Ann Math Artif Intell (2013) 67:1–30
DOI 10.1007/s10472-012-9323-9

Proof systems and transformation games

Yoram Bachrach · Michael Zuckerman · Michael Wooldridge · Jeffrey S. Rosenschein

Published online: 16 November 2012

Abstract We introduce *Transformation Games* (TGs), a form of coalitional game in which players are endowed with sets of initial resources, and have capabilities allowing them to derive certain output resources, given certain input resources. The aim of a TG is to generate a particular target resource; players achieve this by forming a coalition capable of performing a sequence of transformations from a combined set of initial resources to the target resource. TGs can model a number of natural settings, such as cooperative proof systems, where a collection of agents having different expertise work together to derive a proof for a target theorem, or supply chains, where agents cooperate to create a target product from base resources. After presenting the TG model, and discussing its interpretation, we consider possible restrictions on the transformation chain, resulting in different coalitional games. Following the basic model, we consider the computational complexity of several problems in TGs, such as testing whether a coalition wins, checking if a player is a dummy or a veto player, computing the core of the game, computing power indices, and checking the effects of possible restrictions on the coalition. Finally, we consider extensions to the model in which transformations have associated costs.

This is an extended version of a conference paper with the same title [21], which was presented at the 35th International Symposium on Mathematical Foundations of Computer Science (MFCS 2010), in Brno, Czech Republic. This new version contains additional results regarding computing the Shapley value in Transformation Games as well as additional results on Transformation Games with a bounded number of resources and transformations. We have also expanded our examination of how the Transformation Game model can capture multiagent interactions in both supply chains and proof systems, and included revised and extended proofs.

Y. Bachrach (✉)
Microsoft Research, Cambridge, UK
e-mail: yorambac@gmail.com

M. Zuckerman · J. S. Rosenschein
Hebrew University, Jerusalem, Israel

M. Wooldridge
University of Oxford, Oxford, UK

Keywords Cooperative game theory · Supply chains · The core · The Shapley value · Power indices

Mathematics Subject Classifications (2010) 91A12 · 68Q25

1 Introduction

Many multiagent systems crucially rely on cooperation among agents. However, in many such systems agents are also selfish, and act in their own interests. A key tool in analysing interaction in such environments is game theory. Due to the rising prominence of multiagent systems within the field of artificial intelligence, many aspects of game theory have been studied in recent years, and game theoretic techniques have been applied in domains such as electronic commerce, auctions, voting, supply chains, and resource allocation [71]. Cooperation is key in many such domains, and cooperative game theory provides a mathematical framework that enables the analysis of agent behaviour in such domains.

Cooperative game theory considers questions such as which coalitions are likely to form, and how the benefits of cooperation should be distributed between coalition members. These notions are formalised as *solution concepts* such as the core [46], the Shapley value [68], the Banzhaf index [22] and the nucleolus [67] (see, e.g., [25] for a detailed discussion). An important question is the extent to which these solutions can be effectively computed; thus, solutions from cooperative game theory have also been studied by computer scientists [18, 28, 36].

We consider a new model of cooperative activity among self-interested players. In a *Transformation Game* (TG), players must cooperate to generate a certain target resource.[1] In order to generate the resource, each player is endowed with a certain set of initial resources, and in addition, each player is assumed to be capable of *transformations*, allowing it to generate a certain resource, given the availability of a certain input set of resources required for the transformation. Coalitions may thus form *transformation chains* to generate various resources. A coalition of players is successful if it manages to form a transformation chain that eventually generates the target resource. Forming such chains is typically complicated, as there are usually constraints on the structure of the chain. One example of such a constraint is a time restriction, in the form of a deadline. Even when there is no deadline, short chains are typically preferred, since we might expect that the more transformations a chain has, the higher the probability of some transformation failing.

We consider a number of natural restrictions on transformation chains, and investigate the effect of these restrictions on solution concepts and the complexity of computing them. Specifically, we study three types of domains: *unrestricted domains*, where there is no restriction on the chain; *makespan domains*, where each transformation requires a certain amount of time, and the coalition must generate

[1]We use the terms "product" and "resource" interchangeably. Both of these simply refer to items in the set of all items. We usually refer to an item as a "product" if it is generated as a result of some transformation, and as a "resource" if it is the input to a transformation. Of course, a product of one transformation can be one of the resources used to derive a subsequent product.

the target resource before a certain deadline; and *limited transformation domains*, where the coalition must generate the target resource without performing more than a certain number of transformations. We also consider two types of transformations: *simple* transformations, where a transformation models the production of a single output resource from a single input resource, and *complex* transformations, where a transformation may require a *set* of input resources to generate an output resource.

