MAGYAR TUDOMÁNYOS AKADÉMIA
Közgazdaság- és Regionális Tudományi Kutatóközpont

MTA KRTK KTI

Centre for Economic and Regional Studies
HUNGARIAN ACADEMY OF SCIENCES

MŰHELYTANULMÁNYOK DISCUSSION PAPERS

MT-DP – 2014/18

On the Core of Directed Acyclic Graph Games

BALÁZS SZIKLAI - TAMÁS FLEINER - TAMÁS SOLYMOSI

INSTITUTE OF ECONOMICS, CENTRE FOR ECONOMIC AND REGIONAL STUDIES,
HUNGARIAN ACADEMY OF SCIENCES BUDAPEST, 2014

Discussion papers
MT-DP – 2014/18

Institute of Economics, Centre for Economic and Regional Studies,
Hungarian Academy of Sciences

On the Core of Directed Acyclic Graph Games

Authors:

Balázs Sziklai
junior research fellow
Institute of Economics
Centre for Economic and Regional Studies
Hungarian Academy of Sciences
E-mail: sziklai.balazs@krtk.mta.hu

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

Tamás Solymosi
senior research fellow
Institute of Economics
Centre for Economic and Regional Studies
Hungarian Academy of Sciences
associate professor
Department of Operations Research and Actuarial Sciences
E-mail: solymosi.tamas@krtk.mta.hu

July 2014

ISBN 978-615-5447-33-4
ISSN 1785 377X

On the Core of Directed Acyclic Graph Games

Balázs Sziklai – Tamás Fleiner – Tamás Solymosi

Abstract

There lies a network structure between fixed tree and minimum cost spanning tree networks that has not been previously analyzed from a cooperative game theoretic perspective, namely, directed acyclic graph (DAG) networks. In this paper we consider the cost allocation game defined on DAG-networks. We briefly discuss the relation of DAG-games with other network-based cost games. We demonstrate that in general a DAG-game is not concave, even its core might be empty, but we provide an efficiently verifiable condition satisfied by a large class of directed acyclic graphs that is sufficient for 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.

Keywords: cooperative game theory, directed acyclic graphs, core, acyclic directed Steiner tree

JEL classification: C71

Acknowledgement:

The author thanks the funding of the Hungarian Academy of Sciences under its Momentum Programme (LD-004/2010). Research was funded by OTKA grant K108383 and by OTKA grant K101224.

Irányított aciklikus gráf játékok magja

Sziklai Balázs – Fleiner Tamás – Solymosi Tamás

Összefoglaló

Cikkünkben a standard fa játékok egy általánosításával foglalkozunk, amelyben a hálózat irányított aciklikus gráfként modellezhető. Röviden bemutatjuk, hogy a játék milyen kapcsolatban áll más hálózati költségjátékokkal. Megmutatjuk, hogy a játék nem konkáv, és a magja akár is üres lehet. Ugyanakkor egy hatékonyan ellenőrizhető feltételt is adunk, amely mellett a játék magja nem üres és amelyet irányított aciklikus gráfok nagy családja kielégít. Bevezetünk egy kanonizációs eljárást és számos strukturális eredményt bizonyítunk kanonizált irányított aciklikus gráfokon értelmezett játékokra. Többek között karakterizáljuk azoknak a játékosoknak a halmazát, akiknek a kifizetése a magban konstans 0, illetve megadjuk a koalícióknak egy olyan részhalmazát, amelyek már önmagukban meghatározzák a magot.

Tárgyszavak: kooperatív játékelmélet, irányított aciklikus gráfok, mag, irányított aciklikus Steiner fa

JEL kód: C71

On the Core of Directed Acyclic Graph Games

Balázs Sziklai[*][†]   Tamás Solymosi [‡]   Tamás Fleiner[†]

July 22, 2014

Abstract

In this paper we consider a natural generalization of standard tree games where the underlying network is a directed acyclic graph. We briefly discuss the relation of directed acyclic graph (DAG) games with other network-based cost games. We show that in general a DAG-game is not concave, even its core might be empty, but we provide an efficiently verifiable condition satisfied by a large class of directed acyclic graphs that is sufficient for balancedness of the associated DAG-game. We introduce a network canonization process and prove various structural results for the core of canonized DAG-games, for example, 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.

Keywords: Cooperative game theory, Directed acyclic graphs, Core, Acyclic directed Steiner tree

JEL-codes: C71

1 Introduction

Standard tree games form one of the most studied class of cost allocation games. In its most basic form (Megiddo, 1978), we have a (directed) tree, where nodes represent players, arcs represent connection possibilities between the nodes, and a non-negative connection cost is assigned to each arc. There is a special node, the so called root of the tree. This node represents the provider of some kind of service (e.g. electricity) that

[*]The author thanks the funding of the Hungarian Academy of Sciences under its Momentum Programme (LD-004/2010).

[†]Research was funded by OTKA grant K108383.

[‡]Research was funded by OTKA grant K101224 and by the Hungarian Academy of Sciences under its Momentum Programme (LD-004/2010).

can be obtained via the given tree network and the quality of which does not depend on whether the connection is direct or goes through other nodes. The aim of every player is to get connected to the root and receive that service. The cost of an arc, however, is incurred only once, no matter how many players use that link, so forming coalitions results in cost savings. The main question is how to allocate the connection costs between the players to induce cooperation.

