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

On the Core and Nucleolus of Directed Acyclic Graph Games

Balázs Sziklai · Tamás Fleiner · Tamás Solymosi

Received: date / Accepted: date

Abstract We introduce directed acyclic graph (DAG) games, a generalization of standard tree games, to study cost sharing on networks. This structure has not been previously analyzed from a cooperative game theoretic perspective. Every monotonic and subadditive cost game – including monotonic minimum cost spanning tree games – can be modeled as a DAG-game. We provide an efficiently verifiable condition satisfied by a large class of directed acyclic graphs that is sufficient for the balancedness of the associated DAG-game. We introduce a network canonization process and prove various structural results for the core of canonized DAG-games. In particular, we characterize classes of coalitions that have a constant payoff in the core. In addition, we identify a subset of the coalitions that is sufficient to determine the core. This result also guarantees that the nucleolus can be found in polynomial time for a large class of DAG-games.

Keywords Cooperative game theory · Core · Nucleolus · Directed acyclic graphs · Dually essential coalitions

B. Sziklai
Momentum Game Theory Research Group, Centre for Economic and Regional Studies, Hungarian Academy of Sciences and Corvinus University of Budapest, Department of Operations Research and Actuarial Sciences
E-mail: sziklai.balazs@krtk.mta.hu

T. Fleiner
Budapest University of Technology and Economics, Department of Computer Science and Information Theory
E-mail: fleiner@cs.bme.hu

T. Solymosi
Corvinus University of Budapest, Department of Operations Research and Actuarial Sciences, and MTA-BCE Momentum Research Group on Strategic Interactions
E-mail: tamas.solymosi@uni-corvinus.hu

1 Introduction

We model cost sharing on directed acyclic graphs (DAG) via a cooperative transferable utility game. We are interested in stable cost divisions. In particular, we focus on the computability of the core and a special core selection – the nucleolus.

Formally, we have a graph where nodes represent the residences of the players. There is a many-to-one correspondence between the players and the node set. Edges are directed toward a root and carry a non-negative construction cost. The aim of the players is to construct a cheapest directed subgraph that connects them to the root. The nodes are considered public, that is, any player can use any node to connect himself into the network. The possible ways of cooperation and the underlying bargaining process is captured by a transferable utility game.

A typical economic situation that can be modeled with DAG-games is the cost allocation of infrastructural projects. Consider for instance a group of towns that would like to connect themselves to a water reserve. Clearly, not every town has to build a direct pipeline to the source. A possible solution is to connect the nearest towns with each other and then one of the towns with the reserve. The towns that are already connected to the water system can force the rest to pay some of their construction cost, otherwise they can close down the outgoing water flow. On the other hand, no town can be forced to pay more than the cost of directly connecting itself to the water reserve.

This example can be retold in a minimum cost spanning tree (MCST) setting as well. However, if the reserve is on a hill or a mountain (which is quite possible) and the towns that have to be serviced are at different elevations then the DAG-framework can be more convincing. Especially, if we assume that pumping the water uphill is too costly for the towns.

There are many variants of this problem depending on the underlying graph structure (single path (Potters and Sudhölter, 1999), tree (Megiddo, 1978; Bjørndal et al, 2004; Maschler et al, 2010), complete graph (Bird, 1976; Granot and Huberman, 1981, 1984), etc.), on whether the nodes carry cost (or a reward) or not (Granot and Maschler, 1998; van den Nouweland et al, 1993), and on whether there are one or more sources (Rosenthal, 1987; Granot and Granot, 1992; Kuipers, 1997). Sziklai et al (2014) offers a detailed literature overview.

1.1 Significance of DAG-games

Standard tree and minimum cost spanning tree games as well as various other tree enterprises were proposed to analyze the same economic situation: How to share the incurring costs among costumers who would like to gain access to some public service or public facility and who are located in a predetermined spatial structure. Such situations arise frequently in the real world: Farmers share the maintenance costs of an irrigation system, workers share the cost of carpooling, and basically the cost sharing of any infrastructural projects that connects the users to a common source (electric power generators, water supply sources, sewage cleaning facilities, etc.) belongs to this family of problems.

Analyzing cost sharing situations on directed networks, where no cycle is allowed is seemingly a restriction compared to the undirected case. Surprisingly it is the other way around. In fact, *every monotonic, subadditive cots game can be written as a DAG-game* (Theorem 2). The corresponding mapping, however, can result in an exponentially large representation. DAG-games include among others: airport games, standard tree games and monotonic MCST-games (mMCST). In addition multiple source fixed tree games can also be converted into DAG instances by contracting the sources into a single node. Such conversions can have various benefits. In DAG-games the players are hierarchically structured which in some cases allows us to compute the nucleolus of the game in polynomial time or describe the core efficiently (Theorem 6). In comparison, finding the nucleolus of a (non-monotonic) MCST game is $\mathcal{NP}$-hard in general (Faigle et al, 1998)[1]. We will further discuss the significance of DAG-games in Section 4.

1.2 Our contribution

The core and special members of the core, such as the nucleolus are among the most popular solution concepts in cooperative game theory. Unfortunately, they suffer from high computational complexity. The core is described by exponentially many inequalities, and hard to test its membership (Faigle et al, 1997). Similarly, both the computation and verification of the nucleolus it is $\mathcal{NP}$-hard in the number of players (Deng et al, 2009). Consequently, there is a vast literature concerned with finding appropriate classes of games where these solutions can be determined efficiently.

DAG-games model several natural settings, thus, proving results for this game class has wide implications. The following points summarize our findings:

- We introduce a network canonization process which, if successfully carried out, admits a payoff allocation in the core. The condition related to this process is seemingly non-restrictive. Certainly, many naturally occurring networks do fall into this category.
- We identify 'free riders' i.e. players that do not pay anything in any core allocation. Players are basically free riders if there leads two paths from their residence to the root, such that these paths are disjoint (except maybe for some zero cost edges) and use only the cheapest edges leaving any node.
- We characterize coalitions that have constant zero excess in the core. This result sheds light on why the residences of the free riders can be contracted with the root without altering the core of the game.

[1] The proof of Faigle et al (1998) uses reduction to the minimum cover problem and relies heavily on the undirectedness of the edges of the graph. Thus, it does not necessarily indicate that the same difficulties exists in DAG-games. The difference between monotonic and ordinary MCST games can also be of importance here.

- We identify a characterization set: a set of coalitions whose excess determines the core and pinpoints the nucleolus. These so-called dually essential coalitions have nice graph structure. Any member of this family connects to the root via a trunk which has exactly one missing branch.
- We provide a simple condition which ensures that the core and the nucleolus of a fairly large class of DAG-games can be computed in polynomial time.

The first three structural results help us understand what happens in the core, and how to find a core member. The last two have more direct consequences and they can be applied as is in cost sharing problems.

Let us note that DAG-games and mMCST-games share a common difficulty: The computation of the cost of a coalition (i.e. finding the cheapest subnetwork that connects all players in the coalition to the root) is an $\mathcal{NP}$-hard problem. In the case of DAG games it amounts to solving the so-called acyclic directed Steiner tree problem – also known as the Steiner arborescence problem (Hwang et al, 1992). The computation of the entire cost function for all coalitions, therefore, could be prohibitive in practice (for alternative approaches see e.g. (Skorin-Kapov and Skorin-Kapov, 2012)). Unlike for mMCST (and standard tree) games, the core of the cost game associated to our standard DAG-network might be empty, so a stable solution of the cost allocation problem might not exist. In light of these difficulties the positive results of this paper are even more appealing.

2 Game theoretic framework

A *cooperative cost game* is an ordered pair (N, c) consisting of the player set $N = \{1, 2, \dots, n\}$ and a cost function $c : 2^N \to \mathbb{R}_+ \cup \{0\}$ with $c(\emptyset) = 0$. The value $c(S)$ represents how much cost coalition S must bear if it chooses to act separately from the rest of the players. We will denote a specific cost game by Γ. In most cases c is monotonic and subadditive. That is, the more people use the service the more it costs, however there is also an increase of efficiency. Formally, a cost game (N, c) is called *monotonic* if $(S, T \subseteq N, S \subset T) \Rightarrow c(S) \le c(T)$, *subadditive* if, $(S, T \subset N,\ S \cap T = \emptyset) \Rightarrow c(S) + c(T) \ge c(S \cup T)$, and *concave* if its characteristic function is submodular, i.e. if for all $S, T \subseteq N$, $c(S) + c(T) \ge c(S \cup T) + c(S \cap T)$ holds.

A solution for a cost game is a vector $x \in \mathbb{R}^N$. For ease of presentation, we introduce the following notation $x(S) = \sum_{i \in S} x_i$ for any $S \subseteq N$. A solution is called *efficient* if $x(N) = c(N)$ and *individually rational* if $x_i \le c(i)$ for all $i \in N$. The *imputation set* $I(\Gamma)$ of game Γ consists of the efficient and individually rational solutions.

An efficient payoff vector is also called an *allocation.* Given an allocation $x \in \mathbb{R}^N$, we define the *excess* of a coalition S as

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

The excess value $exc(S, x)$ measures the contentment of coalition S under allocation x. The *core* $\mathcal{C}(\Gamma)$ of cost game Γ is the set of allocations where all excesses are non-negative. Formally,

$$\mathcal{C}(\Gamma) = \{x \in \mathbb{R}^N \mid x(N) = c(N),\ x(S) \leq c(S) \text{ for all } S \subset N\}.$$

A game is called *balanced* if its core in non-empty.

We say that a vector $x \in \mathbb{R}^m$ *lexicographically precedes* $y \in \mathbb{R}^m$ (denoted by $x \preceq y$) if either $x = y$ or there exists a number $1 \leq j \leq m$ such that $x_i = y_i$ if $i < j$ and $x_j < y_j$. Let $\Gamma = (N, c)$ be a cost game and let $\theta(x) \in \mathbb{R}^{2^n-2}$ be the excess vector that contains all the excess values (except for the excesses of the grand coalition and the empty set) in a non-decreasing order.

Definition 1 The *nucleolus* of a cooperative game Γ is the subset of the payoff vectors $x \in \mathbf{I}(\Gamma)$ that lexicographically maximize $\theta(x)$. Formally,

$$\mathbf{N}(\Gamma) = \{x \in \mathbf{I}(\Gamma) \mid \theta(y) \preceq \theta(x)\ \forall y \in \mathbf{I}(\Gamma)\}.$$

Schmeidler (1969) proved that $\mathbf{N}(\Gamma)$ consists of a single point, and that it is a continuous function of the characteristic function. Although formally a set, we will consider $N(\Gamma)$ as an allocation vector. It is straightforward that $\mathbf{N}(\Gamma) \in \mathbf{C}(\Gamma)$ whenever $\mathbf{C}(\Gamma)$ is non-empty.

Maschler et al (1979) provided a general scheme for computing the nucleolus in a form of a sequential linear program. Although all the LPs in the sequence consist of exponentially many inequalities they can be solved efficiently if one knows which constraints are redundant. Huberman (1980); Granot et al (1998); Reijnierse and Potters (1998) and recently Solymosi and Sziklai (2016) provided methods to identify coalitions that correspond to non-redundant constraints. Here we focus on the following type of coalitions.

Definition 2 (Dually essential coalitions) Coalition S is called *dually inessential* in cost game $\Gamma = (N, c)$ if its complement can be partitioned as $N \setminus S = (N \setminus T_1) \dot\cup \ldots \dot\cup (N \setminus T_k)$ with $k \geq 2$ such that $c(S) \geq c(T_1) + \ldots + c(T_k) - (k-1)c(N)$. A coalition that is not dually inessential is called *dually essential.* The set of dually essential coalition in Γ is denoted by $\mathcal{DE}(\Gamma)$.

Dually essential coalitions form a characterization set for the nucleolus in balanced games (for details see (Solymosi and Sziklai, 2016)). The characteristic function values of these coalitions determine both the core and the nucleolus in any balanced game. Thus, if the number of dually essential coalitions is polynomially bounded in the number of players, then we can obtain a polynomial size description of the core and the nucleolus of the game can be computed in strongly polynomial time.

3 Definition and basic properties of DAG games

A directed acyclic graph network $\mathcal{D}$ or shortly a *DAG-network* is given by the following:

– $G(V, A)$ is a directed acyclic graph, with a special node - the so-called *root* of G, denoted by $\mathbf{r}$ - such that from every other node of G there leads at least one directed path to the root. G is assumed to be a simple graph, i.e. it has no loops or parallel arcs.
– There is a cost function $\delta : A \to \mathbb{R}^+ \cup \{0\}$ that assigns a non-negative real number to each arc. This value is regarded as the construction cost of the arc.