1.1 Motivating examples

We consider two main motivations for our model of transformation game (TGs): supply chains and cooperative proof systems.

1.1.1 Supply chains

Consider a number of firms, each of which produces a different product from some base resources. In many cases, the outputs produced by one firm are used as input resources for another firm. For example, one firm may drill for crude oil, another may produce refined oil from crude oil, another may package refined oil in barrels for transportation, and yet another firm may transform refined oil into petrol for cars sold to consumers. Thus, these companies form a chain that generates an output product sold to consumers from various base resources. In such cases, some of the firms may buy resources from several firms, or be capable of generating more than one target product. Several key questions arise in such economic domains. First, there may be several possible chains that generate the ultimate product sold to consumers. Which of these chains are more likely to be formed? Second, how would firms share the revenue obtained from selling the product to consumers? Which contracts are likely to form among the firms? Finally, which firm is most important in generating the output product? How can we quantify its importance and bargaining power with regard to other firms in the market?

Our TG model enables the answering of such questions in *simplified domains*. TGs model some important features of the domain, such as the inputs and outputs of the firms, and constraints regarding the costs of transformations or deadlines. However, TGs also omit some features of the domains, such as quantities of resources and products, and consumption of resources during the transformations. The simplicity of the TG framework makes it possible to compute some solutions in polynomial time. Despite these positive results, we show that even such abstract domains are combinatorially rich enough such that some interesting problems are computationally hard. Thus, although TGs offer only a simplified model of supply chains, they can shed light on the behaviour of selfish agents in such domains. TGs also complement other related work on planning within artificial intelligence, which we discuss in Section 7.

1.1.2 Cooperative proof systems

TG can be viewed as a strategic, game-theoretic formulation of *proof systems*. In a formal proof system, the goal is to derive some logical statement from some logical premises by applying logical inference rules. When modelled as a TG, premises and proof rules are distributed across a collection of agents, and proof becomes a cooperative process, with different agents contributing their domain expertise (premises) and capabilities (proof rules). In such a setting, game theoretic solution concepts such as the Banzhaf index provide a measure of the relevant significance of agents (and hence premises and proof rules) in the proof process. Viewed in this way, TGs provide a formal foundation for cooperative theorem proving systems such as those described in [32, 41], as well as cooperative problem solving systems in general [53].

As a concrete example, consider a set of experts who may cooperate to prove a complex mathematical theorem. Each expert knows facts and proofs for various simpler theorems that are likely to assist in proving the complex theorem, but none of the experts can prove the complex theorem on her own. Suppose a coalition of such experts decides to work together, and each of the participants shares all her knowledge. Assuming that the joint knowledge of the coalition suffices to prove the complex theorem, how can they decide who contributed most to generating the proof? In other words, in domains where no single expert can generate a proof, but various *teams* of experts can do so, how can we quantify (in some sense) the contribution of each expert? TGs can be used to quantify agent contributions through the use of game theoretic tools (though, of course, in this scenario other unmodelled considerations might also apply).

2 Preliminaries

We briefly discuss the basic game theoretic concepts that are later used in the context of TGs (see, e.g., [25, 59, 62] for more detailed introductions). A *transferable utility coalitional game* is composed of a set $I = \{a_1, \ldots, a_n\}$ of n players and a characteristic function $v : 2^I \to \mathbb{R}$ that maps any subset (coalition) of the players to a real number. The value $v(C)$ is the total utility the players $C \subseteq I$ can obtain together. The coalition I of all the players is called the *grand coalition*. Often such games are *increasing*, i.e., for all coalitions $C' \subseteq C$ we have $v(C') \leq v(C)$. In *simple* transferable utility coalitional games, v only gives values of 0 or 1 (i.e., $v : 2^I \to \{0, 1\}$), and in this case we say $C \subseteq I$ *wins* if $v(C) = 1$ and *loses* otherwise. We say player i is *critical* in a winning coalition C if the removal of i from that coalition would make it a losing coalition: $v(C) = 1$ and $v(C \setminus \{i\}) = 0$.