More general versions of the cost allocation problem on fixed tree networks were considered by Granot, Maschler, Owen, and Zhu (1996) and Maschler, Potters, and Reijnierse (2010). The closest to our setting is the standard tree enterprise discussed by Maschler, Potters, and Reijnierse (2010). We also allow nodes in the network where no player resides or nodes with more than one residents. Further, we also assume non-negative costs on the arcs and zero costs on the nodes. We, however, generalize the structure of the network by assuming that it is a directed acyclic graph (DAG) in which players can have multiple routes to the root. Naturally, players that have more than one possible way to reach the root have more bargaining power when it comes down to sharing the costs.

A typical economic situation that can be modeled in this way is the cost allocation of infrastructural developments. Consider for example 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. Bergantiños, Lorenzo, and Lorenzo-Freire (2010) and Dutta and Kar (2004) provide further examples of this kind.

One of the consequence of the more general network structure is that even under the aforementioned standardization assumptions the computation of the cost of a coalition (i.e. finding the cheapest subnetwork that connects all players in the coalition to the root) amounts to solving the so-called acyclic directed Steiner tree problem[1], which is NP-hard (Hwang, Richards, and Winter, 1992). The computation of the entire cost function for all coalitions, therefore, could be prohibitive in practice. Another important consequence is that, unlike for 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. We provide a sufficient condition for non-emptiness of the core that is satisfied for a large class of directed acyclic graph games. Unlike for standard tree games, even these canonical DAG-games need not be concave. We provide further structural results

[1]Also known as the Steiner arborescence problem.

with respect to the core. We identify 'free riders' i.e. players that does not pay anything in any core allocation. Additionally, we characterize coalitions that have a constant zero excess in the core. Finally we introduce the concept of dually essential coalitions - a relatively small class of coalitions which are sufficient in themselves to determine the linear inequality system that describes the core. We firmly believe that these results could be utilized in the computation of the nucleolus, or other core-related cooperative solutions for canonical DAG-games.

Although we deal with cost allocation problems on a rooted directed network, some of our results resemble well-known properties of monotonic minimum cost spanning tree (mMCST) games that are associated with undirected networks (Bird, 1976; Granot and Huberman, 1981, 1984; Granot and Maschler, 1998). On the other hand airport games (Potters and Sudhölter, 1999) and irrigation games (Márkus, Pintér, and Radványi, 2011) are special cases of our proposed model. Shortest path games, peer group games and highway games are also very similar in their concept (Rosenthal, 2013; Brânzei, Fragnelli, and Tijs, 2002; Çiftçi, Borm, and Hamers, 2010). Note that each of these games have a non-empty core. In order to give more insight into our model let us compare airport games, standard tree games, DAG-network games, and minimum cost spanning tree games. These games have the same setup, namely they are based on a rooted graph, where players – who are located on the nodes – would like to share the construction cost of the edges. Table 1 summarizes the differences of these games, while Figure 1 shows how they are related to each other.

Game	Graph	Edges	Players/node	Convexity	Core
Airport	path	(un)directed	$0-n$	concave	non-empty
Standard Tree	tree	(un)directed	$0-n$	concave	non-empty
DAG	connected DAG	directed	$0-n$	not concave	can be empty
mMCST	connected	undirected	1	not concave	non-empty

Table 1: Comparison of graph related cost games

Notice that in case of airport games and standard tree games the edges can be considered both directed or undirected.

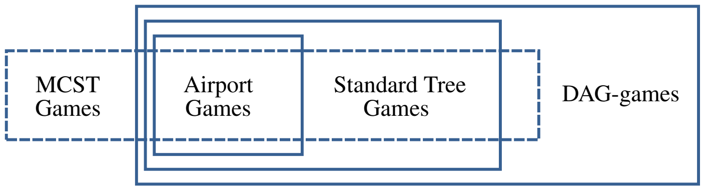

Figure 1: Venn-diagram of graph related cost games

The outline of the paper is as follows. In the second section we introduce the game theoretical framework used in the paper. In the third we formally define directed acyclic graph games. In the forth section we propose a network canonization process and describe its implications. In the fifth section we discuss the structural results with respect to the core. Finally we conclude our findings with some remarks and we review the possible directions for future research especially related to the nucleolus of the game.

2 Game theoretical framework

A *cooperative cost game* is an ordered pair (N, c) consisting of the player set $N = \{1, 2, \dots, n\}$ and a characteristic cost function $c : 2^N \to \mathbb{R}$ with $c(\emptyset) = 0$. The value $c(S)$ represent how much cost coalition S must bear if it chooses to act separately from the rest of the players. Let us denote a specific cost game by Γ. A cost game $\Gamma = (N, c)$ is said to be *concave*[2] if its characteristic function is submodular, i.e. if

$$c(S) + c(T) \geq c(S \cup T) + c(S \cap T), \quad \forall\, S, T \subseteq N.$$

A solution for a cost allocation game is a vector $x \in \mathbb{R}^N$. For convenience, we introduce the following notations $x(S) = \sum_{i \in S} x_i$ for any $S \subseteq N$, and instead of $x(\{i\})$ we simply write $x(i)$. A solution is called efficient if $x(N) = c(N)$ and individually rational if $x(i) \leq c(i)$ for all $i \in N$. The imputation set of the game $X(\Gamma)$ consists of the efficient and individually rational solutions, formally,

$$X(\Gamma) = \{x \in \mathbb{R}^N \mid x(N) = c(N),\ x(i) \leq c(i) \text{ for all } i \in N\}.$$

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 core of the cost allocation game $\mathcal{C}(\Gamma)$ is a set-valued solution where all the excesses are non-negative. Formally,

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

Simplifications could be possible in the linear system defining the core if we focus on the following two types of coalitions.