For a subgraph T, $V(T)$ denotes the node set of T. Similarly $A(T)$ denotes its arc set, while $A_{\mathbf{p}}$ is used for the set of arcs that leave node $\mathbf{p}$. We call nodes that have one leaving arc *passages*, while nodes that have more than one leaving arcs are called *junctions.* Zero cost arcs are abbreviated as *zero arcs.*

Let N be a set of players and let $\mathcal{R} : N \to V$ be the residency function that maps N to the node set of G. If player i is assigned to node $\mathbf{p}$ we say that player i *resides* at $\mathbf{p}$. A node is *occupied* if at least one player resides in it. Note that unoccupied leaves (i.e. nodes with no incoming arcs) are redundant and can be omitted from the network. The residency function is not assumed to be injective and/or surjective, but it is a proper function. It means that any one player resides at exactly one node, but there can be unoccupied nodes or nodes having more than one resident. The set of residents of a subgraph T is denoted by $N(T)$, formally, $N(T) = \mathcal{R}^{-1}(V(T))$. A network $\mathcal{D}$ together with a residency mapping $\mathcal{R}$ defined on $\mathcal{D}$ is called a *player network* and denoted by $(\mathcal{D}, \mathcal{R})$.

For a subgraph T, we define its construction cost $C(T)$ as the total cost of the arcs in T, i.e. $C(T) = \sum_{a \in A(T)} \delta(a)$. A path whose end point is the root is called a *rooted path.* A connected subgraph of G that is a union of rooted paths is called a *trunk.* For each coalition S, let T_S denote the set of trunks that have maximum number of arcs among the cheapest trunks that connect all players in S to the root. The maximality requirement may seem odd at first, but it is needed to ensure the uniqueness of T_N (cf. Section 5).

We say that a trunk T corresponds to a node set B if $V(T) = B$. Similarly we say that a coalition S corresponds to the trunk T if $T \in T_S$. Note that more than one coalition can correspond to the same trunk.

The characteristic function of the cost game that is associated with the player network $(\mathcal{D}, \mathcal{R})$ is defined as follows.

$$c_{(\mathcal{D},\mathcal{R})}(S) \overset{def}{=} C(T) \qquad T \in T_S.$$

The pair $(N, c_{(\mathcal{D},\mathcal{R})})$ is called a *DAG-game.* The definition of $c_{(\mathcal{D},\mathcal{R})}$ is motivated by the fact that by leaving the grand coalition the players in S need not pay more than $c_{(\mathcal{D},\mathcal{R})}(S)$ to get connected to the root. As any trunk in T_S has the same construction cost, $c_{(\mathcal{D},\mathcal{R})}(S)$ is well-defined. Note that finding these trunks may be computationally demanding. It is possible, however, to derive solutions for the game using the network structure, without generating the characteristic function.

It is straightforward to see that the characteristic function of any DAG-game is non-negative, monotone and subadditive (even strongly subadditive, i.e. $c(S) + c(T) \geq c(S \cup T)$ holds for any not necessarily disjoint coalitions S and T). On the other hand, Figure 1A shows an example when a stronger property, submodularity is not satisfied.

Let $S_1 = \{1, 3\}$ and $S_2 = \{2, 3\}$, then

$$3 + 2 = c(S_1) + c(S_2) < c(S_1 \cup S_2) + c(S_1 \cap S_2) = 4 + 2,$$

thus we conclude that DAG-games need not be concave.

The following example demonstrates that DAG-games need not even be balanced. Consider the player network $(\mathcal{D}, \mathcal{R})$ depicted in Figure 1B. The cost of connecting any two-player coalition is 3, however $c_{(\mathcal{D},\mathcal{R})}(N) = 5$ which leaves the core empty.

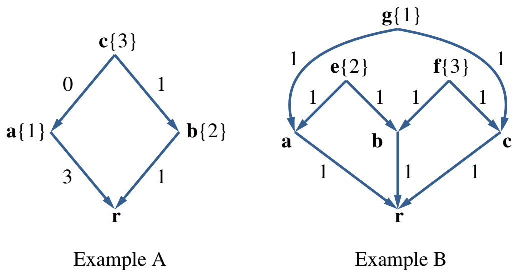

Fig. 1 The first example shows that the characteristic function need not be submodular. Example B displays a DAG-network that induces a cost game with an empty core. The residents of the nodes are given in braces in both cases.

The next theorem specifies the types of networks we are interested in.

Theorem 1 *The following condition*

(*) ***there must be a resident at each passage where the leaving arc is a zero arc and at each node with more than one entering arc and with leaving arc(s) all of positive cost***

is sufficient for a DAG-game to have a non-empty core.

The proof is immediate from Observation 1 and Lemma 2, that we discuss later. Although this property seems unrestrictive it is crucial in the applicability of our results. Let us stress, that dually essential coalitions only characterize the nucleolus in balanced games. Notice that property **(*)** can be checked efficiently. In the following we will assume that **(*)** holds for any $(\mathcal{D}, \mathcal{R})$ network.

4 More on the class of DAG-games

As we stated in the introduction, DAG-games include many important game classes, most notably the mMCST family. Some mMCST games can be converted into DAGs by choosing directions for the links appropriately. Figure

2 depicts a mMCST and a DAG-network that induce the same characteristic cost function. However – as the next example shows – not every mMCST-game can be converted into a DAG in such a straightforward fashion.

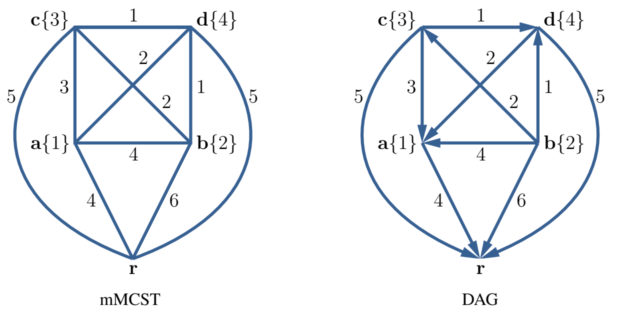

Fig. 2 A mMCST and a DAG-network that describe the same cooperative game.

Example 1 Consider the 3-player mMCST-game depicted in Figure 3. Notice that coalition $\{2,3\}$ uses the $\mathbf{b}-\mathbf{c}$ edge in different direction than the grand coalition.

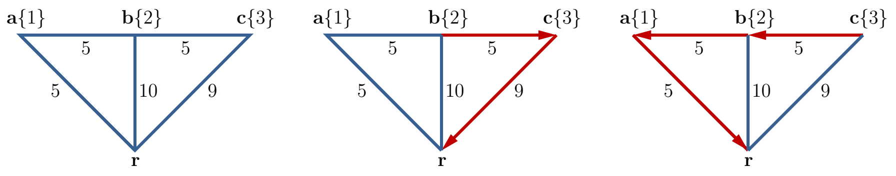

Fig. 3 A mMCST network and the cheapest subnetworks for coalitions $\{2,3\}$ and $\{1,2,3\}$.

Now we provide a construction that works for any monotonic, subadditive game. Let N be a player set and $\hat{c} : 2^N \to \mathbb{R}$ be a monotonic, subadditive cost function. We assign a node $\mathbf{p}_S$ for each coalition $S \subseteq N$ and introduce a new node $\mathbf{r}$, which will be the root of the graph. Each coalition has an arc a_S that leaves $\mathbf{p}_S$ and enters the root, such that $\delta(a_S) = \hat{c}(S)$. Furthermore, for each $i \in N$, there is a zero-arc, denoted by a_S^i that leaves $\mathbf{p}_{\{i\}}$ and enters $\mathbf{p}_{S\cup\{i\}}$ for all $\emptyset \neq S \subseteq N \setminus \{i\}$. Only the singleton coalitions are occupied, i.e. $\mathcal{R}(i) = \mathbf{p}_{\{i\}}$. We call the obtained player-network $(\mathcal{D}, \mathcal{R})$ the *characteristic DAG-representation of* $\hat{c}$.

Theorem 2 *Let N be a player set and $\hat{c} : 2^N \to \mathbb{R}$ a monotonic, subadditive cost function. There exists a DAG-network $\mathcal{D} = (G(V,A),\delta)$ and a residency mapping $\mathcal{R} : N \to V$ such that*

$$c_{(\mathcal{D},\mathcal{R})}(S) = \hat{c}(S) \quad \forall S \subseteq N.$$

The technical details of the proof can be found in the appendix. Figure 4 shows an example of a characteristic DAG-representation.

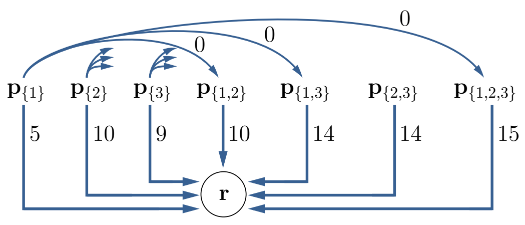

Fig. 4 The characteristic DAG-representation of the mMCST-game of Example 1. In order to retain transparency the leaving zero-arcs of $\mathbf{p}_{\{2\}}$ and $\mathbf{p}_{\{3\}}$ were shortened.

The characteristic DAG-representation has mostly theoretical value. In such networks there are many unoccupied passages with more than one entering arc, hence the **(*)** property is not satisfied. The other drawback is that it can inflate a game with n players, which has a nice DAG-representation of $\mathcal{O}(n)$ nodes and arcs (e.g. the representation of an airport or standard tree game), into a monster network of $\mathcal{O}(2^n)$ nodes and arcs.

As mMCST games are monotonic and subadditive, there indeed exists a DAG-representation for any of them. It can happen that the only DAG-representation of an mMCST-game is the characteristic representation where the **(*)** property is not satisfied. This does not contradict the fact that every mMCST game is balanced, since **(*)** is not a necessary condition for the non-emptyness of the core.

5 The canonization process and its consequences

In this section we will introduce a network canonization process which has two main advantages. Firstly it makes the structure more transparent, allows us to define concepts like 'principal ancestor' (cf. Definition 3) in a simpler way. Secondly it makes the trunk of the grand coalition (T_N) unique. Again this will tremendously help to understand and work with DAG-networks.

We say that a player network $(\mathcal{D}, \mathcal{R})$ is in canonical form if the following properties are fulfilled:

P1 Each junction has a leaving zero arc: $\mathbf{p} \in V,\ |A_{\mathbf{p}}| > 1 \Rightarrow \exists a \in A_{\mathbf{p}}$ such that $\delta(a) = 0$.
P2 For each passage the cost of the leaving arc is positive: $\mathbf{p} \in V,\ A_{\mathbf{p}} = \{a\} \Rightarrow \delta(a) > 0$.
P3 There resides a player in each passage: $\mathbf{p} \in V,\ A_{\mathbf{p}} = \{a\} \Rightarrow$
$\exists i \in N$ such that $\mathcal{R}(i) = \mathbf{p}$.

To transform a DAG-game into a form where property **P1** is fulfilled we have to perform the following procedure for each node $\mathbf{p} \in V$ such that $|A_{\mathbf{p}}| \geq 2$ and $\min_{a \in A_{\mathbf{p}}} \delta(a) = \alpha_{\mathbf{p}} > 0$.

1. Introduce an unoccupied new node $\mathbf{p}'$ with the same set of leaving arcs as $\mathbf{p}$ has, but reduce the cost of the arcs by $\alpha_{\mathbf{p}}$.

2. Erase all the arcs that leave $\mathbf{p}$.
3. Finally, introduce a new arc from $\mathbf{p}$ to $\mathbf{p}'$ with cost $\alpha_{\mathbf{p}}$.

Property **P2** can be achieved by contracting each passage that has a leaving zero arc with the endnode of that arc, by uniting the resident sets of the contracted nodes, and by eliminating that zero arc. Obtaining both **P1** and **P2** require equivalent transformations in the sense that the construction cost of the trunks in T_S is unchanged for any coalition S.

Finally, if $\mathbf{p}$ is an unoccupied passage and $\mathbf{p}$ has only one entering arc then it can be omitted from the network. The entering and leaving arc of $\mathbf{p}$ can be replaced by a single arc with the aggregated construction cost. Needless to say that this procedure does not change the costs of the T_S trunks either. Note that if a passage has more than one entering arc then by property **(*)** it is occupied.

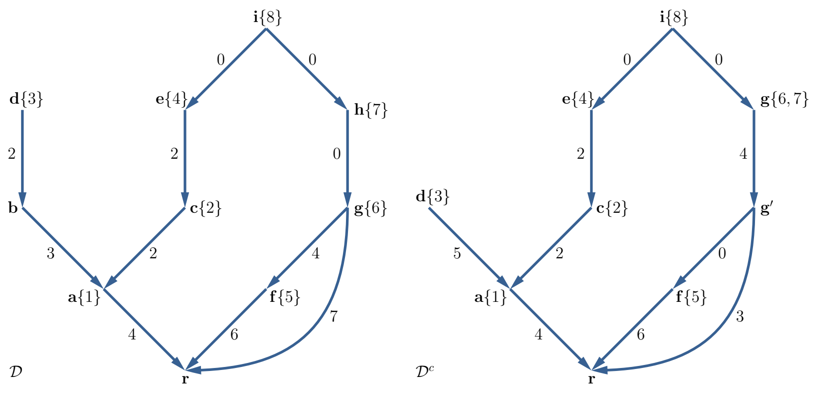

Fig. 5 A DAG-network with player set $N = \{1, 2, 3, 4, 5, 6\}$ before and after canonization. Notice that after canonization $\mathbf{g}'$ has a shortcut to the root.

