteration all nodes but **e** are contracted to the root. The final value 2 of $y$ is only added to $x_{\mathbf{e}}$. The output is Nu$= (6, 4, 6, 6, 8)$. □

**Discussion** We will call the output Nu$(\Gamma)$ the nucleolus of $\Gamma$. The nucleolus value of a player can be obtained by taking the nucleolus-value of his residency, i.e., if $i \in N_{\mathbf{p}}$, then $\text{nu}_i = \text{Nu}_{\mathbf{p}}$.

Note that the players absorbed by the root are not charged any further. Consequently, $\text{nu}_i \leq \text{nu}_j$, whenever $i \in N_{\mathbf{p}}$, $j \in N_{\mathbf{q}}$ and $\mathbf{p} \preceq \mathbf{q}$. We regard this as a *fairness property* which should be satisfied by any reasonable solution. Indeed, player $j$ should pay more since he uses all edges used by player $i$, and probably more.

Let us consider the complexity of the algorithm. The size of the input is of order $\mathcal{O}(q)$. In order to find the values $d_{\mathbf{p}}$, define $d_{\mathbf{p}} = 0$ for all $\mathbf{p} \in V$, followed by $d_{\pi(\mathbf{p})} = d_{\pi(\mathbf{p})} + 1$ for all $\mathbf{p} \in V \setminus \{\mathbf{o}\}$. Hence, all initial steps are of linear order as well. There are $q$ iterations. Within a single iteration, each line takes $\mathcal{O}(q)$

steps. Consequently, the algorithm has a complexity of order $\mathcal{O}(q^2)$. Similar to Galil's procedure (1980) to shorten Megiddo's algorithm, we can improve the performance to $\mathcal{O}(q \log q)$. This improvement will be omitted.

In the next theorem we prove that the algorithm computes the nucleolus.

**Theorem 3.2** *The algorithm above computes for a standard tree network* $\Gamma$ *its nucleolus* Nu. *The nucleolus of the associated game* $(N; c_\Gamma)$ *can be derived by* $\mathrm{nu}_i = \mathrm{Nu}_{\mathbf{p}}$ *for all* $i \in N$, $\mathbf{p} \in V$ *with* $i \in N_{\mathbf{p}}$.

*Proof* The proof is by induction on $q$. If $q = 0$, then $\mathrm{nu}_i = \mathrm{Nu}_{\mathbf{o}} = 0$ for all $i \in N$, which is in line with the algorithm. Suppose that the theorem is valid for tree networks with less than $q$ edges. Let $\Gamma$ be a standard tree network with $q$ edges. The first iteration changes the network as follows:

$$\Gamma \xrightarrow{\text{subtraction}} \Gamma - y\,\Gamma^0 \xrightarrow{\text{contraction}} \overline{\Gamma - y\,\Gamma^0}.$$

After one iteration we have the contracted network $\overline{\Gamma - y\,\Gamma^0}$; a tree network with less than $q$ edges. Let $x$ and $\bar{x}$ denote the output of the algorithm applied to the tree networks $\Gamma$ and $\overline{\Gamma - y\,\Gamma^0}$ respectively. We have $x = \bar{x} + y\,\mathrm{e}_{V\setminus\{\mathbf{o}\}}$, in which $\mathrm{e}_{V\setminus\{\mathbf{o}\}}$ denotes the characteristic vector of $V \setminus \{\mathbf{o}\}$.

Let $z \in \Re^N$ be such that $z_i = x_{\mathbf{p}}$ for all $i \in N$, $\mathbf{p} \in V$ with $i \in N_{\mathbf{p}}$. The induction hypothesis implies that $\bar{x} = \mathrm{Nu}(N; \overline{\Gamma - y\,\Gamma^0})$. The network $\Gamma - y\,\Gamma^0$ generates the same cost game as $\overline{\Gamma - y\,\Gamma^0}$. Thus, we find

$$z = \mathrm{nu}(N; c_{\Gamma - y\,\Gamma^0}) + y\mathrm{e}_{N\setminus N_{\mathbf{o}}}.$$