Definition 1 (Essential coalitions). *Coalition S is called* essential *in game $\Gamma = (N, c)$ if it can not be partitioned as $S = S_1 \dot\cup \dots \dot\cup S_k$ with $k \geq 2$ such that $c(S) \geq c(S_1) + \dots + c(S_k)$.*

[2]Sometimes submodular cost games are called convex instead of concave in the same way we usually speak of the core of a cost game instead of its anti-core. This terminology is appealing since for instance Kuipers's results Kuipers (1996) on convex games naturally extends to concave cost games.

Essential coalitions were introduced in (Huberman, 1980) in order to show that they form a characterization set for the nucleolus. For more on characterisation sets see (Granot, Granot, and Zhu, 1998). By definition, the singleton coalitions are always essential in every game. It is easily seen that each not essential (i.e. *inessential*) coalition has a weakly minorizing partition which consists exclusively of essential coalitions. Moreover, the core is determined by the efficiency equation $x(N) = c(N)$ and the $x(S) \leq c(S)$ inequalities corresponding to the essential coalitions, all the other inequalities can be discarded from the core system.

This observation helps us to eliminate large coalitions which are redundant for the core. We can identify the small redundant coalitions, if we apply idea of essentiality to the dual game. The *dual game* (N, c^*) of game (N, c) is defined by the coalitional function $c^*(S) := c(N) - c(N \setminus S)$ for all $S \subseteq N$. Clearly, $c^*(\emptyset) = 0$ and $c^*(N) = c(N)$.

Definition 2 (Dually essential coalitions)**.** *Coalition S is called* dually essential *in game $\Gamma = (N, c)$ if its complement can not 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 $c^*(N \setminus S) \leq c^*(N \setminus T_1) + \ldots + c^*(N \setminus T_k)$, or equivalently, $c(S) \geq c(T_1) + \ldots + c(T_k) - (k-1)c(N)$.*

Notice that each member of S appears in all of the coalitions T_1, ..., T_k, but every other player appears only in exactly $k - 1$ times in this family. We call such a system of coalitions an *overlapping decomposition* of S.[3]

By definition, all $(n-1)$-player coalitions are dually essential in any game. It is easily checked that if S and T are not dually essential coalitions and T appears in an overlapping decomposition of S, then S cannot appear in an overlapping decomposition of T, consequently, each coalition that is not dually essential (i.e. *dually inessential*) has a weakly minorizing overlapping decomposition which consists exclusively of dually essential coalitions. Moreover, the core of (N, c) can also be determined by the dual efficiency equation $x(N) = c^*(N)$ and the $x(S) \geq c^*(S)$ dual inequalities corresponding to the complements of the dually essential coalitions, all the other dual inequalities can be discarded from the dual core system.

It can be easily verified (e.g. by applying Theorem 2.3 in (Granot, Granot, and Zhu, 1998)) that dually essential coalitions characterize the nucleolus as well. We intend to construct an efficient algorithm for the nucleolus in a subsequent paper based on the structural results presented here.

[3]For a more general definition, where the complements of the overlapping coalitions need not form a partition of the complement coalition, see e.g. (Brânzei, Solymosi, and Tijs, 2005) and the references therein.

3 Definition and basic properties of the game

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 each other node of G there leads at least one directed path to the root. G is considered 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*. Junctions that have more than one leaving zero cost arcs (or simply *zero arcs*) are called *gates*.

Let N be a set of players and let $\mathcal{R} : N \to V \setminus \{\mathbf{r}\}$ 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 leafs 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 (the root is excluded), but there can be other unoccupied nodes or nodes having more than one residents. The set of residents of a subgraph T is denoted by $N(T)$, formally, $N(T) = \mathcal{R}^{-1}(V(T))$.

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. 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 allocation game that is associated with the pair $(\mathcal{D}, \mathcal{R})$, or shortly a *DAG-game* $(\mathcal{D}, \mathcal{R})$, is defined as follows.

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

The definition 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.

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 2/A 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 DAG-network $(\mathcal{D}, \mathcal{R})$ depicted in Figure 2/B. The cost of connecting any two-player coalition is 3, however $c_{(\mathcal{D},\mathcal{R})}(N) = 5$ which leaves the core empty.

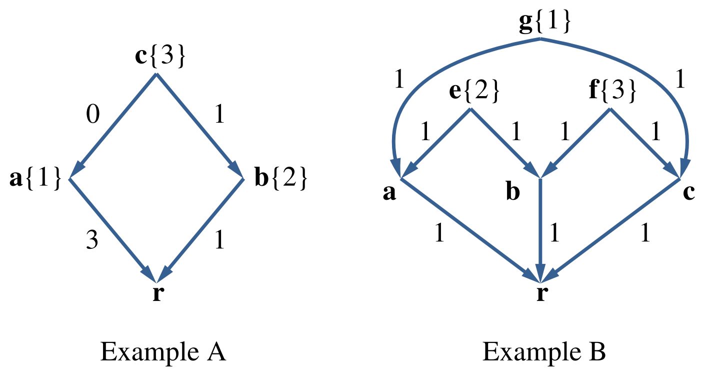

Figure 2: The first example shows that the characteristic function need not be submodular. Example B displays a DAG-network that induces a cost game with an empty core. The residents of the nodes are given in braces in both cases.