Example 2 Let us demonstrate the canonization process on an example. Consider the player-network depicted in Figure 5. There are two junctions $\mathbf{g}$ and $\mathbf{i}$. The former does not have a leaving zero cost arc. Hence we introduce a new node $\mathbf{g}'$, connect $\mathbf{g}'$ with the neighbours of $\mathbf{g}$, delete the leaving arcs of $\mathbf{g}$ and finally connect $\mathbf{g}$ and $\mathbf{g}'$. The new arcs are given appropriate weights. As $\mathbf{h}$ is a passage with a zero cost arc, we contract it with node $\mathbf{g}$. The player sets of the two nodes are united. Node $\mathbf{b}$ is an unoccupied passage, thus we delete this node. We connect $\mathbf{d}$ with $\mathbf{a}$ directly. The new arc has the aggregated cost of the deleted entering and leaving arcs of $\mathbf{b}$. This concludes the three steps of canonization.

Our first observation summarizes the above findings.

Observation 1

– *All networks that satisfy* **(*)** *can be canonized.*
– *The characteristic function is unaffected by the canonization process.*

Although canonization ensures that T_N contains only a single element, this cannot be said in general about the trunks of other coalitions. In the following we will assume that T_S contains only a single trunk for any coalition S. This can always be achieved by perturbing the positive arc costs. The most important solution concepts, such as the core, the nucleolus and the Shapley-value are all continuous functions of the characteristic function, which in turn is a continuous function of the arc costs. Thus, such perturbation does not affect the outcome of the game. Henceforward we will refer to T_S as this unique trunk that has maximum number of arcs among the cheapest trunks that connect all members of S to the root.

From now on we will drop the residency function $\mathcal{R}$ from the notation and simply write $c_{\mathcal{D}}$. We will denote by $\varGamma_{\mathcal{D}}$ the cost game induced by $c_{\mathcal{D}}$, i.e. $\varGamma_{\mathcal{D}} = (N, c_{\mathcal{D}})$. For sake of simplicity, we will also call a DAG-game on a canonized player network as a *canonized DAG-game*. Let us now see some consequences of canonization. We also need to introduce further notions and notation.

For each node $\mathbf{p}$, the cheapest arcs in $A_{\mathbf{p}}$ are called T_N-arcs. The name comes from the fact that an arc is a T_N-arc if and only if it is an element of $A(T_N)$. If $a, a' \in A_{\mathbf{p}}$, a is a T_N-arc and $\delta(a') > \delta(a)$, then a' is called a *shortcut*. Thus every arc that is not a T_N-arc is a shortcut. If two nodes $\mathbf{p}$ and $\mathbf{q}$ are connected with a shortcut then either it is the cheapest route between these two nodes (hence the name) or it is so costly that no coalition uses it at all. In this latter case the arc can be omitted from the network. If $a, a' \in A_{\mathbf{p}}$ are T_N-arcs then the construction cost of both a and a' is zero (this is a consequence of **P1**).

The subgraph associated to the grand coalition (T_N) holds special importance. First this is the graph that will be constructed in the end. All the other arcs are only good for improving the bargaining positions of certain players[2]. Note that T_N is not necessarily a tree as it may contain some additional zero arcs. Unlike other trunks, T_N can be constructed efficiently in linear time by building the cheapest arc(s) leaving each node. The connection cost of any occupied node is at least as much as the cost of the cheapest arc that leaves that node. Furthermore, every unoccupied node has a leaving zero arc, therefore connecting an unoccupied node does not impose extra cost. Thus, including the cheapest arcs from each node connects all nodes to the root. It follows that $V(T_N) = V$ and $E(T_N)$ contains every arc that is not a shortcut.

Another feature of T_N is that it induces a partial order $\prec$ on the nodes. We say that $\mathbf{p}$ is a *ancestor* of $\mathbf{q} \neq \mathbf{p}$ if $\mathbf{p}$ can be reached from $\mathbf{q}$ via a path in T_N, we denote this by $\mathbf{p} \prec \mathbf{q}$. In such cases we also say that $\mathbf{q}$ is an *descendant* of $\mathbf{p}$. Node $\mathbf{p}$ is a *direct ancestor* or *parent* of $\mathbf{q}$ if $\mathbf{p}$ is an ancestor of $\mathbf{q}$ and they are connected with a T_N-arc. This relation is denoted by $\pi(\mathbf{q})$ whenever

[2] Since we analyze the game from a cooperative perspective, we do not formalize the bargaining process that takes place between the players. However, such processes are certainly present. Consider the case of Example 2. Without the shortcut of $\mathbf{g}'$ player 6 may be forced to pay as much as 10 in the core, while with the shortcut present his core payoff is bounded from above by 7.

the direct ancestor is unique. If $\mathbf{p}$ is a parent of $\mathbf{q}$ then $\mathbf{q}$ is referred as a *direct descendant* or *child* of $\mathbf{p}$. The node set that contains $\mathbf{p}$ together with its descendants is called a *full branch* and denoted by $B_{\mathbf{p}}$.

Sometimes we are interested only in some of the descendants of $\mathbf{p}$, therefore we cut off some segments of $B_{\mathbf{p}}$. Removing a node from $B_{\mathbf{p}}$ other nodes can become unreachable too. A specific *branch*, denoted by $B_{\mathbf{p}}^{Q}$ is a subset of $B_{\mathbf{p}}$ that collects nodes that still can reach $\mathbf{p}$ using only T_N-arcs after removing the node set Q from $B_{\mathbf{p}}$. Formally

$$B_{\mathbf{p}}^{Q} \overset{def}{=} \{\mathbf{q} \in B_{\mathbf{p}} \mid \exists\, P_{\mathbf{q}-\mathbf{p}} \text{ such that } V(P_{\mathbf{q}-\mathbf{p}}) \subset B_{\mathbf{p}} \setminus Q\},$$

where $P_{\mathbf{q}-\mathbf{p}}$ denotes a path in T_N that leads from $\mathbf{q}$ to $\mathbf{p}$. In other words a branch is the node set of a union of paths in T_N which have a common origin. To emphasize this a $B_{\mathbf{p}}^{Q}$ branch is also called a $\mathbf{p}$-*branch*. Note that if $B_{\mathbf{p}}^{Q} = B_{\mathbf{p}}^{Q'}$ then $B_{\mathbf{p}}^{Q \cap Q'}$ define the same node set as well. We say that the $B_{\mathbf{p}}^{Q}$ branch is in *standard form* if the cardinality of Q is minimal, in other words if there exists no Q' such that $B_{\mathbf{p}}^{Q'} = B_{\mathbf{p}}^{Q}$ and $|Q'| < |Q|$.

We say that the node set B is *proper* if deleting B from G along with all of its entering and leaving arcs the root can still be reached on a directed path from any of the remaining nodes (i.e. the remaining graph is a trunk).

Let us illustrate the above introduced notions with some examples. Consider again the canonized DAG-network $\mathcal{D}^c$ depicted in Figure 5. The only shortcut in $\mathcal{D}^c$ is the one that connects node $\mathbf{g}'$ with the root. All the other arcs are T_N-arcs. Nodes $\mathbf{a}$ and $\mathbf{d}$ form a branch in $\mathcal{D}^c$, but this branch is not proper as without $\mathbf{a}$, node $\mathbf{c}$ gets disconnected. On the other hand node $\mathbf{f}$ composes a proper branch in itself. Finally, the node set that corresponds to the trunk $T_{\{3,6,8\}}$ is $V \setminus (B_{\mathbf{c}}^{\mathbf{i}} \cup B_{\mathbf{f}}^{\mathbf{g}'}) = \{\mathbf{a}, \mathbf{d}, \mathbf{g}', \mathbf{g}, \mathbf{i}\}$ and $c_{\mathcal{D}^c}(\{3,6,8\}) = 16$.

This last example raises a question. Can we obtain every trunk by removing some branches from the graph? By Lemma 1 the answer is positive.

Lemma 1 *The node set of every trunk that corresponds to a coalition $S \subset N$ can be obtained by deleting some branches from V. The removed branches can be chosen in such way that each of them originates from a passage. Formally for any $S \subset N$ there exists $Q_1, \dots, Q_k \subset V$ and $\mathbf{p}_1, \dots, \mathbf{p}_k \in V$ such that*

$$V(T_S) = V \setminus \cup_{j=1}^{k} B_{\mathbf{p}_j}^{Q_j},$$

where $\mathbf{p}_j$ is a passage for all $j \in \{1, 2, \dots, k\}$.

A proof is provided in the appendix. The obtained $V \setminus \cup_{j=1}^{k} B_{\mathbf{p}_j}^{Q_j}$ expression is called a *standard representation* of $V(T_S)$, if the redundant nodes have been removed from the Q_j sets, i.e. each $B_{\mathbf{p}_j}^{Q_j}$ branch is in standard form. A trunk can have more than one standard representation. Also notice that the $B_{\mathbf{p}_j}^{Q_j}$ branches may not be disjoint. For instance in Example 2 the trunk of coalition $\{1, 2, 3, 5\}$ corresponds to $V \setminus (B_{\mathbf{e}} \cup B_{\mathbf{g}})$.

6 Structure of the core

The main results of the paper are divided into two sections. In the first section we present the structural results. We show that canonized DAG-games are balanced. We identify nodes that behave like the source. Their residents do not invest in the construction of the network. Moreover if a set of players reaches such a secondary source – which we will call a free node – they do not participate in the construction anymore. That is, the residents of a branch that originates from a free node need not pay more in the core than the construction cost of the branch itself. Uncovering the free nodes has many advantages. One of them is – as we will see in the next section – that the free nodes can be contracted with the root. Although this transformation changes the characteristic function of the game, it does not affect the core or the nucleolus.

The following extension of the cost function will be needed. We define $\tau(Q,S)$ as the cost of the arcs in T_S that go out from node set Q, i.e.

$$\tau(Q,S) \overset{def}{=} \sum_{a\in(\cup_{q\in Q}A_q)\cap A(T_S)} \delta(a).$$

We define the *standard allocation* $\hat{x}$ of $\Gamma_{\mathcal{D}}$ as follows. For each player $i \in N$ let $\hat{x}(i) = \frac{\delta(a_{\mathbf{p}})}{|N(\mathbf{p})|}$ where $i \in N(\mathbf{p})$ and $a_{\mathbf{p}}$ is one of the leaving T_N-arcs of $\mathbf{p}$. For instance in the canonized network of Example 2 $\hat{x}(6) = \hat{x}(7) = 2$ and $\hat{x}(8) = 0$. In our first lemma we show that the standard allocation is a core selector.

Lemma 2 *$\mathcal{C}(\Gamma_{\mathcal{D}}) \neq \emptyset$ for any DAG-network $\mathcal{D}$ in canonical form.*

Proof We will use the standard allocation. The residents of a junction do not have to pay according to the standard allocation, while the residents of a passage pay for the construction cost of the T_N-arc that leaves the passage. Since unoccupied nodes can only be junctions, it follows that for any $B \subseteq V$,

$$\hat{x}(N(B)) = \tau(B,N). \tag{1}$$

That is, the players of B pay for all the arcs that leave B in T_N. This means, in particular, that $\hat{x}(N) = \tau(V,N) = c_{\mathcal{D}}(N)$. On the other hand, for any $S \subseteq N$

$$\begin{aligned}\hat{x}(S) = \sum_{\mathbf{p}\in\mathcal{R}(S)} |S\cap N(\mathbf{p})| \cdot \frac{\delta(a_{\mathbf{p}})}{|N(\mathbf{p})|} \leq \sum_{\mathbf{p}\in\mathcal{R}(S)} \delta(a_{\mathbf{p}}) \leq \\ \leq \sum_{\mathbf{p}\in V(T_S)} \delta(a_{\mathbf{p}}) \leq C(T_S) = c_{\mathcal{D}}(S),\end{aligned}$$

where $\mathcal{R}(S) = \{\mathcal{R}(i) : i \in S\}$. The last inequality holds, because $\sum_{\mathbf{p}\in V(T_S)} \delta(a_{\mathbf{p}})$ collects the cost of the cheapest arcs of each node in T_S, but $A(T_S)$ may contain shortcuts as well.