The characteristic function defines the value a coalition can obtain, but does not indicate how to distribute this value among the players within the coalition. An *imputation* $(p_1, \ldots, p_n)$ is a division of the gains of the grand coalition among all players, where $p_i \in \mathbb{R}$, such that $\sum_{i=1}^{n} p_i = v(I)$. We call p_i the payoff of player a_i, and denote the payoff of a coalition C as $p(C) = \sum_{i \in \{j | a_j \in C\}} p_i$.

Cooperative game theory offers a number of solution concepts, defining imputations that are likely to occur. A minimal requirement of an imputation is *individual-rationality* (IR): for every player $a_i \in C$, we have $p_i \geq v(\{a_i\})$. Thus, individual rationality says that an agent must receive at least as much payoff as it could obtain by working alone. Extending IR to coalitions, we say a coalition B *blocks* the imputation $(p_1, \ldots, p_n)$ if $p(B) < v(B)$. If a blocked imputation is chosen, the grand coalition is *unstable*, since the blocking coalition can do better by working without the other players. The most studied solution concept relating to stability is the core. The core of a game is the set of all imputations $(p_1, \ldots, p_n)$ that are not blocked by any coalition, so for any coalition C we have $p(C) \geq v(C)$.

In general, the core can contain multiple imputations, and can also be empty. Another solution, which defines a *unique* imputation, is the Shapley value. The Shapley value of a player depends on his marginal contribution over all possible coalition permutations. We denote by π a permutation (ordering) of the players, so $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ and π is reversible, and by Π the set of all possible such permutations. Denote by $S_\pi(i)$ the predecessors of i in π, so $S_\pi(i) = \{j \mid \pi(j) < \pi(i)\}$. The Shapley value is then given by the imputation $sh(v) = (sh_1(v), \ldots, sh_n(v))$ where

$$sh_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} [v(S_\pi(i) \cup \{i\}) - v(S_\pi(i))]$$

An important application of the Shapley value is that of power indices, which try to measure a player's ability to change the outcome of a game, and are used for example to measure political power. A game theoretic concept that is also used to measure power is the *Banzhaf power index*, which depends on the number of coalitions in which a player is critical, out of all the possible coalitions. The Banzhaf power index is given by: $\beta(v) = (\beta_1(v), \ldots, \beta_n(v))$ where:

$$\beta_i(v) = \frac{1}{2^{n-1}} \sum_{S \subseteq I | a_i \in S} [v(S) - v(S \setminus \{i\})]$$

More generally, power indices (of which the Shapley value and Banzhaf index are specific examples) are game theoretic tools that enable the finding of a *fair* division of the utility achieved among the participants in the game. They can be characterized as solutions that fulfill various fairness axioms, such as giving equal shares to players that are equivalent in the game, giving a share of zero to players that have no influence on the outcome of the game (for any sub-coalition), etc. Alternatively, they can be viewed as game theoretic tools for measuring the relative *importance* of agents in the game, based on their impact on the revenue generated under various assumptions on the coalition formation process. For a more complete discussion of these issues, see [1, 59].

3 Transformation games

Transformation games (TGs) involve a set of players, $I = \{a_1, \ldots, a_n\}$, a set of *resources* $R = \{r_1, \ldots, r_k\}$, and a certain *goal resource* $r_g \in R$. It is assumed that each player a_i is endowed with a set of resources $R_i \subseteq R$. Players have *capabilities* that allow them to generate a target resource when they have certain input resources. We model these capabilities via *transformations*. A transformation is a pair $\langle B, r \rangle$ where B is a subset $B \subseteq R$, indicating the resources required for the transformation, and $r \in R$ is the resource generated by the transformation. We denote the set of all such possible transformations (over R) by D. The capabilities of each player a_i are given by a set $D_i \subseteq D$. We say a transformation $d = \langle B, r \rangle$ is *simple* if $|B| = 1$ (i.e., it generates a target resource given a *single* input resource), and *complex* if $|B| > 1$. At this point, it is worth mentioning some caveats:

- First, our model of TGs has no notion of resource *quantity*. For example, the TG framework cannot explicitly express constraints such as "4 nails and 5 pieces of wood are required to build a table."

- Second, we do not model resource *consumption*: thus when a player generates a resource from base resources, the player ends up with *both* the base resources *and* the generated resource. This may at first sight seem a strange modelling choice, but it is very natural in many settings; for example, as we will see in Section 3.3, it makes sense when modelling *cooperative logical proofs*.
- Third, in the basic model, there is no *cost* associated with a transformation. Thus, performing a transformation does not reduce the utility obtained by the coalition, and players do not incur costs for performing a transformation. (In Section 6, we drop this assumption, and present a model which allows for such costs.)