Later we will show that the condition

(*) there must be a resident 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. Notice that property (*) can be checked efficiently. In the following we will assume that (*) holds for any $(\mathcal{D}, \mathcal{R})$ network.

Finally we note that in general it is computationally hard to calculate the characteristic function value of a given coalition. Finding an element of T_S for an arbitrary $S \subset N$ is equivalent to the acyclic directed Steiner tree problem, which is – as we mentioned earlier – NP-hard.

4 The canonization process and its consequences

We say that DAG-game $\Gamma_{(\mathcal{D},\mathcal{R})}$ is in canonical form if the following properties are fulfilled:

P1 Each junction has a leaving zero arc.

P2 For each passage the cost of the leaving arc is positive.

P3 There resides a player in each passage.

P4 Each arc is used at least by one coalition.

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 $\min_{e \in A_{\mathbf{p}}} \delta(a) = \alpha_{\mathbf{p}} > 0$.

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.

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.

Finally arcs not used in any of the T_S subgraphs can be deleted, since they do not affect the characteristic function. Checking **P4** could be computationally demanding. However, we only need it to simplify the proofs, **P4** can be neglected for the algorithms.

Figure 3 illustrates the canonization process.

Our first observation summarizes the above findings.

Observation 3.

- *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 other such sets of trunks. 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

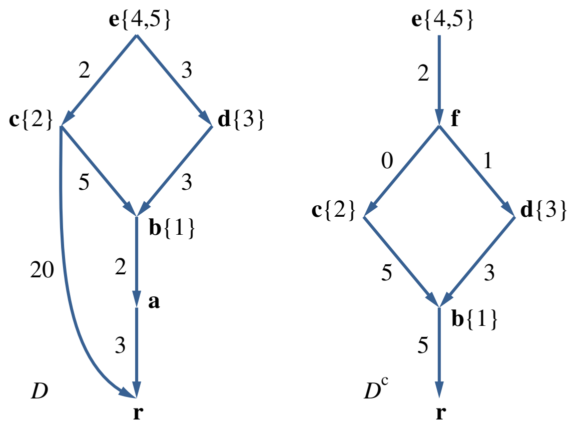

Figure 3: A DAG-network with player set $N = \{1, 2, 3, 4, 5\}$ before and after canonization.

the positive arc costs. 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.

As the residency function $\mathcal{R}$ becomes fixed after the canonization, from now on we will drop it from the notation and simply write $c_{\mathcal{D}}$. We will denote by $\Gamma_{\mathcal{D}}$ the cost game induced by $c_{\mathcal{D}}$, i.e. $\Gamma_{\mathcal{D}} = (N, c_{\mathcal{D}})$. Let us now see some consequences of canonization. We also need to introduce further notions and notations.

For each node $\mathbf{p}$, the cheapest arcs in $A_{\mathbf{p}}$ are called T_N-arcs. The name comes from the fact that (if **P1** holds) 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 there exists a shortcut between $\mathbf{p}$ and $\mathbf{q}$ it is always cheaper than any alternative path between these two nodes due to **P4** and the non-negativity of the arc costs (hence the name). 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. Note that T_N is not necessarily a tree as it may contain some additional zero arcs[4]. Secondly, T_N 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 the direct ancestor is unique (gates have more than one parent). If $\mathbf{p}$ is a parent of $\mathbf{q}$ then $\mathbf{q}$ is referred as a *direct*

[4]Unlike other trunks, T_N can be constructed efficiently in linear time. 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 every 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.

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 and notations with some examples. Consider again the canonized DAG-network $\mathcal{D}^c$ depicted in Figure 3. The only shortcut in $\mathcal{D}^c$ is the one that connects node $\mathbf{f}$ with node $\mathbf{d}$. All the other arcs are T_N-arcs. The full branch $B_{\mathbf{d}}$ contains only node $\mathbf{d}$, since $\mathbf{d} \not\prec \mathbf{f}$. Furthermore, $B_{\mathbf{d}}$ is a proper branch, for removing $\mathbf{d}$ together with the entering and leaving arcs the graph is still a trunk. Finally, the node set that corresponds to the trunk $T_{\{1,3,4\}}$ is $V \setminus B_{\mathbf{c}}^{\mathbf{f}}$ and $c_{\mathcal{D}^c}(\{1,3,4\}) = 11$.

Finally we conclude this chapter with a representation lemma that helps us visualize the graph structure of trunks.

Lemma 4. *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 $Q_1, \ldots, Q_k \subset V$ and $\mathbf{p}_1, \ldots, \mathbf{p}_k \in V$ 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 $j \in \{1, 2, \ldots, k\}$.

Proof. Any trunk T has a representation where $V(T)$ is obtained by removing branches from V. This is trivial as 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 $\{\mathbf{p}_1, \ldots, \mathbf{p}_k\} \subset V \setminus V(T_S)$ 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 $j = 1, \ldots, k$.