In order to derive that $z$ equals $\mathrm{nu}(N; c_\Gamma)$, we first show its efficiency. For the rest of the proof we use the abbreviations $\tilde{\Gamma} = \Gamma - y\,\Gamma^0$ and $\tilde{z} = \mathrm{nu}(N; c_{\tilde{\Gamma}})$. Because $\sum_{\mathbf{p}\neq\mathbf{o}} d_{\mathbf{p}} = \sum_{\mathbf{p}\neq\mathbf{o}} i_{\mathbf{p}} = |\{e_{\mathbf{p}} \in E \mid \pi(\mathbf{p}) \neq \mathbf{o}\}|$, we have $\sum_{\mathbf{p}\neq\mathbf{o}} g_{\mathbf{p}} = \sum_{\mathbf{p}\neq\mathbf{o}} n_{\mathbf{p}}$. Thus

$$z(N) = \tilde{z}(N) + y \sum_{\mathbf{p}\neq\mathbf{o}} n_{\mathbf{p}} = c_{\tilde{\Gamma}}(N) + y \sum_{\mathbf{p}\neq\mathbf{o}} g_{\mathbf{p}} = c_\Gamma(N),$$

and $z$ is efficient.

Since $\Gamma$ and $\tilde{\Gamma}$ only differ in costs, the set of coalitions $\mathcal{E}$ is the same for both tree networks. The induction hypothesis and Corollary 2.6 give that for all $t \in \Re$ the collections $\left\{S \in \mathcal{E} \mid \mathrm{exc}_{\tilde{\Gamma}}(S, \tilde{z}) \leq t\right\}$ are balanced whenever they are not empty. We must show that these collections are also balanced with respect to $z$ and the game $(N; c_\Gamma)$. Therefore we compute and compare the excesses of the coalitions in $\mathcal{E}$ with respect to the networks $\Gamma$ and $\tilde{\Gamma}$.

For $i \in N_{\mathbf{o}}$, we have $z_i = \tilde{z}_i = 0$ and $\mathrm{exc}_\Gamma(N\setminus\{i\}, z) = 0$.

For $\mathbf{p} \in V \setminus \{\mathbf{o}\}$ with $\pi(\mathbf{p}) = \mathbf{o}$ we have

$$\begin{aligned} c_\Gamma(N(T_\mathbf{p})) &= c_{\tilde{\Gamma}}(N(T_\mathbf{p})) + \sum_{\mathbf{q} \in V(T_\mathbf{p})\setminus\{\mathbf{o}\}} y(n_\mathbf{q} + d_\mathbf{q} - i_\mathbf{q}) \\ &= c_{\tilde{\Gamma}}(N(T_\mathbf{p})) + y\,|N(T_\mathbf{p})\backslash N_\mathbf{o}| + y\,|\{e_\mathbf{q} \in E(T_\mathbf{p}) \mid \pi(\mathbf{q}) \neq \mathbf{o}\}| \\ &\quad - y\,|\{e_\mathbf{q} \in E(T_\mathbf{p}) \mid \pi(\mathbf{q}) \neq \mathbf{o}\}| \\ &= c_{\tilde{\Gamma}}(N(T_\mathbf{p})) + y\,|N(T_\mathbf{p})\backslash N_\mathbf{o}|. \end{aligned}$$

To verify the second equality, recall that trunk $T_\mathbf{p}$ is spanned by the nodes that can be reached without passing $\mathbf{p}$ and that by definition $\mathbf{p} \notin V(T_\mathbf{p})$. The result above yields

$$\begin{aligned} \mathrm{exc}_\Gamma(N(T_\mathbf{p}), z) &= \big(c_{\tilde{\Gamma}}(N(T_\mathbf{p})) + y\,|N(T_\mathbf{p})\backslash N_\mathbf{o}|\big) - \big(\tilde{z}(N(T_\mathbf{p})) + y\,|N(T_\mathbf{p})\backslash N_\mathbf{o}|\big) \\ &= \mathrm{exc}_{\tilde{\Gamma}}(N(T_\mathbf{p}), \tilde{z}) \\ &= 0. \end{aligned}$$

The latter equality is valid because $N$ is the disjoint union of $N(T_\mathbf{p})$ and $N(B_\mathbf{p})$, $\tilde{z}$ is a core element of $(N; \tilde{c})$, and, since $\pi(\mathbf{p}) = 0$, $\tilde{c}(N(T_\mathbf{p})) + \tilde{c}(N(B_\mathbf{p})) = \tilde{c}(N)$.