Bringing these components together, a TG, Γ, is a structure

$$\Gamma = \langle I, R, R_1, \ldots, R_n, D_1, \ldots, D_n, r_g \rangle$$

where:

- I is a set of players ($|I| = n$);
- R is a set of resources ($|R| = k$);
- for each $a_i \in I$, $R_i \subseteq R$ is the set of resources with which player a_i is initially endowed;
- for each $a_i \in I$, $D_i \subseteq D$ is the set of transformations that player a_i can carry out; and
- $r_g \in R$ is a resource representing the goal of the game.

We sometimes consider transformations that require a certain amount of time. In such settings, let a_i be a player with capability $d \in D_i$. We denote the time that player a_i needs in order to perform the transformation as $t_i(d) \in \mathbb{N}$.

3.1 Transformations

Given a TG, we can define the set of resources a coalition $C \subseteq I$ can derive. We say a coalition C is endowed with a resource r, and denote this by $has(C, r)$, if there exists a player $a_i \in C$ such that $r \in R_i$. We denote the set of resources a coalition is endowed with as $R_C = \{r \in R \mid has(C, r)\}$. We now define an infix relation $\Rightarrow\ \subseteq 2^I \times R$, with the intended interpretation that $C \Rightarrow r$ means that coalition C can produce resource r. We inductively define the relation $\Rightarrow$ as follows. We have $C \Rightarrow r$ iff either:

- $has(C, r)$ (i.e., the coalition C is directly endowed with resource r); or else
- for some $\{r_{b_1}, r_{b_2}, \ldots, r_{b_m}\} \subseteq R$ we have

  $$C \Rightarrow r_{b_1}, C \Rightarrow r_{b_2}, \ldots, C \Rightarrow r_{b_m}$$

  and for some player $a_i \in C$ we have

  $$\left\langle \{r_{b_1}, r_{b_2}, \ldots, r_{b_m}\}, r \right\rangle \in D_i.$$

3.2 Unrestricted and restricted TGs

We now show how a TG $\Gamma = \langle I, R, R_1, \ldots, R_n, D_1, \ldots, D_n, r_g \rangle$ induces a transferable utility coalitional game (I, v_Γ). We first discuss a simple unrestricted form of TGs, then consider various restrictions and their impact on the game.

Definition 1 Unrestricted-TG: An unrestricted TG (UTG) Γ with the goal resource r_g induces a simple transferable utility coalitional game where a coalition C wins if it can derive r_g and loses otherwise:

$$v_\Gamma(C) = \begin{cases} 1 & \text{if } C \Rightarrow r_g \\ 0 & \text{otherwise} \end{cases}$$

The game Γ will usually be clear from context, and so we usually drop reference to it and write v rather than v_Γ.

We can modify this model slightly to take into account the total number of transformations used to generate resources, and the time required to generate a resource. We denote the fact that a coalition C can generate a resource r using at most k transformations by $C \Rightarrow_k r$. Consider a sequence of resource subsets $S = \langle Q_1, Q_2, \ldots Q_k \rangle$, such that each Q_i contains one additional resource over the previous Q_{i-1} (so $Q_i = Q_{i-1} \cup \{r'_i\}$). We say C *allows* the sequence S if for any index i, C can generate r'_i (the additional item for the next resource subset in the sequence) given base resources in Q_{i-1} (so C is capable of a transformation $d = \langle A, r'_i \rangle$, where $A \subseteq Q_{i-1}$). A sequence $S = \langle Q_1, Q_2, \ldots Q_k \rangle$ (with k subsets) that C allows is called a $k-1$-*transformation sequence for resource r by coalition C* if $r \in Q_k$ and the first subset in the sequence is the subset of resources the coalition C is endowed with, $Q_1 = R_C$ (since C requires $k-1$ transformations to obtain r this way). If there exists such a sequence, we denote this by $C \Rightarrow_k r$. We denote the minimal number of transformations that C needs to derive r as $d(C, r) = \min\{b \mid C \Rightarrow_b r\}$, and if C cannot derive r we denote $d(C, r) = \infty$.