Due to the definition of T_S there exists at least one such passage. Let us remind the reader that T_S is the trunk that has *maximum number of arcs* among the cheapest subgraphs that connect S to the root. Therefore any junction that connects to a such trunk with a zero arc by definition is included in T_S even if no player of S resides there. 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 $V(T_S) = V \setminus (\cup_{j=1}^k B_{\mathbf{p}_j}^{Q_j})$.

□

The obtained $V \setminus \cup_{j=1}^k B_{\mathbf{p}_j}^{Q_j}$ expression is called the *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.

5 The core of the canonized DAG-game

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.

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

In our first lemma we show that the core of a canonized DAG-network game is never empty.

Lemma 5. *$\mathcal{C}(\Gamma_{\mathcal{D}}) \neq \emptyset$ for any DAG-network $\mathcal{D}$ in canonical form.*

Proof. 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 $\mathbf{p}$. We claim that $\hat{x}$ is a core allocation. Let $V^\star \subseteq V$ denote the set of occupied nodes in G and let $B \subset V$ be arbitrary. Note that unoccupied nodes can only be junctions, which have a leaving zero arc, i.e. $\delta(a_{\mathbf{p}}) = 0$ for all $\mathbf{p} \in B \setminus V^\star$. Then

$$\hat{x}(N(B)) = \sum_{\mathbf{p} \in B \cap V^\star} |N(\mathbf{p})| \cdot \frac{\delta(a_{\mathbf{p}})}{|N(\mathbf{p})|} = \sum_{\mathbf{p} \in B \cap V^\star} \delta(a_{\mathbf{p}}) + \sum_{\mathbf{p} \in B \setminus V^\star} \delta(a_{\mathbf{p}}) = \tau(B,N). \quad (1)$$

In conclusion, $\hat{x}(N(B)) = \tau(B,N)$ for any node set B. In particular, $\hat{x}(N) = \tau(V,N) = c_{\mathcal{D}}(N)$. On the other hand, for any $S \subseteq N$

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

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. $\square$

The standard allocation is similar to the Bird-rule which was proposed for MCST games (Bird, 1976). There is an extensive literature devoted to this type of rules. 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 case when these two paths completely coincide is also included in this definition. Thus if there leads 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.

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[5]). Notice that a junction can not be a principal ancestor of any of its descendants. The only principal ancestor that is not a passage is the root.

Definition 6. *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 every free node is a junction. 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.

Lemma 7. *Let* $B^Q_{\mathbf{p}}$ *be any branch originating from node* $\mathbf{p}$. *If* $exc(N(V \setminus B_{\mathbf{p}}), y) = 0$ *for any core allocation* y, *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

[5]In the Appendix, we provide an efficient algorithm that finds the principal ancestor of each node in a DAG-network.

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

would contradict the non-negativity of excesses. □

Naturally nodes that can reach the root via a zero cost path are free, but there are less obvious instances. The next theorem gathers the type of nodes that are free.

Theorem 8. *Node* $\mathbf{p}$ *belongs to* F *if and only if* $\Pi(\mathbf{p}) = \mathbf{r}$.

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

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

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

where $\varepsilon > 0$ is a sufficiently small real number ($\varepsilon = \frac{\min_{a \in A} \delta(a)}{|N|+1}$ will do). 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

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

where 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

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

where we used that $\hat{x}(S \setminus i_{\mathbf{p}}) \leq \tau(\mathcal{R}(S), N)$ 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 can not 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 a slightly stronger statement.

Lemma 9. *If* $\Pi(\mathbf{p}) = \mathbf{r}$ *then* $\mathbf{p}$ *is free and* $exc(N(V \setminus B_{\mathbf{p}}), x) = 0$ *for any* $x \in \mathcal{C}(\Gamma_{\mathcal{D}})$.

Let $d(\mathbf{q})$ denote the size 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 one of $\mathbf{p}$'s parent is free. Let this node be denoted by $\mathbf{f}$ (see Figure 4, Example I.) and 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.

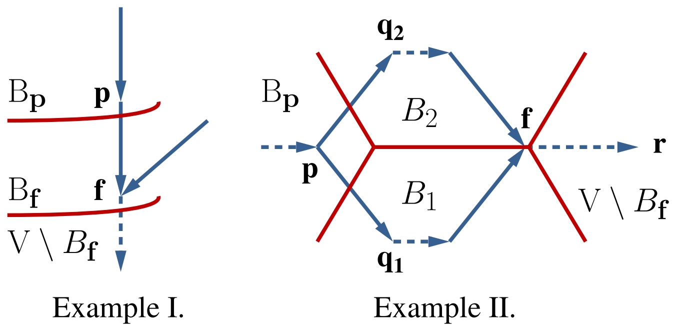

Figure 4: Subgraphs of T_N. Dashed lines indicate a path or paths.

The second case is when none of $\mathbf{p}$'s parent is free. As the principal ancestor of $\mathbf{p}$ is the root $\mathbf{p}$ must be a gate. There leads 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 $\mathbf{f}$ (see Figure 4, Example II.). 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 leads 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 7. 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 $\tau(B_1, N) = y(N(B_1))$. As $N(B_1)$ 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 7 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, i.e. $\mathbf{p}$ is free. This concludes the proof of Lemma 9 and Theorem 8. $\square$

A coalition S is said to be *saturated* if $i \in S$ whenever $c(S) = c(S \cup \{i\})$. Granot, Granot and Zhu proved that saturated coalitions together with the grand coalition and the $n-1$ player coalitions characterize the nucleolus of any monotone cost game (Granot, Granot, and Zhu, 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 the light of these two result we may restrict our attention to this type of coalitions. In case of DAG-games this property comes with a nice structure. Saturated coalitions incorporate every player of the trunk on which they reside, 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

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

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,
- $B_{\mathbf{p}}^Q$ does not contain a free node.