For $\mathbf{p} \in V \setminus \{\mathbf{o}\}$ with $\pi(\mathbf{p}) \neq \mathbf{o}$ we have

$$\begin{aligned} c_\Gamma(N(T_\mathbf{p})) &= c_{\tilde{\Gamma}}(N(T_\mathbf{p})) + y\,|N(T_\mathbf{p})\backslash N_\mathbf{o}| + y\,|\{e_\mathbf{q} \in E(T_\mathbf{p}) \mid \pi(\mathbf{q}) \neq \mathbf{o}\} \cup \{e_\mathbf{p}\}| \\ &\quad - y\,|\{e_\mathbf{q} \in E(T_\mathbf{p}) \mid \pi(\mathbf{q}) \neq \mathbf{o}\}| \\ &= c_{\tilde{\Gamma}}(N(T_\mathbf{p})) + y\,|N(T_\mathbf{p})\backslash N_\mathbf{o}| + y. \end{aligned}$$

Hence, $\mathrm{exc}_\Gamma(N(T_\mathbf{p}), z) = \mathrm{exc}_{\tilde{\Gamma}}(N(T_\mathbf{p}), \tilde{z}) + y$.

Let $i \in N\backslash N_\mathbf{o}$. If $i$ is the unique resident of some leaf $\mathbf{p}$, then $N\backslash\{i\} = N(T_\mathbf{p})$, so $\mathrm{exc}_\Gamma(N\backslash\{i\}, z) = \mathrm{exc}_{\tilde{\Gamma}}(N\backslash\{i\}, \tilde{z}) + y$. Otherwise, $c_\Gamma(N\backslash\{i\}) = c_\Gamma(N)$, yielding

$$\mathrm{exc}_\Gamma(N\backslash\{i\}, z) = c_\Gamma(N) - z(N\backslash\{i\}) = z_i = \tilde{z}_i + y = \mathrm{exc}_{\tilde{\Gamma}}(N\backslash\{i\}, \tilde{z}) + y.$$

Resuming, we find $\{S \in \mathcal{E} \mid \mathrm{exc}_\Gamma(S, z) < 0\} = \emptyset$ and, for $0 \leq t < y$,

$$\{S \in \mathcal{E} \mid \mathrm{exc}_\Gamma(S, z) \leq t\} = \big\{N(T_\mathbf{p}) \mid \pi(\mathbf{p}) = \mathbf{o}\big\} \cup \{N\backslash\{i\} \mid i \in N_\mathbf{o}\}.$$

This collection is, if non-empty, balanced since

$$\sum_{\mathbf{p};\pi(\mathbf{p})=\mathbf{o}} \mathrm{e}_{N(T_\mathbf{p})} + \sum_{i \in N_\mathbf{o}} \mathrm{e}_{N\backslash\{i\}} = (d_\mathbf{o} + n_\mathbf{o} - 1)\mathrm{e}_N.$$

For $t \geq y$, we have

$$\{S \in \mathcal{E} \mid \mathrm{exc}_\Gamma(S, z) \leq t\} = \big\{S \in \mathcal{E} \mid \mathrm{exc}_{\tilde{\Gamma}}(S, \tilde{z}) \leq t - y\big\}.$$

This collection is, if non-empty, balanced because $\tilde{z} = \mathrm{nu}(N; c_{\tilde{\Gamma}})$ and Corollary 2.6. Now the induction step and the theorem follow by applying Corollary 2.6 on $z$ with respect to $c_\Gamma$. □

## 4 An interpretation of the nucleolus of a tree game: a painting story

This section provides an easily understandable interpretation to the algorithm presented in the previous section. This interpretation is of prime importance as it provides an additional insight that enables people to rationally adopt the nucleolus, or reject it, in any application at hand.

We start by viewing a tree network as a system of *road segments* connecting *villages* to the *capital city* (the root). The players are now *villagers* who are responsible for maintaining the road segments, i.e., painting them regularly. The costs of an edge in the original game are now viewed as the *length* (in km) of the corresponding line-segment. This is also equal to the time of painting the road segment, given that each villager paints at unit speed. We are going to describe a way of painting in which the time each villager spends is numerically equal to his nucleolus payoff.