The standard allocation is just a weighted version of the Bird-rule which was proposed for MCST games (Bird, 1976). There is an extensive literature devoted to the various solution concepts of MCST games. Without attempting to be comprehensive we refer the reader to (Bergantiños and Vidal-Puga, 2007; Bogomolnaia and Moulin, 2010; Trudeau, 2012).

Notice that by monotonicity of the characteristic function, core vectors are non-negative. Indeed, $x_i = x(N) - x(N \setminus i) \geq c_{\mathcal{D}}(N) - c_{\mathcal{D}}(N \setminus i) \geq 0$ for any $i \in N$ and $x \in \mathcal{C}(\Gamma_{\mathcal{D}})$.

The following definitions will be useful. We say that node $\mathbf{q}$ is a *key ancestor* of node $\mathbf{p}$, if there are two paths in T_N from $\mathbf{p}$ to $\mathbf{q}$ such that these paths are arc-disjoint except maybe for some zero arcs (*semi-arc-disjoint* from now on). The degenerate cases when these two paths partially or (completely) coincide are also included in this definition. Thus if there exists a zero cost path from $\mathbf{p}$ to $\mathbf{q}$ then $\mathbf{q}$ is a key ancestor of $\mathbf{p}$. Clearly, each junction has at least one key ancestor. On the other hand, by property **P2**, a passage could not have a key ancestor, so we define the only key ancestor of a passage to be itself. For similar reasons we define the root to be the key ancestor of itself.

Definition 3 The *principal ancestor* of node $\mathbf{p}$ is a unique node $\mathbf{q} \in V$, denoted by $\Pi(\mathbf{p})$ that is a key ancestor of $\mathbf{p}$ and $\mathbf{q} \prec \mathbf{q}'$ for every other key ancestor $\mathbf{q}'$ of $\mathbf{p}$ (i.e. the key ancestor closest to the root).

Notice that a junction cannot be a principal ancestor of any of its descendants. The only principal ancestor that is not a passage is the root. Note also, that principle ancestors can be identified easily (an efficient algorithm is provided in (Sziklai et al, 2014)).

Definition 4 We say that an occupied node $\mathbf{p}$ is *free* if $x(N(\mathbf{p})) = 0$ for any core element x, i.e. the residents of $\mathbf{p}$ do not have to pay to get connected to the root. An unoccupied node $\mathbf{p}$ is called free if $\Pi(\mathbf{p}) = \mathbf{r}$. The set of free nodes is denoted by F.

Note that if $\mathbf{p}$ is a passage then the standard allocation would assign positive value to $N(\mathbf{p})$. In other words – with the exception of the root – every free node is a junction.

Example 3 Consider the DAG-network depicted in Figure 6. The nodes $\mathbf{a}, \mathbf{b}, \mathbf{d}, \mathbf{e}, \mathbf{f}$ and $\mathbf{h}$ are passages. Thus the key and principal ancestors of these nodes are themselves. Node $\mathbf{c}$ has two (semi-)arc-disjoint paths in T_N that enter the root, hence the principal ancestor of $\mathbf{c}$ is the root. Although there exists two paths from $\mathbf{g}$ to $\mathbf{a}$, the latter node is not a key ancestor of $\mathbf{g}$, since one of the paths does not belong to T_N. The key ancestors of $\mathbf{g}$ are $\mathbf{g}$ and $\mathbf{e}$. Among these two nodes $\mathbf{e}$ is closer to the root, hence $\mathbf{e}$ is the principal ancestor of $\mathbf{g}$. Finally, the principal ancestor of $\mathbf{i}$ is $\mathbf{d}$. Note, that if $\mathbf{d}$ was a junction, the principal ancestor of $\mathbf{i}$ would be the root.

In our next theorem we will characterize the set of free nodes. Before we proceed let us state a simple lemma that will play a crucial role in the proof.

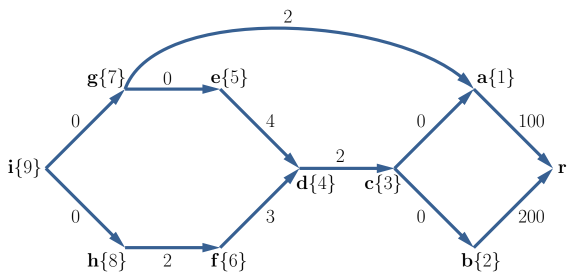

Fig. 6 A canonized network with nine players. Note that aside for the root node **c** is free as well.

Lemma 3 *Let $B^Q_{\mathbf{p}}$ be any branch originating from node* **p**. *If, for every core allocation y, $exc(N(V \setminus B_{\mathbf{p}}), y) = 0$, then $y(N(B^Q_{\mathbf{p}})) \leq \tau(B^Q_{\mathbf{p}}, N)$. In other words the residents of $B^Q_{\mathbf{p}}$ do not pay more than the costs of their T_N-arcs.*

Proof We proceed by contradiction. Suppose for some $y \in \mathcal{C}(\Gamma)$, $y(N(B^Q_{\mathbf{p}})) > \tau(B^Q_{\mathbf{p}}, N)$, then

$$c_{\mathcal{D}}(N((V \setminus B_{\mathbf{p}}) \cup B^Q_{\mathbf{p}}))) = c_{\mathcal{D}}(N(V \setminus B_{\mathbf{p}})) + \tau(B^Q_{\mathbf{p}}, N)$$
$$exc(N((V \setminus B_{\mathbf{p}}) \cup B^Q_{\mathbf{p}}))), y) = 0 + \tau(B^Q_{\mathbf{p}}, N) - y(N(B^Q_{\mathbf{p}})) < 0$$

would contradict the non-negativity of excesses.

In other words, since the residents of $V \setminus B_{\mathbf{p}}$ always pay for their own construction cost, and connecting $B^Q_{\mathbf{p}}$ to $V \setminus B_{\mathbf{p}}$ incurs at most $\tau(B^Q_{\mathbf{p}}, N)$ extra cost, the residents of $B^Q_{\mathbf{p}}$ are not willing to pay more than that in any core allocation.

The residents of the root or players that can reach the root via a zero cost path do not pay anything in the core. It is natural to assume that any other player must contribute to the construction costs, or at least there exists a core allocation when the payoff of such player is positive. Surprisingly this assertion is wrong. In Example 3 players 1 and 2 must bear huge costs, since connecting nodes **a** and **b** is an expensive task. However, neither of them can force player 3 to pay some of their construction costs, at least not in the core. As it will follow from our next theorem, node **c** is free, hence the payoff of player 3 is zero in any core allocation.

Theorem 3 *Node* **p** *belongs to F if and only if $\Pi(\mathbf{p}) = \mathbf{r}$.*

Proof If **p** is unoccupied we have nothing to prove, therefore we may assume that $|N(\mathbf{p})| > 0$.

First we prove the *only if* part. Suppose **p** is a free node but its principal ancestor **q** is a passage. We modify the standard allocation in the following way. Let $i_{\mathbf{p}}$ be a resident of **p** and $i_{\mathbf{q}}$ be a resident of **q** and let

$$y(i_{\mathbf{p}}) = \varepsilon,$$
$$y(i_{\mathbf{q}}) = \hat{x}(i_{\mathbf{q}}) - \varepsilon,$$
$$y(j) = \hat{x}(j) \text{ for any other player } j \in N,$$

where $\varepsilon > 0$ is a sufficiently small real number (e.g. $\varepsilon = \frac{\min_{a\in A}\delta(a)}{2|N|+1}$). Note that $\hat{x}(i_{\mathbf{q}}) > 0$ due to **P2**. We prove that $y \in \mathcal{C}(\Gamma_{\mathcal{D}})$. If S is such that $i_{\mathbf{p}}, i_{\mathbf{q}} \in S$ then $y(S) = \hat{x}(S)$. If $i_{\mathbf{p}} \notin S \ni i_{\mathbf{q}}$ then $y(S) < \hat{x}(S)$. The only interesting case is when $i_{\mathbf{p}} \in S \not\ni i_{\mathbf{q}}$. If $a_{\mathbf{q}} \in A(T_S)$ then

$$y(S) = y(S \setminus i_{\mathbf{p}}) + \varepsilon \le \hat{x}(S \setminus i_{\mathbf{p}}) + \hat{x}(i_{\mathbf{q}}) \le \hat{x}(S \cup N(\mathbf{q})) \le c_{\mathcal{D}}(S \cup N(\mathbf{q})) = c_{\mathcal{D}}(S),$$

where we used that $\varepsilon < \hat{x}(i_{\mathbf{q}})$. The last equality comes from the fact that $N(\mathbf{q})$ can join S for free as S builds $a_{\mathbf{q}}$ anyway. If $a_{\mathbf{q}} \notin A(T_S)$ then there is at least one shortcut in T_S. Let this shortcut be a'. Then

$$y(S) = y(S \setminus i_{\mathbf{p}}) + \varepsilon = \hat{x}(S \setminus i_{\mathbf{p}}) + \varepsilon \le \tau(\mathcal{R}(S), N) + \delta(a') \le c_{\mathcal{D}}(S),$$

where we used that $\hat{x}(S \setminus i_{\mathbf{p}}) \le \tau(\mathcal{R}(S), N)$ by (1). The last inequality is obviously true since apart from the cheapest arcs that leave $\mathcal{R}(S)$, the members of S need to build at least one shortcut, namely a'. We cannot overestimate the costs as the cheapest arc that leave the origin of a' – the cost of which is included in $\tau(\mathcal{R}(S), N)$ – is a zero arc due to **P1**.

To justify the other direction we prove that if $\Pi(\mathbf{p}) = \mathbf{r}$ then $\mathbf{p}$ is free and $exc(N(V \setminus B_{\mathbf{p}}), x) = 0$ for any $x \in \mathcal{C}(\Gamma_{\mathcal{D}})$. This is a slightly stronger statement from which the *if* part of the proof clearly follows.

Let $d(\mathbf{q})$ denote the length of the shortest path in T_N leading from $\mathbf{q}$ to $\mathbf{r}$. We proceed by induction on $d(\mathbf{p})$. If $d(\mathbf{p}) = 0$ then $\mathbf{p}$ is the root for which $exc(N(V \setminus B_{\mathbf{r}}), x) = exc(\{\emptyset\}, x) = 0$ is satisfied. Let us assume that $d(\mathbf{p}) = l$ and the lemma is true for any node $\mathbf{p}'$ with $d(\mathbf{p}') < l$ where $l > 0$ integer. Two cases are possible. The first is when $\mathbf{p}$ has a free parent $\mathbf{f}$. Let y be an arbitrary core element. Applying the induction step we obtain $exc(N(V \setminus B_{\mathbf{f}}), y) = 0$. Both $\mathbf{p}$ and $\mathbf{f}$ are junctions therefore $c_{\mathcal{D}}(N(V \setminus B_{\mathbf{f}})) = c_{\mathcal{D}}(N((V \setminus B_{\mathbf{f}}) \cup \{\mathbf{p}\}))$. Hence $y(N(\mathbf{p})) > 0$ would imply $exc(N((V \setminus B_{\mathbf{f}}) \cup \{\mathbf{p}\}), y) < 0$ – a contradiction.

The second case is when $\mathbf{p}$ does not have a free parent. As the principal ancestor of $\mathbf{p}$ is the root, there exists paths from $\mathbf{p}$ to $\mathbf{r}$ in T_N which are semi-arc-disjoint. There may be some intermediary nodes that coincide on these paths. Let the first such node denoted by $\mathbf{f}$. Note that the principal ancestor of $\mathbf{f}$ is the root ($\mathbf{f}$ may be the root itself) therefore we can apply the induction step. That means that $\mathbf{f}$ is free and $exc(N(V \setminus B_{\mathbf{f}}), y) = 0$ for any core allocation y. This also implies that $\tau(B_{\mathbf{f}}, N) = y(N(B_{\mathbf{f}}))$.