Lemma 10. *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.*

Proof. Since $\pi(\mathbf{p})$ is free, it is a junction and $x(N(\pi(\mathbf{p}))) = 0$. We know from Lemma 9 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 7 it can be shown that $x(N(B^Q_{\mathbf{p}})) \leq \tau(B^Q_{\mathbf{p}}, N)$.

Each node of Q has (at least) two semi-arc-disjoint paths that leads to the root. As $B^Q_{\mathbf{p}}$ does not contain a free node one of these paths for each node by-passes $B^Q_{\mathbf{p}}$. 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

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

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

Lemma 11. *Let $\cup_{j=1}^k B_{\mathbf{p}_j}^{F_j}$ 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 $\cup_{j=1}^k B_{\mathbf{p}_j}^{F_j}$ 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 $\cup_{j=1}^k B_{\mathbf{p}_j}^{F_j}$ consist of a single node, then $k = 1$ and $B_{\mathbf{p}_1}^{F_1}$ must be a building block. Now suppose the lemma is true for node sets with less than l nodes and let $|\cup_{j=1}^k B_{\mathbf{p}_j}^{F_j}| = l$. Let $B_{\mathbf{p}_1}^{Q}$ be a branch where $Q = B_{\mathbf{p}_1} \cap F$ and let $B_{\mathbf{p}_1}^{Q'}$ be the standard form of this branch. Note that $B_{\mathbf{p}_1}^{Q'}$ is a building block and it is a subset of $B_{\mathbf{p}_1}^{F_1}$. Let us delete $B_{\mathbf{p}_1}^{Q'}$ from $B_{\mathbf{p}_1}^{F_1}$. If $Q' \cap B_{\mathbf{p}_1}^{F_1}$ 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 $B_{\mathbf{p}_1}^{F_1}$ then those must be passages. Let us denote these by $\mathbf{p}'_1, \dots, \mathbf{p}'_K$. Note that $\pi(\mathbf{p}'_1), \dots, \pi(\mathbf{p}'_K) \in F$. Hence the remaining nodes can be written as $\cup_{i=1}^K B_{\mathbf{p}'_i}^{F_1} \cup_{j=2}^k B_{\mathbf{p}_j}^{F_j}$. By reindexing $\mathbf{p}'_i$ we are done as $|\cup_{i=1}^K B_{\mathbf{p}'_i}^{F_1} \cup_{j=2}^k B_{\mathbf{p}_j}^{F_j}| < l$. □

Now we are ready to characterize the set $\mathcal{S}_0$.

Theorem 12. *$S \in \mathcal{S}_0$ if and only if $V(T_S)$ can be written as*

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

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 the light of Lemma 10 and Lemma 11 the *only if* part can be verified easily. If the trunk of coalition S can be represented as $V(T_S) = V \setminus \cup_{j=1}^k B_{\mathbf{p}_j}^{F_j}$, 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. $S \in \mathcal{S}_0 \Rightarrow V(T_S) = V \setminus \cup_{j=1}^k B_{\mathbf{p}_j}^{F_j}$. From Lemma 4 we know that we can choose a representation of $V(T_S)$ where $\mathbf{p}_j$ – the origin of the removed $B_{\mathbf{p}_j}^{F_j}$ 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 $V(T_S) = V \setminus B_{\mathbf{p}}^{Q}$. 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 8. 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 $N(V \setminus B_{\mathbf{p}}^{Q})$ was zero under the standard allocation it is not zero under y. Now let $V(T_S) = V \setminus \cup_{j=1}^{k} B_{\mathbf{p}_j}^{F_j}$ 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 leads a path in T_N from $\mathbf{f}$ to $\pi(\mathbf{p}_j)$ through $B_{\mathbf{p}_j}^{F_j}$. There leads 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 leads 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 12 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 8), 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 Lemmas 9 and 7). A natural simplification would be to contract the free nodes with the root. Unfortunately this transformation would alter the characteristic function of the game, therefore we follow another approach to describe the core.

The next lemma gives an upper bound on how much certain branches are willing to pay in the core. Let a_s be a shortcut 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 13. *Let $\mathbf{p}$ be a junction with a critical shortcut $a_s \in A_{\mathbf{p}}$. If $B_{\mathbf{p}}^{Q}$ is a $\mathbf{p}$-branch then $x(N(B_{\mathbf{p}}^{Q})) \leq \tau(B_{\mathbf{p}}^{Q}, N) + \delta(a_s)$ for any core allocation x.*

Proof. If we replaced a_s with a zero arc, there would exist two semi-arc-disjoint paths from $\mathbf{p}$ to the root. One that leads through an original zero arc of $\mathbf{p}$, and one through a_s. We will use a similar argument as in Lemma 9. We color the nodes of the former path