### 4.1 The painting story

The initial positions of the villagers at which they start the painting are determined as follows. In principle, all villagers start at their residencies, except for one per village. Each village $\mathbf{p} \in V \setminus \{\mathbf{o}\}$, is asked to choose a *representative* and send him to $\pi(\mathbf{p})$. Because not all villages are inhabited, this must be done with some care.

First, each village $\mathbf{p}$ located at an endpoint of the tree sends a representative forward to $\pi(\mathbf{p})$. (This is possible, because the end vertices are occupied).

Thereafter, each village $\mathbf{p}$ at which just one or more representatives arrived sends a representative to $\pi(\mathbf{p})$. This can be someone who was sent previously from an endpoint. In this case this person represents two villages. (This again is possible, because by now these villages are occupied.)

Subsequently, each village $\mathbf{p}$, at which just and for the first time a representative has arrived, chooses a representative and sends him to $\pi(\mathbf{p})$. This continues until all villages have chosen a representative. Villagers not chosen to represent start at their residencies. The painting proceeds in accordance with the following rules:

(i) Every villager traverses from his starting point to the capital city, and then from the capital city back to his residency. If he comes across an unpainted road, he paints it. If he comes to a painted part, he skips it. When he has returned to his residency, he stops painting.

(ii) Road segments are painted with a speed of $k$ km per hour, where $k$ is the number of villagers painting at that segment.

(iii) The time needed for a villager to traverse a segment already painted is negligible.

We are interested in the length of the time period that each resident is working. Let us consider an example.

*Example 4.1* Consider the standard tree network $\Gamma$ depicted in Fig. 2a. This time the numbers inside a node represent the names of its inhabitants. All roads have a length of 8 kilometers.

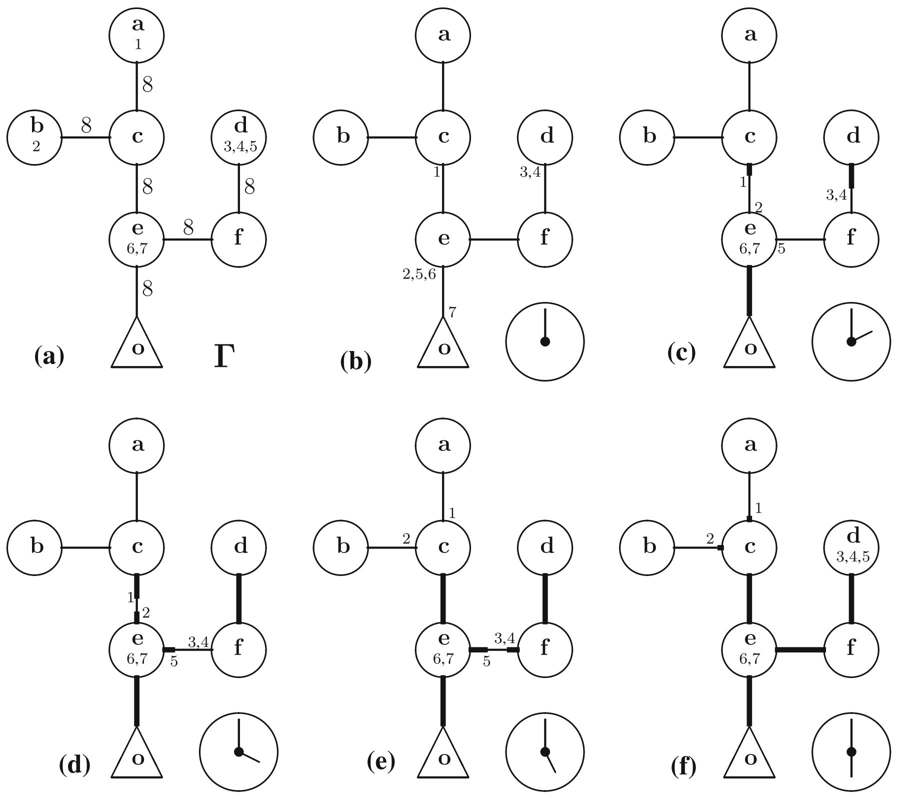

**Fig. 2** The standard tree network Γ (**a**) and the locations of the painters after 0 (**b**), 2 (**c**), 4 (**d**), 5 (**e**), and 6 h (**f**) of work respectively