There exists two arc-disjoint path from $\mathbf{p}$ to $\mathbf{f}$ in T_N. Let $\mathbf{q}_1$ and $\mathbf{q}_2$ be the direct ancestors of $\mathbf{p}$ that lie on these paths. We can separate the node set $B_{\mathbf{f}} \setminus B_{\mathbf{p}}$ into two $\mathbf{f}$-branch B_1 and B_2 such that $\mathbf{q}_1 \in B_1$, $\mathbf{q}_2 \in B_2$ and $B_1 \cap B_2 = \{\mathbf{f}\}$. For instance such a partition can be obtained by coloring the path from $\mathbf{q}_1$ to $\mathbf{f}$ red and the path from $\mathbf{q}_2$ to $\mathbf{f}$ blue (as $\mathbf{f}$ is contained in both paths we can pick either one of the colors, say red). Then we color each node one-by-one in $B_{\mathbf{f}} \setminus B_{\mathbf{p}}$ in the following way. Take a direct descendant of a colored node. If it has a red parent we paint it red, if it has a blue one we paint it blue. If it has both a red and a blue parent paint it arbitrarily with one color. Let B_1 contain the red nodes, while B_2 the blue ones in addition with $\mathbf{f}$. Indeed the node sets defined in this way are $\mathbf{f}$-branches which satisfy $B_1 \cup B_2 = B_\mathbf{f} \setminus B_\mathbf{p}$ and $B_1 \cap B_2 = \{\mathbf{f}\}$. This leads us to

$$y(N(B_\mathbf{f})) = \tau(B_\mathbf{f}, N) = \tau(B_1 \cup B_\mathbf{p}, N) + \tau(B_2, N)$$
$$[\tau(B_1 \cup B_\mathbf{p}, N) - y(N(B_1 \cup B_\mathbf{p}))] + [\tau(B_2, N) - y(N(B_2)))] = 0.$$

We implicitly used that $\mathbf{f}$ is a junction, therefore its cheapest arc is a zero arc. Furthermore, $y(N(\mathbf{f})) = 0$ since $\mathbf{f}$ is free. Therefore it implies no additional cost that both B_1 and B_2 contain $\mathbf{f}$. The node set $B_1 \cup B_\mathbf{p}$ is an $\mathbf{f}$-branch and so is B_2, therefore the sums in the square brackets are non-negative by Lemma 3. It follows that $\tau(B_2, N) = y(N(B_2))$. Now let us move the $B_\mathbf{p}$ branch from B_1 to B_2. With exactly the same argument we can show that $\tau(B_1, N) = y(N(B_1))$. The coalitions $N(B_1)$ and $N(B_2)$ pay only for their own branch's construction cost i.e. the cost of the cheapest arcs that leave the B_1 and B_2 branch. From $N(B_\mathbf{p}) = N(B_\mathbf{f}) \setminus N(B_1 \cup B_2)$ it follows that $exc(N(V \setminus B_\mathbf{p}), y) = 0$. By Lemma 3 the $B_\mathbf{p}^Q$ branch pays at most $\tau(B_\mathbf{p}^Q, N)$ for any $Q \subset B_\mathbf{p}$. In particular $y(N(\mathbf{p})) = 0$ for any core element y, that is, $\mathbf{p}$ is free. This concludes the proof of Theorem 3.

A coalition S is said to be *saturated* if $c(S) = c(S \cup \{i\})$ implies $i \in S$. In other words if S is a saturated coalition then every new member will impose extra cost on the coalition. Let $\mathcal{S}^*(\mathit{\Gamma})$ denote the set of all saturated coalitions and let

$$\mathcal{S}(\mathit{\Gamma}) = \mathcal{S}^*(\mathit{\Gamma}) \cup \{N \setminus i \mid i \in N\} \cup \{N\}.$$

Granot, Granot and Zhu proved that the collection $\mathcal{S}(\mathit{\Gamma})$ characterizes the nucleolus of any monotone cost game (Granot et al, 1998). Moreover the efficiency equation $x(N) = c(N)$ and the $x(S) \leq c(S)$ inequalities corresponding to the saturated coalitions determine the core of such games as well. In light of these results we may restrict our attention to saturated coalitions. In case of DAG-games this property comes with a nice structure. A saturated coalition incorporates every player of the trunk on which it resides, formally S is saturated if and only if $S = N(V(T_S))$.

There are many coalitions whose excess is zero in any core allocation. For instance it is easy to prove that if $\mathbf{p}$ is a passage that is a direct descendant of the root, then $N(B_\mathbf{p})$ is such a coalition. In the following we characterize the set of saturated coalitions that bear this property. Let $\mathcal{S}_0$ denote the set of saturated coalitions whose excess is zero for any core allocation, formally

$$\mathcal{S}_0(\mathit{\Gamma}) \overset{def}{=} \{S \subseteq N \mid S \text{ saturated and } c(S) = x(S) \text{ for any } x \in \mathcal{C}(\mathit{\Gamma})\}.$$

In our next lemma we identify certain branches that pay only for their own construction cost i.e. the cost of the cheapest arcs that leave the branch. A $B_\mathbf{p}^Q$ branch is called a *building block* if it has the following properties:

– $\mathbf{p}$ is a passage whose parent is free,
– all the nodes in Q are free,
– $B^Q_{\mathbf{p}}$ does not contain a free node.

The proofs of the following two lemmata can be found in the appendix.

Lemma 4 *If $B^Q_{\mathbf{p}}$ is a building block, then $x(N(B^Q_{\mathbf{p}})) = \tau(B^Q_{\mathbf{p}}, N)$ for any core allocation x.*

For instance in Example 3 the branch $B_{\mathbf{d}}$ composes a building block. The next lemma tells us how to decompose certain branches into building blocks and free nodes.

Lemma 5 *Let $\cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$ be a union of branches such that $\mathbf{p}_j$ is a passage, $\pi(\mathbf{p}_j) \in F$ and $F_j \subset F$ for $j = 1, \dots, k$. Then $\cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$ can be decomposed into a disjoint union of building blocks and free nodes.*

In Example 3 the branch $B_{\mathbf{a}}$ can be decomposed, since $\mathbf{a}$ is a passage and its parent is free. The decomposition involves two building blocks $B^{\mathbf{c}}_{\mathbf{a}}, B_{\mathbf{d}}$ and a free node $\mathbf{c}$.

These seemingly technical results will prove to be quite useful. First with the help of these two lemmata we can characterize $\mathcal{S}_0(\Gamma)$. Secondly we will see that the residents of $B^{\mathbf{c}}_{\mathbf{a}} = \{a\}$ cannot expect any help from those players who reside in a descendant of $\mathbf{a}$ (at least not in the core). Thus these building blocks can be 'separated' from the rest of the graph without altering the core or the nucleolus of the game. We will come back to this question in the next section.

Theorem 4 *$S \in \mathcal{S}_0$ if and only if S is saturated and $V(T_S)$ can be written as*

$$V(T_S) = V \setminus \cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$$

where $\mathbf{p}_j$ is a passage $\pi(\mathbf{p}_j) \in F$ and $F_j \subset F$ for all $j \in \{1, 2, \dots, k\}$.

Proof In light of Lemma 4 and Lemma 5 the *if* part can be verified easily. If the trunk of coalition S can be represented as $V(T_S) = V \setminus \cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$, then $V(T_S)$ is the complement of a disjoint union of building blocks and free nodes. As the residents of building blocks and the free nodes pay only for their own construction cost, the *rest of the players* have to pay for their own part of T_N. Thus from the $c_{\mathcal{D}}(N) = x(N)$ equality it follows that $c_{\mathcal{D}}(S) = x(S)$ for any core allocation x. Note that we implicitly used that every resident of $V(T_S)$ is involved in building T_S, that is S is saturated.

Now we prove the other direction i.e. $S \in \mathcal{S}_0 \Rightarrow V(T_S) = V \setminus \cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$. From Lemma 1 we know that we can choose a representation of $V(T_S)$ where $\mathbf{p}_j$ – the origin of the removed $B^{F_j}_{\mathbf{p}_j}$ branch – is a passage for all $j \in \{1, 2, \dots, k\}$. Furthermore, $\pi(\mathbf{p}_j) \in V(T_S)$ and $F_j \subset V(T_S)$ for all $j \in \{1, 2, \dots, k\}$.

If T_S has a shortcut then the standard allocation induces a non-zero excess for S. It follows that T_S is a connected subgraph of T_N. First let us consider a simple graph structure when only one branch is missing, that is $V(T_S) = V \setminus B^Q_{\mathbf{p}}$. If $\pi(\mathbf{p})$ is not free then there exist a core allocation y where $y(N(\mathbf{p})) > \tau(\mathbf{p}, N)$. The argument is similar to the reasoning used in the first part of Theorem 3. As $\pi(\mathbf{p})$ is not free, $\Pi(\pi(\mathbf{p}))$ is a passage. A coalition that contains a player from $N(\mathbf{p})$ has to use this passage or go around with a shortcut. In either case the standard allocation can be modified: a little amount can be transferred from $N(\Pi(\pi(\mathbf{p})))$ to $N(\mathbf{p})$ without leaving the core. Thus if the excess of $N(V \setminus B^Q_{\mathbf{p}})$ was zero under the standard allocation it is not zero under y. Now let $V(T_S) = V \setminus \cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$ and let us use the standard representation of $V(T_S)$. Take an arbitrary $\pi(\mathbf{p}_j)$. Basically the same argument works as above, we only need to show that $\Pi(\pi(\mathbf{p}_j))$ is in $V(T_S)$. Suppose on the contrary that $\Pi(\pi(\mathbf{p}_j)) \notin V(T_S)$. We know that every path from $\pi(\mathbf{p}_j)$ to the root that lies in T_N crosses $\Pi(\pi(\mathbf{p}_j))$. Since T_S is a subgraph of T_N it follows that $\pi(\mathbf{p}_j) \notin V(T_S)$. However, in the standard representation $\mathbf{p}_j$ was chosen such way that $\pi(\mathbf{p}_j) \in V(T_S)$ – a contradiction.

Finally, we need to prove that if $F_j \not\subset F$ then $S \notin \mathcal{S}_0$. Let $\mathbf{f}$ be an arbitrary non-free element of a given F_j. There exists a path in T_N from $\mathbf{f}$ to $\pi(\mathbf{p}_j)$ through $B^{F_j}_{\mathbf{p}_j}$. There exists another path in T_S, arc-disjoint from the previous one to the root. By our previous observation if this path contains a shortcut, then $S \notin \mathcal{S}_0$. Thus this path lies entirely in T_N. Since $\pi(\mathbf{p}_j)$ is free there exists two semi-arc-disjoint paths from $\pi(\mathbf{p}_j)$ to the root. It is impossible that the path from $\mathbf{f}$ to the root intersects both of these paths at a passage, since then they would not be semi-arc-disjoint. Thus there exist two semi-arc-disjoint paths from $\mathbf{f}$ to the root i.e. $\mathbf{f}$ is free.

Notice that this direction did not require for coalition S to be saturated. Non-saturated coalitions can have zero excess in the core, in particular when there are occupied free nodes in the trunk of S.

The interpretation of Theorem 4 becomes simpler when we consider the free nodes as some kind of secondary roots. The residents of a free node do not have to pay (Theorem 3), and the residents of a full branch that originates from a free node pay only for their own branch's construction cost (a consequence of Theorem 3 and Lemma 3). These results already suggest that contracting the free nodes with the root does not alter the structure of the core. In the next section we will introduce graph transformations that simplify the network. In addition we offer an approach to efficiently describe the core of a large family of DAG-games.

We conclude this section with a lemma that gives an upper bound on how much certain branches are willing to pay in the core. Let a_s be an arc that originates from a non-free node $\mathbf{p}$. We say that a_s is *critical* if replacing a_s with a zero arc would set $\mathbf{p}$ free.

Lemma 6 *Let* $\mathbf{p}$ *be an arbitrary node with a critical arc* $a_s \in A_{\mathbf{p}}$ *and let* $B^Q_{\mathbf{p}}$ *be a* $\mathbf{p}$*-branch. If* a_s *is not a shortcut then* $x(N(B^Q_{\mathbf{p}})) \leq \tau(B^Q_{\mathbf{p}}, N)$ *for any core allocation* x*. If* a_s *is a shortcut then* $x(N(B^Q_{\mathbf{p}})) \leq \tau(B^Q_{\mathbf{p}}, N) + \delta(a_s)$ *for any core allocation* x*.*

This lemma is basically a corollary of Lemma 3. For a detailed proof, see (Sziklai et al, 2014).

7 Finding the core of a DAG-game

In this section we uncover the graph structure of dually essential coalitions. As it will turn out it is simple and easy to deal with. With the help of the dually essential coalitions we prove that contracting the free nodes with the root or rerouting critical shortcuts to the root does not alter the core or the nucleolus of the game. The main result is that whenever the size of the set of dually essential coalitions is polynomially bounded in the number of players and we can efficiently enumerate them, then the core and the nucleolus can be computed in polynomial time.

First we show that dual essentiality is a stricter property than saturatedness.

Lemma 7 *If $\Gamma = (N, c)$ is a monotonic cost game, then $\mathcal{DE}(\Gamma) \subseteq \mathcal{S}(\Gamma)$. That is, each dually essential coalition is either saturated or consist of $n-1$ players.*

Proof Let S be a non-saturated coalition with at most $n-2$ players. We will show that S is dually inessential. As S is not saturated there exists $i \in N \setminus S$ such that $c_{\mathcal{D}}(S) = c_{\mathcal{D}}(S \cup \{i\})$. Let $S_1 := S \cup \{i\}$ and $S_2 := N \setminus \{i\}$. Then $S_1 \cup S_2 = N$ and $S_1 \cap S_2 = S$ therefore we can use Definition 2 since

$$\begin{aligned}
c_{\mathcal{D}}(N) &\geq c_{\mathcal{D}}(N \setminus \{i\}),\\
c_{\mathcal{D}}(S) &\geq c_{\mathcal{D}}(S) + c_{\mathcal{D}}(N \setminus \{i\}) - c_{\mathcal{D}}(N),\\
c_{\mathcal{D}}(S) &\geq c_{\mathcal{D}}(S_1) + c_{\mathcal{D}}(S_2) - c_{\mathcal{D}}(N).
\end{aligned}$$

In other words S appears in an overlapping decomposition of S_1 and S_2, therefore it cannot be dually essential.