Definition 2 DTG: A transformation restricted TG (DTG[2]) with the goal resource r_g and with the transformation bound k is the game where a coalition C wins if it can derive r_g *using at most k transformations* and loses otherwise:

$$v(C) = \begin{cases} 1 & \text{if } C \Rightarrow r_g \text{ and } d(C, r_g) \leq k \\ 0 & \text{otherwise.} \end{cases}$$

Similarly, we consider the makespan domain, where each transformation requires a certain amount of time. The main difference between the makespan domain and the DTG domain is that transformations may be done *simultaneously*.[3] We denote the fact that a coalition C can generate a resource r in time of at most t by $C \Rightarrow^t r$. We define the notion recursively. If a coalition is endowed with a resource, it can generate this resource instantaneously (with time limit of 0), i.e., if $has(C, r)$ then $C \Rightarrow^0 r$. Now consider a coalition C such that $C \Rightarrow^{t_1} r_{b_1}, C \Rightarrow^{t_2} r_{b_2}, \ldots, C \Rightarrow^{t_m} r_{b_m}$, and player $a_i \in C$ who is capable of the transformation $d = \langle \{r_{b_1}, r_{b_2}, \ldots, r_{b_m}\}, r \rangle$ (so $d \in D_i$), requiring a transformation time t, so $t_i(d) = t$. Given a coalition C, we denote the time in which a coalition can perform a transformation as $t_C(d) = \min_{a_i \in C} t_i(d)$, the minimal time in which the transformation can be performed, across all players in the coalition. We denote the time in which the coalition can obtain *all* of the

[2]DTG stands for Delimitated Transformation Game.

[3]E.g., if it takes 5 hours to convert oil to gasoline and 4 hours to convert oil to plastic, if we have oil we can obtain both gasoline and plastic in 5 hours, using parallel transformations.

base resources $r_{b_1}, \ldots, r_{b_m}$ as $s = \max t_i$. The final transformation (which generates r) requires a time of t, so $C \Rightarrow^{s+t} r$.

Different ways of obtaining the target resource result in different time bounds, and we consider the optimal way of obtaining the target resource (the *minimal* time a coalition C requires to derive r). We denote the minimal transformation time that C needs to derive r by $t(C, r)$, where this value is defined as follows:

$$t(C, r) = \begin{cases} \min\{b \mid C \Rightarrow^b r\} & \text{if } C \Rightarrow r \\ \infty & \text{otherwise.} \end{cases}$$

Similarly to DTGs, we define makespan (time limited) TGs:

Definition 3 TTG: A time limited TG (TTG) with goal resource r_g and time limit t is the game where a coalition C wins if it can derive r_g *with time of at most t* and loses otherwise:

$$v(C) = \begin{cases} 1 & \text{if } C \Rightarrow r_g \text{ and } t(C, r_g) \leq t \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to see that all the above versions of TGs are increasing (monotone), so if $C' \subseteq C$ then $v(C') \leq v(C)$: adding more players makes more possible transformations available to the coalition, so increasing the coalition keeps all the previously possible transformation paths and potentially allows for more efficient derivations.

3.3 Solving transformation games to analyse supply chains and proof systems

Earlier, we introduced two key motivating scenarios for TGs: supply chains and proof systems. We now discuss how the TG model can be used as a framework through which to analyse these frameworks.

3.3.1 Supply chains

TGs provide an abstract model of supply chains. They capture many important features of supply chains, such as the structure of the chains, possible transformations, costs, and deadlines. On the other hand, they do not capture some elements of significance in real-world supply chains, such as required quantities of resources or the elimination of base resources in certain transformations. The TG model is thus sufficiently rich to describe some industries, such as some sectors in the pharmaceutical and electronics industries, where an important factor in the supply chain is the expertise in transforming base resources to products, rather than quantities of resources. As we later demonstrate, while it is easy to devise a richer model that captures many more properties of supply chains, making the model richer is likely to make it computationally intractable to answer many important questions regarding the supply chain.

We now discuss key questions that arise when analysing supply chains and their relation to various game theoretic notions. In Section 1.1.1 we considered a simple supply chain domain, and important questions regarding such chains. First, there may be several possible chains that generate the ultimate product sold to consumers. Which of these chains are more likely to be formed? Second, how would the firms share the revenue obtained from selling the product to consumers? Which contracts are likely to form among the firms? Finally, which firm is most important in generating the output resource? How can we quantify its importance and bargaining power with regard to other firms in the market?