First, let us say at 12 o'clock, the villagers take their positions to start the painting. Since each village can determine its representative, several options are possible. We describe one of them. First, the endpoints **a**, **b**, and **d** send villagers 1, 2, and 5 as their representatives to nodes **c**, **c**, and **f** respectively. Subsequently, villages **c** and **f** send villagers 2, and 5 as their representatives to node **e** respectively. Finally, village **e** chooses villager 7 to start in the capital city. Figure 2b depicts the initial locations of the villagers.

After 2 h, edge $e_{\mathbf{e}}$ has been painted completely. Villagers 6 and 7 return to their residency **e** and stop painting. Villagers 2 and 5 start their ways back to their residencies (Fig. 2c). After another 2 h, edge $e_{\mathbf{d}}$ has been completed. Villagers 3 and 4 move down one edge. The procedure continues in the way the figure displays. After 6 h, only villagers 1 and 2 still have painting obligations (Fig. 2f). They both paint for another 7 h. The villagers 1, . . . , 7 then have painted for 13, 13, 6, 6, 6, 2, and 2 h respectively. The following theorem states that these times form the nucleolus of the associated game.

**Theorem 4.2** *Let* Γ *be a standard tree network modeling a system of roads to be painted. If the painting is according to the painting story above, then each villager paints for a duration equal to his nucleolus payoff of the game* $(N; c_\Gamma)$.

*Proof* The algorithm and the story have two main differences. First, the algorithm is discrete, while in the story time is continuous. Second, in the algorithm edges are contracted, where in the story edges remain present, leaving the original tree structure intact. The first difference can be avoided by considering only moments in time in $\{0, t_1, \ldots, t_q\}$, where $t_k$ denotes the moment when for the $k$th time an edge is completed in the story. The second difference can be overcome by modifying the interpretations of the variables of the algorithm as follows. At each moment,

– $V_a$ denotes the set of nodes $\mathbf{p}$ in $V \setminus \{\mathbf{o}\}$ for which $e_{\mathbf{p}}$ has not been painted completely,
– $r(\mathbf{p})$ denotes the node in $(V_a \cup \{\mathbf{o}\})\setminus\{\mathbf{p}\}$ closest to $\mathbf{p}$ on the road from $\mathbf{p}$ to $\mathbf{o}$, $(\mathbf{p} \in V_a)$
– $n_{\mathbf{p}}$ denotes the number of villagers of which the roads from their residencies to $\mathbf{p}$ have been painted completely, $(\mathbf{p} \in V_a)$
– $a_{\mathbf{p}}$ denotes the size of the uncompleted part of edge $e_{\mathbf{p}}$, $(\mathbf{p} \in V_a)$
– $d_{\mathbf{p}}$ still equals $|\{q \in V_a \mid \pi(\mathbf{q}) = \mathbf{p}\}|$, $i_{\mathbf{p}}$ still indicates whether $\pi(\mathbf{p}) = \mathbf{o}$, and $g_{\mathbf{p}}$ still equals $n_{\mathbf{p}} + d_{\mathbf{p}} - i_{\mathbf{p}}$, $(\mathbf{p} \in V_a)$

Note that initially the interpretations coincide. Denote the $k$th value of $y$ in the algorithm by $y_k$. Let $k \in \{1, \ldots, q\}$ and suppose that the story has followed the algorithm up to time $t_{k-1}$, i.e., $t_1 = y_1, t_2 - t_1 = y_2, \ldots, t_{k-1} - t_{k-2} = y_{k-1}$, and the $\ell$th edge that has been contracted is the edge with completion time $t_\ell$ in the story $(\ell < k)$.

Consider the painting process at moment $t_{k-1}$. Let $\mathbf{p} \in V_a$. We are interested in the number of villagers that are painting somewhere on $e_{\mathbf{p}}$. Let $P$ be the largest subset of $V(B_{\mathbf{p}})$ such that $\mathbf{p} \in P$ and all roads between villages in $P$ are painted completely. Then $n_{\mathbf{p}}$ equals $|N(P)|$.