The following theorem characterizes dually essential coalitions.

Theorem 5 *The dually essential coalitions of the cost game $\Gamma_{\mathcal{D}}$ are the coalitions with $n-1$ players and saturated coalitions whose trunks correspond to node sets of the form $V \setminus B^U_{\mathbf{q}}$ where $B^U_{\mathbf{q}}$ is a proper branch and $\mathbf{q}$ is a passage.*

Proof We have already seen in Lemma 7 that only saturated and $(n-1)$-player coalitions are dually essential. By Lemma 1 we know that trunks of (saturated) coalitions can be generated by removing branches from G. The one thing we have to prove is that coalitions that correspond to trunks that have more missing branches are dually inessential. Let S be a saturated coalition for which $V(T_S) = V \setminus \cup_{j=1}^{k} B^{Q_j}_{p_j}$ where $k \geq 2$. As $\mathcal{D}$ is in canonical form there resides at least one player in each of the branches. These branches may intersect with each other in general, however – by choosing the Q_j sets appropriately – we can always arrange them in such way that they are mutually disjoint. Among such representations we choose one where Q_k is either empty or a subset of $V(T_S)$.

For convenience's sake let us introduce the following notation $B_1 = \cup_{j=1}^{k-1} B_{p_j}^{Q_j}$ and $B_2 = B_{p_k}^{Q_k}$. Then let $S_1 = N \setminus N(B_1)$ and $S_2 = N \setminus N(B_2)$. In this way $N \setminus S = (N \setminus S_1) \dot{\cup} (N \setminus S_2)$. To prove that $c_{\mathcal{D}}(S) \geq c_{\mathcal{D}}(S_1) + c_{\mathcal{D}}(S_2) - c_{\mathcal{D}}(N)$ holds as well it is enough to show that the following two inequalities are true.

$$c_{\mathcal{D}}(S_1) \leq c_{\mathcal{D}}(S) + \tau(B_2, N) - \tau(Q_k, S) \tag{2}$$

$$c_{\mathcal{D}}(S_2) \leq c_{\mathcal{D}}(N) - \tau(B_2, N) + \tau(Q_k, S) \tag{3}$$

Note that it takes at most $\tau(B_2, N)$ to connect the players residing at B_2 to T_S. As $B_{p_k}^{Q_k}$ is a proper branch it follows that the nodes in Q_k are junctions. Since the nodes in Q_k are direct ancestors of some nodes in B_2 they are connected with zero arcs. Therefore we can save at least $\tau(Q_k, S)$ amount of cost by connecting Q_k through the branch B_2 and not through the arcs in $(\cup_{\mathbf{q} \in Q_k} A_q) \cap A(T_S)$. It is possible that aside from Q_k there are other nodes that can reach the root in a cheaper way using the arcs of B_2, but no nodes of $V(T_S)$ is forced to take a more expensive path. Summarizing the above findings we gather that

$$c_{\mathcal{D}}(S_1) \leq c_{\mathcal{D}}(S) + \tau(B_2, N) - \tau(Q_k, S)$$

We can estimate $c_{\mathcal{D}}(S_2)$ by keeping track how the cost changes as we swift from T_N to T_{S_2}. As $N(B_2)$ are not in S_2 we can delete B_2 and subtract $\tau(B_2, N)$ amount of cost from $c_{\mathcal{D}}(N)$. Deleting B_2 from T_N only the direct descendants of B_2 can get disconnected. Therefore the only nodes that may not be connected to the root are Q_k and their descendants. By building $(\cup_{\mathbf{q} \in Q_k} A_q) \cap A(T_S)$ – the exact same arcs that we deleted in case of S_1 – we can ensure that every node in $V \setminus B_2 \setminus \{\mathbf{r}\}$ has a leaving arc. None of these arcs enter to B_2, thus we obtained a trunk. Therefore the cost of reconnecting Q_k is at most $\tau(Q_k, S)$. Altogether we can estimate the cost of S_2 by

$$c_{\mathcal{D}}(S_2) \leq c_{\mathcal{D}}(N) - \tau(B_2, N) + \tau(Q_k, S).$$

Now adding (2) and (3) together, then subtracting $c_{\mathcal{D}}(N)$ from both sides yield us the desired result.

If S_1 and S_2 are dually essential coalitions then we are done. However, it can happen that one or both of them are dually inessential. Thus to prove dual inessentiality of S we may have to refine $c_{\mathcal{D}}(S) \geq c_{\mathcal{D}}(S_1) + c_{\mathcal{D}}(S_2) - c_{\mathcal{D}}(N)$. If $k \geq 3$ then T_{S_1} has more missing subbranches. Also S_1 and S_2 may not even be saturated. However, by repeatedly using the reasoning in Lemma 7 and the above argument we can obtain a weakly minorizing overlapping decomposition of S. Since each refinement uses larger coalitions we cannot run into a cycle[3]. This concludes the proof of the Theorem.

[3] For a detailed example of the decomposition process see (Sziklai, 2015).

Theorem 5 is surprisingly analogous to the one derived by Maschler et al (2010) for standard tree games (see Lemma 2.3 in the cited paper). Although they do not speak of characterization sets the relationship between the two results is unquestionable.

The graph characterization of dually essential coalitions is one of the main results of this paper. It will allow us to describe the core and to compute the nucleolus for a large family of DAG-games. The next few lemmata introduces further simplifications in the network.

Lemma 8 *Let $S \in \mathcal{DE}(\mathit{\Gamma}_{\mathcal{D}})$ and $V(T_S) = V \setminus B^U_{\mathbf{p}}$ its standard representation. Then $\mathbf{q} \in B^U_{\mathbf{p}} \Rightarrow \mathit{\Pi}(\mathbf{q}) \in B^U_{\mathbf{p}}$.*

Proof Suppose $\mathit{\Pi}(\mathbf{q}) \notin B^U_{\mathbf{p}}$. There exists two semi-arc disjoint paths in T_N from $\mathbf{q}$ to $\mathit{\Pi}(\mathbf{q})$. Only one of these paths may use $\mathbf{p}$. Thus there is zero arc that leaves $B^U_{\mathbf{p}}$, which contradicts that T_S has maximum number of arcs among the cheapest trunks that connect all the players in S to the root.

A network can be simplified if it has critical shortcuts. Lemma 6 suggests that it does not matter where a critical shortcut enters. Thus critical shortcuts can be treated as if they were pointing to the root.

Lemma 9 *Let $\mathcal{D}$ be a DAG-network and $\mathit{\Gamma} = (N, c_{\mathcal{D}})$ be the corresponding game. Let $\mathbf{s}$ be a junction with a critical shortcut $a_{\mathbf{s}}$. Finally, let $\mathcal{D}'$ be a network that is obtained from $\mathcal{D}$ by rerouting $a_{\mathbf{s}}$ to the root and let $\mathit{\Gamma}' = (N, c_{\mathcal{D}'})$. The core and the nucleolus is unchanged by this transformation, formally $\mathcal{C}(\mathit{\Gamma}) = \mathcal{C}(\mathit{\Gamma}')$ and $\mathcal{N}(\mathit{\Gamma}) = \mathcal{N}(\mathit{\Gamma}')$.*

Proof It is enough to prove that $\mathcal{DE}(\mathit{\Gamma}) = \mathcal{DE}(\mathit{\Gamma}')$ and for any $S \in \mathcal{DE}(\mathit{\Gamma})$, $c_{\mathcal{D}}(S) = c_{\mathcal{D}'}(S)$. We will use the graph representation of dually essential coalitions that we uncovered in Theorem 5. It is easy to check that the $(n-1)$-player coalitions have the same characteristic function value in both games. Let $S = N \setminus i$ and let $\mathbf{t}$ denote the node where $a_{\mathbf{s}}$ enters. If player i resides in a junction then $T_S = T_N$, that is the trunk of S does not contain any shortcuts, hence it is unimportant where $a_{\mathbf{s}}$ points to. If player i resides in a passage $\mathbf{v}$ and T_S does not contain $\mathbf{v}$, then the descendants of $\mathbf{v}$ must use an alternative path to reach the root. If a direct descendant of $\mathbf{v}$ is a passage then $\mathbf{v} \in T_S$, hence all its children must be junctions. Due to the canonization every such junction must have an arc that points either to an ancestor of $\mathbf{v}$ or to a node that is unrelated to $\mathbf{v}$. That means that if $a_{\mathbf{s}} \in A(T_S)$ then $\mathbf{t} \in T_S$ both in $\mathcal{D}$ and in $\mathcal{D}'$. Thus it does not matter if $a_{\mathbf{s}}$ points to $\mathbf{t}$ or to the root, the cost of T_S does not change.

Now let $|S| < n-1$. Due to Theorem 5 we may assume $V(T_S) = V \setminus B^U_{\mathbf{p}}$. We need to prove that for any such $S \in \mathcal{DE}(\mathit{\Gamma})$ it is true that $c_{\mathcal{D}}(S) = c_{\mathcal{D}'}(S)$. Let $\mathbf{q}_1, \ldots, \mathbf{q}_k$ be the direct ancestors of $\mathbf{s}$, and let $\mathbf{t}$ be the node where $a_{\mathbf{s}}$ enters. If $\mathbf{s} \in B^U_{\mathbf{p}}$ then T_S is the same in both games. If $\mathbf{q}_i \notin B^U_{\mathbf{p}}$ for some $i = 1, \ldots, k$ then $\mathbf{s}$ can be connected to the root without using $a_{\mathbf{s}}$ in both games. Hence it is indifferent whether the shortcut of $\mathbf{s}$ enters to $\mathbf{t}$ or to $\mathbf{r}$. If all the $\mathbf{q}_1, \ldots, \mathbf{q}_k$ nodes are in the removed $B^U_{\mathbf{p}}$ branch and $\mathbf{s}$ connects to the root via $\mathbf{t}$, then the construction cost of S does not change by rerouting $a_{\mathbf{s}}$ from $\mathbf{t}$ to $\mathbf{r}$.

Now we prove that $\mathbf{q}_1, \ldots, \mathbf{q}_k, \mathbf{t} \in B^U_{\mathbf{p}}$ is impossible. By contradiction suppose that all the direct ancestors of $\mathbf{s}$ and the end node of the critical shortcut belongs to the missing branch. By definition if $\delta(a_{\mathbf{s}})$ is set to zero $\mathbf{s}$ becomes free. That is there leads two semi-arc disjoint paths from $\mathbf{s}$ to the root. The paths in T_N between $\mathbf{s}$ and $\mathbf{r}$ use a node from $\mathbf{q}_1, \ldots, \mathbf{q}_k, \mathbf{t}$, hence all the paths go through $B^U_{\mathbf{p}}$. If there are two semi-arc disjoint pathes that go through $B^U_{\mathbf{p}}$, then only one of them may use $\mathbf{p}$. Thus there is zero arc that leaves $B^U_{\mathbf{p}}$, which contradicts that T_S has maximum number of arcs among the cheapest trunks that connect all the players in S to the root.

The next lemma states that free nodes can be contracted with the root.

Lemma 10 *Let $\mathcal{D}$ be a DAG-network and $\Gamma = (N, c_{\mathcal{D}})$ be the corresponding game. Let $\mathbf{p}$ be any free node. Finally, let $\mathcal{D}'$ be a network that is obtained from $\mathcal{D}$ by contracting $\mathbf{p}$ with the root and let $\Gamma' = (N, c_{\mathcal{D}'})$. The leaving arcs of $\mathbf{p}$ are deleted, while the entering arcs now point to $\mathbf{r}$. The core and the nucleolus is unchanged by this transformation, formally $\mathcal{C}(\Gamma) = \mathcal{C}(\Gamma')$ and $\mathcal{N}(\Gamma) = \mathcal{N}(\Gamma')$.*

Proof Again we could proceed by checking whether $\mathcal{DE}(\Gamma) = \mathcal{DE}(\Gamma')$ and whether for any $S \in \mathcal{DE}(\Gamma)$, $c_{\mathcal{D}}(S) = c_{\mathcal{D}'}(S)$. However, by using Lemma 9 we can give a much simpler proof.