Our first question regarding a supply chain domain was which chains are likely to form. More formally, we may wish to determine which coalitions can generate the target product with the minimal cost, time, or number of transformations. We may also try and find the set of coalitions that can generate the target product with a cost or time below a certain threshold (or determine if any such coalitions exist). These problems deal with finding the *optimal* coalitions.

The remaining questions relate the agreements between firms across the supply chain regarding sharing the income from selling the target product. Consider a supply chain where the target product could be sold for its market price, providing the generating coalition a certain monetary reward. This value generated from the target product could be distributed among the participants of the forming chain in many ways. We might want to predict how this value would be *distributed* or *shared* among the members of the chain. This could be formulated as *solving* the game. Several game theoretic solution concepts can be used, depending on the specific question asked regarding the distribution of payoffs.

Power indices, such as the Banzhaf index and the Shapley value, make it possible to identify the most *critical* parts of the chains—those that have the biggest impact on the revenue generated. Alternatively, they could be considered as *fair* ways of allocating the revenue among the participants of the chain, in the sense that they distribute greater value to the more critical parts of the chain. The core can be used to determine which coalitions (chains) are stable and are likely to be long-lasting. Some agreements regarding sharing the utility are not likely to last long, as a subset of the firms in the chain could form an alternative agreement that is more profitable for them. By computing a game theoretic solution to the TG, we can identify reasonable agreements among firms in the supply chain.

As discussed above, some key questions regarding supply chain domains can be formulated as computing optimal coalitions in TGs,[4] or as solving the TG using various game theoretic concepts, such as the core, the Banzhaf index, or the Shapley value. Therefore, characterizing the computational complexity of solving these problems is an important research question.

3.3.2 Proof systems

Structurally, TGs are similar to logical proof systems (see, e.g., [44, p.48]). In a proof system in formal logic, we have a set of formulae of some logic, known as the *premises*, and a collection of *inference rules*, the role of which is to allow us to derive new formulae from existing formulae. Formally, if $\mathbb{L}$ is the set of formulae of the logic, then an inference rule ρ can be understood as a relation $\rho \subseteq 2^{\mathbb{L}} \times \mathbb{L}$. Given a set of premises $\Delta \subseteq \mathbb{L}$ and a set of inference rules $\rho_1 \ldots, \rho_k$, a *proof* is a finite sequence of formulae $\phi_1, \ldots, \phi_l$, such that for all i, $1 \leq i \leq l$, either:

1. $\phi_i \in \Delta$ (i.e., ϕ_i is a premise); or else

[4]The wording *optimal coalition* relates to several possible definitions: coalitions that generate the target product while minimizing costs; coalitions minimizing the time to generate the product; coalitions that can construct the target product using as few transformations as possible.

2. there exists some subset $\Delta' \subseteq \{\phi_1, \ldots, \phi_{i-1}\}$ and some $\rho_j \in \{\rho_1, \ldots, \rho_k\}$ such that $(\Delta', \phi_i) \in \rho_j$ (i.e., ϕ_i can be derived from the formulae preceding ϕ_i by some inference rule).

Typical notation is that $\Delta \vdash_{\rho_1,\ldots,\rho_k} \phi$ means that ϕ can be derived from premises Δ using rules $\rho_1, \ldots, \rho_k$. Such proofs can be modelled in our framework as follows. Resources R are logical formulae $\mathbb{L}$, and the initial allocation of resources $R_1, \ldots, R_n$ equates to the premises; capabilities $D_1, \ldots, D_n$ equate to inference rules.

Notice that the assumption that resources are not "consumed" during the transformation process is very natural when considered in this setting: in classical logic proofs, premises and lemmas can be reused as often as required (although this is not the case in "resource aware" logics such as linear logic [50]). Clearly the relationship between TGs and proofs is very natural: such formal proof systems can be directly modelled within our framework. There are two main differences, however, as follows.

First, in proof systems inference rules are usually given a succinct specification, as a "pattern" to be matched against premises. The classical proof rule modus ponens, for example, is usually specified as the following pattern:

$$\frac{\phi; \quad \phi \to \psi}{\psi}$$

This pattern says that if we have derived ϕ, and we have derived that $\phi \to \psi$, then we can derive ψ. Here, ϕ and ψ are meta-logical variables, which can be instantiated with any formula.

Second, we take a *strategic* view: a proof modelled within our system is obtained through a cooperative process. TGs can be understood as a formulation both of cooperative theorem proving systems [32, 41], as well as cooperative problem solving systems in general [53]. In such systems, agents have different areas of expertise (= resources) as well as different capabilities (= transformations). Game theoretic concepts such as the Banzhaf index provide a measure of how important different premises and inference rules are with respect to being able to prove a theorem.