If $i_{\mathbf{p}} = 0$, i.e., $\pi(\mathbf{p}) = \mathbf{o}$, all members of $N(P)$ and all representatives of the villages in $\{\mathbf{q} \in V_a \mid \pi(\mathbf{q}) = \mathbf{p}\}$ are painting on $e_{\mathbf{p}}$, so in total $n_{\mathbf{p}} + d_{\mathbf{p}}$ are painting on $e_{\mathbf{p}}$. If $d_{\mathbf{p}} = 1$, the same set of villagers is painting on $e_{\mathbf{p}}$, *except* for the representative of $\mathbf{p}$, who is painting somewhere on the road from $\pi(\mathbf{p})$ to the city. In both cases the number of villagers currently working on $e_{\mathbf{p}}$ equals $g_{\mathbf{p}}$. Therefore, $t_k - t_{k-1} = \min\left\{\frac{a_{\mathbf{p}}}{g_{\mathbf{p}}} \mid \mathbf{p} \in V_a,\ g_{\mathbf{p}} \neq 0\right\}$, which equals $y_k$.

Hence, by induction, $t_k = \sum_{\ell=1}^{k} y_\ell$ and the $k^{\text{th}}$ edge that has been contracted in the algorithm equals the edge with completion time $t_k$ in the story $(k \in \{1, \ldots, q\})$. Since $\mathrm{Nu}_{\mathbf{p}} = \max\{\sum_{\ell=1}^{k} y_\ell \mid$ the $k$th edge that is contracted is situated between $\mathbf{p}$ and $\mathbf{o}\}$, and the length of time inhabitants of $\mathbf{p}$ paint equals $\max\{t_k \mid$ the $k$th edge that is fully painted is situated between $\mathbf{p}$ and $\mathbf{o}\}$, the algorithm and the painting story comport. □

## 5 Monotonicity properties of the nucleolus

This section shows that the nucleolus of a standard tree game is monotonic in two senses:

(i) If players are omitted from the network, none of the remaining players pay less than before, i.e., the nucleoli of all subgames of $(N; c_\Gamma)$ form a population monotonic allocation scheme (cf. Sprumont (1990)).
(ii) If costs of edges are increased, none of the players pays less.

**Theorem 5.1** *Let $\Gamma$ be a standard tree network. Let $\tilde{\Gamma}$ be obtained from $\Gamma$ by increasing the costs on some edges and/or decreasing the number of residents in some nodes. Then the remaining players pay a weakly larger contribution under the nucleolus in $\tilde{\Gamma}$.*

*Proof* Because of Theorem 4.2 we can compare $\Gamma$ and $\tilde{\Gamma}$ by means of the painting story. Let $\sigma$ be the coarsest partition of the time-schedule of the painting on $\Gamma$ into open time-segments during which no change in the number of painters on any edge takes place. Let $\tilde{\sigma}$ be the same for $\tilde{\Gamma}$. Let $\tau$ be the meet of the two partitions, i.e., the coarsest partition having the property that each segment in $\sigma$ or $\tilde{\sigma}$ is a union of segments in $\tau$. Denote the elements of $\tau$ by $\tau_1 \dots \tau_k$ in chronological order.

**Claim** *At any time, the number of players painting on any edge is weakly larger in $\Gamma$ than in $\tilde{\Gamma}$, except, perhaps, when that edge is completely painted in $\Gamma$.*

*Proof of the claim* At the beginning, i.e., in time segment $\tau_1$, weakly more people paint on any arc in $\Gamma$ than in $\tilde{\Gamma}$. Let $j$ be a natural number and suppose (in order to apply induction) that in all open time segments $\tau_1, \dots, \tau_j$ weakly more people paint on any arc in $\Gamma$ than in $\tilde{\Gamma}$, except for edges that have been painted completely in $\Gamma$.