If $\mathbf{f}$ is a free node and there is a zero-arc that leaves $\mathbf{f}$ and enters the root, then $\mathbf{f}$ can be contracted with the root. Similarly we can contract every node from where the root can be reached on a zero cost path. Obviously the characteristic function is unaffected by this transformation. Now take a free node $\mathbf{p}$ and transform the network in the following way. We assign additional $\varepsilon > 0$ costs to the zero arcs of $\mathbf{p}$ one after another until one of them becomes a critical arc. Note that if all the zero arcs are replaced in this way, we may have to canonize the graph again. This however does not affect the game as **P1** leaves the characteristic function untouched and $\mathbf{p}$ will still have a critical arc. The core inequalities are continuous in the arc costs. Since each coalition uses at most one non-zero arc that leaves $\mathbf{p}$, each core inequality is shifted by at most ε. By Lemma 9 we can reroute the critical arc to the root. By taking $\varepsilon \to 0$ we obtain a new network with the same core, but where $\mathbf{p}$ has a zero-arc that enters the root. Thus $\mathbf{p}$ can be contracted with the root.

Note that rerouting the critical shortcuts and contracting the free nodes with the root do change the characteristic function of the game. Thus they are not equivalent transformations, as opposed to the canonization which leaves the characteristic function untouched.

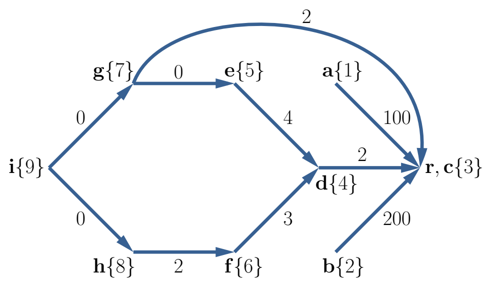

Fig. 7 Simplified network of Example 3. Node **c** is contracted with the root, while the shortcut of **g** is rerouted from **a** to **r**.

8 Applicability of our results

Let us examine Example 3 one last time. In light of Lemmata 9 and 10 the network can be simplified (see Figure 7). Although the DAG-games induced by the player-networks of Figure 6 and 7 generate different characteristic functions, they are equivalent from the point of view of the core or the nucleolus[4].

Many technical results (e.g. Theorem 3 and Lemma 6) become apparent due to these transformations. By looking at Figure 7 it is clear that player 3 will not pay anything in the core. It is also evident that players 1 and 2 cannot expect help from other players. They have to construct their rather expensive links alone. Moreover player 7 will not play more that two units in the core due to its shortcut.

Whether the core can be described efficiently with dually essential coalitions, depends on how many distinct proper branches of standard form exist in the network. Unfortunately as the next example shows there can be exponentially many dually essential coalitions in a DAG-game.

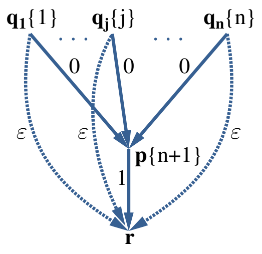

Fig. 8 A DAG-network with exponential many proper branches. Solid lines indicate T_N-arcs, while dotted lines are shortcuts.

[4] Note that the simplifications of Lemmata 9 and 10 alter the Shapley-value (and many other solution concept) of the game. This also sheds a light why the Shapley-value is not a good solution concept in case of DAG-games. It is easy to generate an example where the Shapley-value lies outside the core. Alternatively one could argue that the core conditions are too demanding. It is unrealistic that the residents of free nodes - like player 3 in this example - do not contribute to the construction costs.

Example 4 Consider the DAG-network depicted in Figure 8. The root has only one direct descendant, namely $\mathbf{p}$, while the nodes $\mathbf{q}_1, \ldots, \mathbf{q}_n$ are the children of $\mathbf{p}$. Each of the $\mathbf{q}_j$ nodes have one additional arc – a shortcut – that enters the root. The cost of the shortcuts are chosen in such way that their total cost is less than the cost of the T_N-arc of $\mathbf{p}$. For instance let $\delta(a_{\mathbf{p}}) = 1$, and $\delta(a_{\mathbf{s}}) = \varepsilon = \frac{1}{n+1}$ for each shortcut $a_s \in A$. Let us assume that one player resides in each node, the jth player at $\mathbf{q}_j$ and the $(n+1)$st player at $\mathbf{p}$. Let N' denote the set of the first n players. For an arbitrary $S \subset N'$, T_S correspond to $V \setminus B_{\mathbf{p}}^{Q_S}$, where $Q_S \overset{def}{=} \{\mathbf{q}_j | \ j \in S\}$. Thus any subset of N' is dually essential. As there are n player in N' there are at least 2^n dually essential coalitions in this game.

The good news is that whenever we can efficiently enumerate all proper branches of the network we can obtain a description of the core. If $B_{\mathbf{p}}^{Q}$ is a proper branch, then $V \setminus B_{\mathbf{p}}^{Q}$ corresponds to a trunk T. However, this trunk might be not the cheapest trunk for coalition $S = N(V \setminus B_{\mathbf{p}}^{Q})$. It might happen that the residents of Q can save some cost by constructing $B_{\mathbf{p}}^{Q}$ or a part of it. Checking whether T is the cheapest trunk, involves the $\mathcal{NP}$-hard Steiner arborescence problem. Luckily we do not need to check whether $T_S = T$ or not. If it is then S is dually essential. If it isn't, then S is dually inessential and $c_{\mathcal{D}}(S) = C(T_S) < C(T)$. The dually inessential coalitions are redundant in the computation of the core or the nucleolus of the game. By weakening the $c_{\mathcal{D}}(S) \geq x(S)$ inequality that corresponds to a dually inessential coalition we cannot cut into the core, i.e. the coalition remains redundant.

Now we provide a large family of DAG-games where there are polynomially many proper branches. We define the *width* of a DAG-network as the maximum number of nodes that can be chosen from the node set, such that any two chosen nodes are incomparable, i.e. neither of them is an ancestor or descendant of the other[5].

Lemma 11 *Let $\mathcal{D}$ be a canonized DAG-network of width k and let m denote the number of nodes in the graph. There are at most $\mathcal{O}(m^{k+1})$ number of proper branches in $\mathcal{D}$.*

Proof We argue that every branch in $\mathcal{D}$ can be characterized by choosing at most $k+1$ nodes appropriately. Then the lemma follows from the fact that $\binom{m}{k+1} \leq m^{k+1}$. From Lemma 8 we know that if $\mathbf{q} \in B_{\mathbf{p}}^{Q}, \mathbf{q} \neq \mathbf{p}$ then every key ancestor of $\mathbf{q}$ lies inside $B_{\mathbf{p}}^{Q}$. This implies that each $\mathbf{q}'$ such that $\mathbf{p} \preceq \mathbf{q}' \preceq \mathbf{q}$ is contained in $B_{\mathbf{p}}^{Q}$. Let $\mathbf{q}_1, \ldots, \mathbf{q}_\ell$ the maximum number of incomparable nodes that we can choose from $B_{\mathbf{p}}^{Q}$ such that no descendant of $\mathbf{q}_j$ is an element of $B_{\mathbf{p}}^{Q}$ for $1 \leq j \leq \ell$. We claim that the nodes $\mathbf{p}, \mathbf{q}_1, \ldots, \mathbf{q}_\ell$ characterize $B_{\mathbf{p}}^{Q}$. We use a coloring argument. First we color the nodes $\mathbf{q}_1, \ldots, \mathbf{q}_\ell$ then we color all their descendants till we reach $\mathbf{p}$, finally we color $\mathbf{p}$. Notice that a node is colored if and only if it is an element of $B_{\mathbf{p}}^{Q}$. No descendant of the $\mathbf{q}_j$ nodes

[5] If we think about the DAG as a partially ordered set then width is equivalent to the cardinality of the *maximum antichain* the poset has.

is an element of $B^Q_{\mathbf{p}}$ and we included all the descendant of the $\mathbf{q}_j$ nodes. An uncolored node $\mathbf{p}' \in B^Q_{\mathbf{p}}$ would contradict that we choose the maximum number of incomparable nodes. Since $\ell \leq k$ each branch can be described by at most $k+1$ node.

Our last theorem summarizes our findings.

Theorem 6 *There exists a polynomial time algorithm in the number of players to compute the core and the nucleolus of any DAG-game that is induced by a fixed width canonized DAG-network.*

Theorem 6 is a direct corollary of Lemma 11 and the fact that the cost of the $(n-1)$-player coalitions can be easily calculated for any DAG-game. Hence, all the dually essential coalitions can be accounted for. Dually essential coalitions form a characterization set for the nucleolus and describe the core, thus, we are done.

Note that efficiency in Lemma 11 is measured in the number of nodes while the time complexity of the algorithm in Theorem 6 is measured in the number of players. This does not create any inconsistency as we are only interested in saturated coalitions. Hence there is a one-to-one correspondence between the $B^Q_{\mathbf{p}}$ branches and the interesting coalitions.

Finally, let us note, that similarly to the painting algorithm which computes the nucleolus for standard tree games (Maschler et al, 2010), a fast graph based algorithm can be designed for the nucleolus of some families of canonized DAG-games. Sziklai (2015) demonstrates how such an algorithm can be implemented in DAG-networks that have no shortcuts. Furthermore, he conjectures that the extension of this algorithm works for DAG-networks where every shortcut is critical. It is unclear how hard the general case is. However, due to Theorem 2, it is unlikely that a polynomial time algorithm exists for networks where property (*) does not hold.

Acknowledgements

The authors would like to thank Prof. Tamás Király of ELTE Operations Research Department and the anonymous referees for their valuable comments. Research was funded by OTKA grants K101224 and K108383 and by the Hungarian Academy of Sciences under its Momentum Programme (LD-004/2010). This research was also partially supported by MTA-ELTE Egerváry Research Group and by Pallas Athene Domus Scientiae Foundation. The views expressed are those of the authors' and do not necessarily reflect the official opinion of Pallas Athene Domus Scientiae Foundation.

References

Bergantiños G, Vidal-Puga JJ (2007) A fair rule in minimum cost spanning tree problems. Journal of Economic Theory 137:326–352

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

Bjørndal E, Koster M, Tijs S (2004) Weighted allocation rules for standard fixed tree games. Math Methods of Operations Research 59(2):249–270

Bogomolnaia A, Moulin H (2010) Sharing a minimal cost spanning tree: Beyond the folk solution. Games and Economic Behavior 69:238–248

Deng X, Fang Q, Sun X (2009) Finding nucleolus of flow game. Journal of Combinatorial Optimization 18(1):64–86

Faigle U, Kern W, Hochstättler W, Fekete S (1997) On the Complexity of Testing Membership in the Core of Min-Cost Spanning Tree Games. International Journal of Game Theory 26(3):361–366

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

Granot D, Granot F (1992) Computational complexity of a cost allocation approach to a fixed cost spanning forest problem. Mathematics of Operations Research 17(4):765–780

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

Granot D, Huberman G (1984) On the core and nucleolus of minimum cost spanning tree games. Mathematical Programming 29(3):323–347

Granot D, Maschler M (1998) Spanning network games. International Journal of Game Theory 27:467–500

Granot D, Granot F, Zhu WR (1998) Characterization sets for the nucleolus. International Journal of Game Theory 27(3):359–374

Huberman G (1980) The nucleolus and essential coalitions. In: Bensoussan A, Lions JL (eds) Analysis and Optimization of Systems, Elsevier B.V., Lecture Notes in Control and Information Sciences, vol 28, pp 416–422

Hwang FK, Richards DS, Winter P (1992) The Steiner Tree Problem, Annals of Discrete Mathematics, vol 53. North-Holland

Kuipers J (1997) Minimum cost forest games. International Journal of Game Theory 26:367–377

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

Maschler M, Potters J, Reijnierse H (2010) The nucleolus of a standard tree game revisited: a study of its monotonicity and computational properties. International Journal of Game Theory 39(1-2):89–104

Megiddo N (1978) Computational Complexity of the Game Theory Approach to Cost Allocation for a Tree. Math of Operations Research 3(3):189–196

van den Nouweland A, Tijs S, Maschler M (1993) Monotonic Games Are Spanning Network Games. International Journal of Game Theory 21(4):419–27

Potters JAM, Sudhölter P (1999) Airport problems and consistent allocation rules. Mathematical Social Sciences 38:83–102

Reijnierse H, Potters JAM (1998) The $\mathcal{B}$-nucleolus of tu-games. Games and Economic Behavior 24:77–96

Rosenthal EC (1987) The minimum cost spanning forest game. Economics Letters 23:355–357

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

Skorin-Kapov D, Skorin-Kapov J (2012) A note on steiner tree games. Networks 59(2):215–225

Solymosi T, Sziklai B (2016) Characterization sets for the nucleolus in balanced games. Operations Research Letters 44:520–524

Sziklai B (2015) On the computation of the nucleolus of cooperative transferable utility games. Phd thesis, Eötös Loránd University, Budapest

Sziklai B, Fleiner T, Solymosi T (2014) On the core of directed acyclic graph games. IEHAS Discussion Papers MT-DP 2014/18