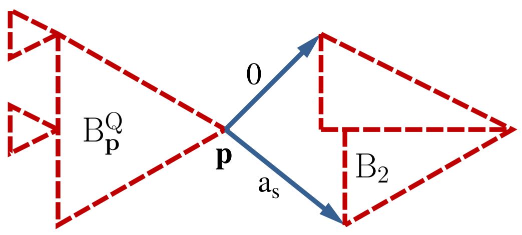

Figure 5: Schematic picture of $\mathcal{D}$. Dashed lines indicate branches.

red while the nodes of the latter path blue. The nodes contained in both paths (e.g. the root) are assigned both colors, except for $\mathbf{p}$ that is painted only red. Then we color each node in V one-by-one 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 we *paint it red*. Among the possible colorings we chose one where every node in $B^Q_{\mathbf{p}}$ was painted red. Let B_1 contain the red nodes, while B_2 the blue ones. Every node has been assigned at least one color i.e. $B_1 \cup B_2 = V$. The intersection of B_1 and B_2 contains nodes that coincide on the red and the blue paths. These nodes are free by construction. In $T_{N(B_1)}$ and $T_{N(B_2)}$ every player can reach the root by using only arcs of T_N. Thus if x is an arbitrary core allocation, then

$$\begin{aligned} c_{\mathcal{D}}(N) &= c_{\mathcal{D}}(N(B_1)) + c_{\mathcal{D}}(N(B_2)), \\ c_{\mathcal{D}}(N) - x(N) - x(N(B_1 \cap B_2)) &= c_{\mathcal{D}}(N(B_1)) - x(N(B_1)) + c_{\mathcal{D}}(N(B_2)) - x(N(B_2)), \\ 0 &= exc(N(B_1), x) + exc(N(B_2), x), \end{aligned}$$

where the last equality comes from the fact that $x(N(B_1 \cap B_2)) = 0$, as $(B_1 \cap B_2) \subset F$. From the non-negativity of the excesses we obtain that $exc(N(B_2), x) = 0$. Finally

$$\begin{aligned} &0 \le c_{\mathcal{D}}(N(B_2 \cup B^Q_{\mathbf{p}})) \le c_{\mathcal{D}}(N(B_2)) + \delta(a_s) + \tau(B^Q_{\mathbf{p}}, N), \\ &0 \le exc(N(B_2 \cup B^Q_{\mathbf{p}}), x) \le c_{\mathcal{D}}(N(B_2)) - x(N(B_2)) + \delta(a_s) + \tau(B^Q_{\mathbf{p}}, N) - x(N(B^Q_{\mathbf{p}})), \\ &0 \le 0 + \delta(a_s) + \tau(B^Q_{\mathbf{p}}, N) - x(N(B^Q_{\mathbf{p}})). \end{aligned}$$

□

Next we uncover the graph structure of dually essential coalitions. As it will turn out it is simple and easy to deal with. First we show that dual essentiality is a stricter property than saturatedness.

Lemma 14. *In a DAG-network game dually essential coalitions are 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 can not be dually essential. □

The following theorem characterizes dually essential coalitions.

Theorem 15. *The dually essential coalitions of the cost game $\Gamma_{\mathcal{D}}$ are the coalitions with $n-1$ player 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 14 that only saturated and $n-1$ player coalitions are dually essential. By Lemma 4 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. Note that in the standard representation of $V(T_S)$, each of the Q_j node sets is either empty or a subset of $V(T_S)$.

For convenience's sake let us introduce the following notation $B_1 = \cup_{j=1}^{k-1} B^{Q_j}_{p_j}$ and $B_2 = B^{Q_k}_{p_k}$. Then let $S_1 = N \setminus N(B_1)$ and $S_2 = N \setminus N(B_2)$. In this way $S_1 \cup S_2 = N$ and $S_1 \cap S_2 = S$. 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 $B^{Q_k}_{p_k}$ 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. □

Notice that Theorem 15 is surprisingly analogous to the one derived by Maschler, Potters, and Reijnierse (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 result is unquestionable.

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.

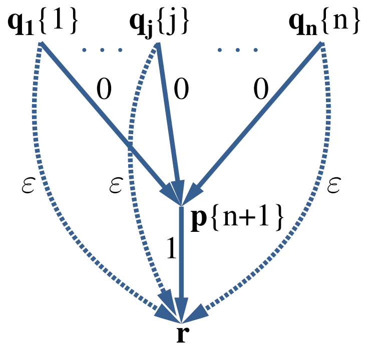

Figure 6: A DAG-network with exponential many proper branches. Solid lines indicate T_N-arcs, while dotted lines are shortcuts.

Consider the DAG-network depicted in Figure 6. 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 player. 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.

6 Concluding remarks

In this paper we introduced a new class of cooperative cost game that is based on directed acyclic graph networks. We analyzed the properties of the game and gave sufficient conditions for the non-emptiness of the core. We identified 'free riders', i.e. players that does not pay anything in any core allocation. Additionally, we characterized coalitions that have a constant zero excess in the core. We also introduced the concept of dually essential coalitions - a class of coalitions which are sufficient in themselves to determine the linear inequality system that describes the core or the nucleolus.