Until the end of $\tau_j$, weakly more work has been done on each edge in the $\Gamma$-case than in the $\tilde{\Gamma}$-case. In particular, any edge finished by that time in $\tilde{\Gamma}$ is finished in $\Gamma$ as well. Let $e_{\mathbf{p}}$ be any edge not yet finished in both procedures at the end of $\tau_j$. We are going to compare for both instances and for each $\mathbf{q} \in V \backslash \{\mathbf{o}\}$, the number of people that $\mathbf{q}$ sends to $e_{\mathbf{p}}$ in period $\tau_{j+1}$ to work on. The summary of cases below shows that the more edges have been painted completely, the more people $\mathbf{q}$ sends:

– If $\mathbf{q} = \mathbf{p}$, then the representative of $\mathbf{p}$ works at $e_{\mathbf{p}}$ only if the road from $\pi(\mathbf{p})$ to the capital city has been painted completely. All (other) inhabitants of $\mathbf{q}$ work on $e_{\mathbf{p}}$.
– If $\mathbf{q} \succ \mathbf{p}$ (i.e., $\mathbf{p}$ is situated on the path from $\mathbf{q}$ to the capital city) and all edges between $\mathbf{q}$ and $\mathbf{p}$ have been painted completely, all inhabitants of $\mathbf{q}$ work on $e_{\mathbf{p}}$.
– If $\mathbf{q} \succ \mathbf{p}$ and all edges between $\mathbf{q}$ and $\mathbf{p}$ *but* $e_{\mathbf{q}}$ have been painted completely, only the representative of $\mathbf{q}$ works on $e_{\mathbf{p}}$.
– If $\mathbf{q} \succ \mathbf{p}$ and at least one of the edges between $\pi(\mathbf{q})$ and $\mathbf{p}$ has not been painted completely yet, no inhabitants nor the representative of $\mathbf{q}$ work on $e_{\mathbf{p}}$.
– Finally, if $\mathbf{q} \not\succeq \mathbf{p}$, no inhabitants of $\mathbf{q}$ work (ever) on $e_{\mathbf{p}}$.

Since at the end of period $\tau_j$ weakly more edges have been completed in $\Gamma$ than in $\tilde{\Gamma}$, village $\mathbf{q}$ sends weakly more people in the $\Gamma$-case. Since no people move from one edge to another during time segment $\tau_{j+1}$, this will be valid until the end of period $\tau_{j+1}$. By induction, it will be true until edge $e_{\mathbf{p}}$ will be finished completely in $\Gamma$. This proves the claim.

As a corollary of the claim we find that each edge is finished weakly earlier in $\Gamma$ than in $\tilde{\Gamma}$. We conclude the proof by stating that for all $p \in V$

$\mathrm{Nu}_{\mathbf{p}}$ with respect to $\Gamma$

$$\begin{aligned} &= \max\{t \mid \text{some edge between } \mathbf{p} \text{ and } \mathbf{o} \text{ is contracted at time } t \text{ in } \Gamma\} \\ &\leq \max\{t \mid \text{some edge between } \mathbf{p} \text{ and } \mathbf{o} \text{ is contracted at time } t \text{ in } \tilde{\Gamma}\} \\ &= \mathrm{Nu}_{\mathbf{p}} \text{ with respect to } \tilde{\Gamma} \end{aligned}$$

□

## References

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

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

Kohlberg E (1971) On the nucleolus of a characteristic function game. SIAM J Appl Math 20:62–65

Maschler M (1992) The bargaining set, kernel and nucleolus—a survey. In: Aumann RJ, Hart S (eds) Handbook of game theory with economic applications. Elsevier, North Holland, Vol 1, pp 591–667

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

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

Megiddo N (1974) On the nonmonotonicity of the bargaining set, the kernel and the nucleolus of a game. SIAM J Appl Math 27:355–358

Megiddo N (1978) Computational complexity of the game theory approach to cost allocation for a tree. Math Oper Res 3:189–196

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

Sobolev A (1975) A characterization of optimality principles in cooperative games by functional equations (Russian). Math Methods Soc Sci 6:94–151

Sönmez T (1994) Population monotonicity of the nucleolus on a class of public good problems. Mimeo, University of Rochester

Sprumont Y (1990) Population monotonic allocation schemes for cooperative games with transferable utility. Games Econ Behav 2:378–394

Thomson W (1983a) The fair division of a fixed supply among a growing population. Math Oper Res 8:319–326

Thomson W (1983b) Problem of fair division and the egalitarian solution. J Econ Theory 31:211–226