Trudeau C (2012) A new stable and more responsive cost sharing solution for minimum cost spanning tree problems. Games and Econ Beh 75:402–412

Appendix A

Theorem 2 *Let N be a player set and $\hat{c} : 2^N \to \mathbb{R}$ a monotonic, subadditive cost function. There exists a DAG-network $\mathcal{D} = (G(V, A), \delta)$ and a residency mapping $\mathcal{R} : N \to V$ such that*

$$c_{(\mathcal{D},\mathcal{R})}(S) = \hat{c}(S) \quad \forall S \subseteq N.$$

Proof We will use the characteristic DAG-representation of $\hat{c}$. We have to show that the cost of the cheapest trunk of any coalition S equals to $\hat{c}(S)$. This is trivially true for the singleton coalitions. Each player – as a singleton – will use its direct connection to the root as any other route could be only more expensive due to the monotonicity of $\hat{c}$. Now let S be an arbitrary non-singleton coalition. We may suppose that $\delta(a_S) > 0$, otherwise both $\hat{c}(S)$ and $c_{(\mathcal{D},\mathcal{R})}(S)$ are trivially zero. Let T' be the trunk whose arc set consists of a_S and the zero arcs $\{a^i_S \mid i \in S\}$ and whose node set consists of the nodes spanning these arcs. We will prove $T' \in T_S$. Let $T \in T_S$ and suppose that $\{a_{S_1}, \dots, a_{S_k}\}$ are the non-zero arcs of T. If $S = S_\ell$ for some $\ell \in \{1, \dots, k\}$, then by deleting the a_{S_j}, $j \neq \ell$ arcs and the zero arcs that enter $\mathbf{p}_{S_j}$, $j \neq \ell$ we can obtain T'. This would imply $T' \in T_S$ as the cost could only decrease by deleting these arcs. If $S \subset S_\ell$ for some $\ell \in \{1, \dots, k\}$, then again we can obtain a weakly cheaper trunk by connecting the members of S via a_S and deleting all the other a_{S_ℓ} arcs. Thus $S_\ell \subset S$ for all $\ell \in \{1, \dots, k\}$. Although a player may be connected to more than one $\mathbf{p}_{S_\ell}$ node, in real he only needs one of the a_{S_ℓ} arcs to reach the root. Let us assign the players of S to one of the a_{S_ℓ} arcs. Let us denote by $S^\ell \subset S$ those players of S that were assigned to S_ℓ. Note that the $\{S^\ell \mid \ell \in \{1, \dots, k\}, S^\ell \neq \emptyset\}$ coalitions comprise a partition of S. If $S^\ell = \emptyset$ for some $\ell \in \{1, \dots, k\}$ then a_{S_ℓ} and the entering zero arcs of $\mathbf{p}_{S_\ell}$ can be deleted. If $\emptyset \neq S^\ell \subset S_\ell$ then the players of S^ℓ can be reassigned to a_{S^ℓ}, that is, we can delete a_{S_ℓ} and the entering zero arcs of $\mathbf{p}_{S_\ell}$ and construct a_{S^ℓ} and the the entering zero arcs of $\mathbf{p}_{S^\ell}$ instead. Due to the monotonicity, the cost can only decrease this way, while all the players of S are still able to reach the root.

Let us perform this transformation for all the S_ℓ coalitions. We obtain a trunk $\hat{T} \in T_S$ such that

$$A(\hat{T}) = \{a_{S^\ell} \mid \ell \in \{1,\dots,k\}, S^\ell \neq \emptyset\} \cup \{a^i_{S^\ell} \mid \ell \in \{1,\dots,k\}, S^\ell \neq \emptyset, i \in S\}.$$

By the subadditivity of $\hat{c}$

$$\hat{c}(S) \leq \sum_{\ell \in \{1,\dots,k\}, S^\ell \neq \emptyset} \hat{c}(S^\ell)$$

$$\delta(a_S) \leq \sum_{\ell \in \{1,\dots,k\}, S^\ell \neq \emptyset} \delta(a_{S^\ell})$$

Thus, T' is as least as cheap as $\hat{T}$, from which $T' \in T_S$ follows.

Lemma 1 *The node set of every trunk that corresponds to a coalition $S \subset N$ can be obtained by deleting some branches from V. The removed branches can be chosen in such way that each of them originates from a passage. Formally for any $S \subset N$ there exists $Q_1,\dots,Q_k \subset V$ and $\mathbf{p}_1,\dots,\mathbf{p}_k \in V$ such that*

$$V(T_S) = V \setminus \cup_{j=1}^k B_{\mathbf{p}_j}^{Q_j},$$

where $\mathbf{p}_j$ is a passage for all $j \in \{1,2,\dots,k\}$.

Proof Any trunk T has a representation where $V(T)$ is obtained by removing branches from V (any single node is a branch in itself if we trim all its children). The only thing we need to prove is that these branches can be picked in such way that each of them originates from a passage. Let $\{\mathbf{p}_1,\dots,\mathbf{p}_k\} \subset V \setminus V(T_S)$ denote those passages that connect to $V(T_S)$ from the outside, i.e. for which $\pi(\mathbf{p}_j) \in V(T_S)$ for all $j = 1,\dots,k$. Due to the definition of T_S there exists at least one such passage. Note that any entering zero arc is included in the trunk of S even if no player of S resides there, due to the definition of T_S. If we remove all the $B_{\mathbf{p}_1},\dots,B_{\mathbf{p}_k}$ branches from V it can happen that we removed some nodes in $V(T_S)$ as well i.e. $V \setminus (\cup_{j=1}^k B_{\mathbf{p}_j}) \subset V(T_S)$. In order to retain all the nodes of $V(T_S)$ we trim the $B_{\mathbf{p}_j}$ branches where they intersect with $V(T_S)$. Let $Q_j = V(T_S) \cap B_{\mathbf{p}_j}$ then $B_{\mathbf{p}_j}^{Q_j}$ is a proper branch for any j and $V(T_S) = V \setminus (\cup_{j=1}^k B_{\mathbf{p}_j}^{Q_j})$.

Lemma 4 *If $B_{\mathbf{p}}^Q$ is a building block, then $x(N(B_{\mathbf{p}}^Q)) = \tau(B_{\mathbf{p}}^Q, N)$ for any core allocation x.*

Proof Since $\pi(\mathbf{p})$ is free, it is a junction and $x(N(\pi(\mathbf{p}))) = 0$. We know from Theorem 3 that $exc(N(V \setminus B_{\pi(\mathbf{p})}), x) = 0$ for any core allocation x. It follows that $exc(N((V \setminus B_{\pi(\mathbf{p})}) \cup \{\pi(\mathbf{p})\}), x) = 0$ is also true. With a similar argument as in Lemma 3 it can be shown that $x(N(B_{\mathbf{p}}^Q)) \leq \tau(B_{\mathbf{p}}^Q, N)$.

Each node of Q has (at least) two semi-arc-disjoint paths that leads to the root. As $B_{\mathbf{p}}^Q$ does not contain a free node one of these paths for each node by-passes $B_{\mathbf{p}}^Q$. We prove this by contradiction. Let $\mathbf{q} \in Q$ an arbitrary free node. Suppose there exists two semi-arc-disjoint paths in T_N, P_1 and P_2 that leads from $\mathbf{q}$ to the root and crosses $B^Q_{\mathbf{p}}$. Let $\mathbf{q}_1 \in B^Q_{\mathbf{p}} \cap V(P_1)$ be such that there exist no other $\mathbf{q}' \in B^Q_{\mathbf{p}} \cap V(P_1)$ such that $\mathbf{q}' \prec \mathbf{q}_1$. Similarly let $\mathbf{q}_2$ be the node closest to the root that is an element of both $B^Q_{\mathbf{p}}$ and P_2. As $\mathbf{q}_1$ and $\mathbf{q}_2$ lie on semi-arc-disjoint paths, one of them – say $\mathbf{q}_1$ – is not $\mathbf{p}$. Thus the P_1 path leaves the $B^Q_{\mathbf{p}}$ node set at $\mathbf{q}_1$ on a zero-arc. There leads a path in T_N from $\mathbf{q}_1$ to $\pi(\mathbf{p})$ through $B^Q_{\mathbf{p}}$ that is arc-disjoint of P_1. As $\pi(\mathbf{p})$ is free there leads two semi-arc-disjoint paths P_3 and P_4 from $\pi(\mathbf{p})$ to the root. Without loss of generality we may assume that P_1 intersects with P_3 first (or at the same time as it intersects with P_4). Let us denote this node by $\mathbf{q}^*$. Note that if $\mathbf{q}^*$ is a common node of P_3 and P_4 it is a junction, otherwise the two paths would not be semi-arc-disjoint. Let P_A be the path that starts from $\mathbf{q}_1$, follows P_1 till $\mathbf{q}^*$, then reaches the root following P_3. Let P_B be the path that originates at $\mathbf{q}_1$,reaches $\pi(\mathbf{p})$ using only T_N-arcs and nodes from $B^Q_{\mathbf{p}}$, and goes to the root following P_4. By construction P_A and P_B are semi-arc-disjoint, thus $\mathbf{q}_1$ is free, which contradicts the assumption that $B^Q_{\mathbf{p}}$ is a building block.

It follows that there exists a path in T_N for every $\mathbf{q} \in Q$ that leads to the root, that does not pass through any node of $B^Q_{\mathbf{p}}$. A straightforward consequence is that $B^Q_{\mathbf{p}}$ is a proper branch and every node in $V \setminus B^Q_{\mathbf{p}}$ can reach the root by using only T_N-arcs. Note that there is no zero-arc that leaves $B^Q_{\mathbf{p}}$ and enters in $V \setminus B^Q_{\mathbf{p}}$, otherwise $B^Q_{\mathbf{p}}$ would contain a free node. Thus the node set $V \setminus B^Q_{\mathbf{p}}$ corresponds to a trunk, namely to $T_{N(V \setminus B^Q_{\mathbf{p}})}$. Finally, for any core allocation x

$$\begin{aligned} c_{\mathcal{D}}(N) - x(N) &= [c_{\mathcal{D}}(N(V \setminus B^Q_{\mathbf{p}})) - x(N(V \setminus B^Q_{\mathbf{p}}))] + [\tau(B^Q_{\mathbf{p}}, N) - x(N(B^Q_{\mathbf{p}}))] \\ 0 &= [exc(N(V \setminus B^Q_{\mathbf{p}})), x)] + [\tau(B^Q_{\mathbf{p}}, N) - x(N(B^Q_{\mathbf{p}}))] \end{aligned}$$

Both expressions in the square brackets are non-negative, thus $x(N(B^Q_{\mathbf{p}})) = \tau(B^Q_{\mathbf{p}}, N)$.

Lemma 5 *Let $\cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$ be a union of branches such that $\mathbf{p}_j$ is a passage, $\pi(\mathbf{p}_j) \in F$ and $F_j \subset F$ for $j = 1, \ldots, k$. Then $\cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$ can be decomposed into a disjoint union of building blocks and free nodes.*

Proof The proof proceeds by induction on the number of nodes. If $\cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}$ consist of a single node, then $k = 1$ and $B^{F_1}_{\mathbf{p}_1}$ must be a building block. Now suppose the lemma is true for node sets with less than l nodes and let $| \cup_{j=1}^k B^{F_j}_{\mathbf{p}_j}| = l$. Let $B^Q_{\mathbf{p}_1}$ be a branch where $Q = B_{\mathbf{p}_1} \cap F$ and let $B^{Q'}_{\mathbf{p}_1}$ be the standard form of this branch. Note that $B^{Q'}_{\mathbf{p}_1}$ is a building block and it is a subset of $B^{F_1}_{\mathbf{p}_1}$. Let us delete $B^{Q'}_{\mathbf{p}_1}$ from $B^{F_1}_{\mathbf{p}_1}$. If $Q' \cap B^{F_1}_{\mathbf{p}_1}$ is not empty we delete those nodes too (these are free as all the nodes of Q' are free). If some descendant of a node in Q' is a junction then it is free therefore it can be deleted too. If we deleted all the free nodes in this way and there are still some nodes in $B^{F_1}_{\mathbf{p}_1}$ then those must be passages. Let us denote these by $\mathbf{p}'_1, \ldots, \mathbf{p}'_K$. Note that $\pi(\mathbf{p}'_1), \ldots, \pi(\mathbf{p}'_K) \in F$. Hence the remaining nodes can be written as $\cup_{i=1}^K B^{F_1}_{\mathbf{p}'_i} \cup_{j=2}^k B^{F_j}_{\mathbf{p}_j}$. By reindexing $\mathbf{p}'_i$ we are done as $| \cup_{i=1}^K B^{F_1}_{\mathbf{p}'_i} \cup_{j=2}^k B^{F_j}_{\mathbf{p}_j}| < l$.