Considering their structure and complexity directed acyclic graph games lie somewhere between standard tree games and monotonic minimum cost spanning tree games. There is a vast amount of literature concerning both of these class of games. It is an interesting question how the known results relate to DAG-games. One difficulty that arises with the appearance of shortcuts – i.e. when we extend the network structure from standard tree to a directed acyclic graph – is that determining the cost of a coalition becomes computationally hard. Finding the cheapest trunk that connects a set of nodes to the root is equivalent to the so called acyclic directed Steiner tree problem, which is known to be NP-hard. That is, even if the cardinality of the dually essential coalitions is polynomially bounded from above, we will not be able to efficiently determine the core by a linear program. Nevertheless, we believe that the structural results presented here, especially the characterization of the set of free nodes and the $\mathcal{S}_0$ set, will compose the basis of any further analysis that focuses on the core and related allocations.

The complexity issue related to the computation of the characteristic function also emerges for monotonic minimum cost spanning tree games. For this latter class of games finding the nucleolus is in itself NP-hard (Faigle, Kern, and J., 1998). It seems that the hardness comes from the undirectedness of the edges, and its unrelated to how many players reside in a node. In a DAG-network payments flow in one direction, toward the root which makes the players hierarchically structured. Thus it seems possible that some kind of painting algorithm works for DAG-networks (Maschler, Potters, and Reijnierse, 2010). In a subsequent paper we will provide an efficient algorithm that finds the nucleolus for canonized directed acyclic graph games.

A possibility to expand the model is to consider more service provider, i.e. more than one root in the graph. Since the free nodes are already behaving like some kind of secondary roots, this will not change the character or difficulty of the problem. Our conjecture is that contracting the free nodes with the root results in a game where the core and the nucleolus is unchanged. However this transformation alters the characteristic function of the game.

References

Bergantiños, G., L. Lorenzo, and S. Lorenzo-Freire (2010): "The family of cost monotonic and cost additive rules in minimum cost spanning tree problems," *Social Choice and Welfare*, 34, 695–710.

Bergantiños, G., and J. J. Vidal-Puga (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.

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

Brânzei, R., V. Fragnelli, and S. Tijs (2002): "Tree-connected peer group situations and peer group games," *Mathematical Methods of Operations Research*, 55, 93–106.

Brânzei, R., T. Solymosi, and S. Tijs (2005): "Strongly essential coalitions and the nucleolus of peer group games," *International Journal of Game Theory*, 33, 447–460.

Çiftçi, B., P. Borm, and H. Hamers (2010): "Highway games on weakly cyclic graphs," *European Journal of Operational Research*, 204(1), 117–124.

Dutta, B., and A. Kar (2004): "Cost monotonicity, consistency and minimum cost spanning tree games," *Games and Economic Behavior*, 48, 223–248.

Faigle, U., W. Kern, and K. J. (1998): "Computing the Nucleolus of Min-cost Spanning Tree Games is NP-hard," *International Journal of Game Theory*, 27.

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

Granot, D., and G. Huberman (1981): "Minimum Cost Spanning Tree Games," *Mathematical Programming*, 21, 1–18.

——— (1984): "On the core and nucleolus of minimum cost spanning tree games," *Mathematical Programming*, 29(3), 323–347.

Granot, D., M. Maschler, G. Owen, and W. R. Zhu (1996): "The kernel/nucleolus of a standard tree game," *International Journal of Game Theory*, 25, 219–244.

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

Huberman, G. (1980): "The nucleolus and essential coalitions," in *Analysis and Optimization of Systems*, ed. by A. Bensoussan, and J. L. Lions, vol. 28 of *Lecture Notes in Control and Information Sciences*, pp. 416–422. Elsevier B.V.

Hwang, F. K., D. S. Richards, and P. Winter (1992): *The Steiner Tree Problem*, vol. 53 of *Annals of Discrete Mathematics*. North-Holland.

Kuipers, J. (1996): "A polynomial time algorithm for computing the nucleolus of convex games," Report m 96-12, University of Maastricht.

Márkus, J., M. Pintér, and A. Radványi (2011): "The Shapley value for airport and irrigation games," *Mathematical Programming*.

Maschler, M., J. Potters, and H. Reijnierse (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," *Mathematics of Operations Research*, 3(3), 189–196.

Potters, J., and P. Sudhölter (1999): "Airport Problems and Consistent Allocation Rules," *Mathematical Social Sciences*, 38, 83–102.

Rosenthal, E. C. (2013): "Shortest Path Games," *European Journal of Operation Research*, 224, 132–140.

Trudeau, C. (2012): "A new stable and more responsive cost sharing solution for minimum cost spanning tree problems," *Games and Economic Behavior*, 75, 402–412.

Appendix A

Let us provide a simple method to identify the principal ancestor of each node in T_N. Double every zero arc in T_N and set the capacity of every arc to 1. Now determining the key ancestors of a node $\mathbf{p}$ becomes easy. Let $f_{\mathbf{pq}}$ denote the maximum flow between $\mathbf{p}$ and $\mathbf{q}$. If $f_{\mathbf{pq}} \geq 2$, then $\mathbf{q}$ is a key ancestor of $\mathbf{p}$. The key ancestor closest to the root will be the principal ancestor. Note that **P4** is not needed as the algorithm works with T_N

This is a very costly way to map the principal ancestors of the nodes, it takes around $O(m^5)$ time, where m denotes the number of nodes in G. It seems likely that by dynamic programming the running time can be reduced to $O(m^3)$ or even lower. However our goal was to prove that this problem can be solved in polynomial time. We leave the question of efficiency to future research.