In Section 1.1.2 we considered a set I of experts that could work together to prove a complex theorem ϕ. We can denote the set of base theorems that expert i can prove as T_i. Although for any i the theorems in T_i are correct, only expert i knows how to prove them, so she can easily reveal the theorems she knows, as this information is not useful without the proof itself. The key question in the proof system was identifying the most influential experts—those who contributed most to the proof of the final theorem. When it is impossible to prove the target theorem unless all the experts are present, one could argue that all of them are equally important (each of them is critical for the grand coalition). Also, when any of the experts can prove the target theorem on its own, they are all equally important. However in many such systems there would be many subsets of experts that could devise a proof for the target theorem, which requires a more methodical way of assessing their importance. Formally, consider a possible proof P for the target theorem, which only uses the base theorems $S \subseteq \bigcup_{i \in I} T_i$. Consider the expert coalition C. If the base theorems that P uses, S_p, is one such that $S_p \subseteq \bigcup_{i \in C} T_i$ then C can prove ϕ. A coalition C can prove that target theorem if there *exists* a proof P such that S_p, the base theorems that P uses, is such that $S_p \subseteq \bigcup_{i \in C} T_i$. Thus, the proof system domain partitions the set of possible expert *coalitions* into two—those that can prove the target theorem, and those that cannot.

Such a proof system domain can be expressed as a TG, where the "resources" are facts and theorems, and transformations enable the derivation of further theorems using simpler theorems.[5] In such a system, a proof that requires fewer transformations could be considered a simpler proof, which is desirable. The TG represents the proof system domain in a way that allows capturing the notion of an expert's relative importance or influence through game theoretic tools.

Consider the case where we wish to give each agent an influence measure, such that the sum of these measures is a single unit. There are many possible desirable properties for such a measure of influence. For example, if an expert is not critical for any expert set C,[6] we would want its measure of influence to be zero. Also, if two experts i and j are interchangeable, so for any expert subset C such that $i \notin C$ and $j \notin C$ the coalition $C \cup \{i\}$ can prove the target theorem iff $C \cup \{i\}$ can prove it, we would want i and j to have an identical measure of influence. Such desiderata are called fairness axioms.

As discussed in Section 2, game theoretic solutions such as the Shapley value and Banzhaf index fulfill different such sets of fairness axioms. For example, both these solutions give an expert that is not critical for any set a share of zero, and give interchangeable experts equal shares. A more complete discussion of the axioms that the Banzhaf index and Shapley value fulfill is given in [1, 59]. Thus these indices can be used to determine the relative importance of experts in proof system domains, again highlighting the importance of examining the complexity of computing them.

4 Problems and algorithms

Section 3.3 discussed the potential uses of the TG model for analysing supply chains and proof systems. Each of these domains raises important natural problems regarding TGs. We now formally define these problems, based on the definitions of TGs in Section 3. We consider a TG $\Gamma = \langle I, R, R_1, \ldots, R_n, D_1, \ldots, D_n, r_g \rangle$. We assume the game is given in a concise form, consisting of a list of the agents I, a list of the resources R, a list R_i for every agent i consisting of all the resources an agent is endowed with, a list of transformations D_i for every agent i (each with the input resources and output given as a list) and the identity of the goal resource r_g.[7] We examine the following problems regarding the game.

[5]We can consider standard inference rules to be transformations, so the agents would all have the same set of transformations but different base resources, or we could consider more sophisticated transformations.

[6]Following the definitions of Section 2, we say an expert i is critical for a subset C of agents (such that $i \in C$) if C can prove the target theorem, but $C \setminus \{i\}$ cannot.

[7]Our input size is thus polynomial in the number of agents, resources and transformations. An alternative representation is a full characteristic function, mapping every agent subset to their value in the TG. This alternative representation is of course exponential in the number of agents, so it is not tractable. Similarly to pseudo-polynomial algorithms which operate on inputs given in unary rather than in a binary representation, algorithms on the naive representation are likely to be polynomial in the input size, as this input size is so big. We are interested in algorithms that are polynomial in the size of the input game when it is given in the concise form—such algorithms must be polynomial in the number of agents, resources and transformations, making them usable in practice.

Table 1 Complexity of TG problems

	UTG	DTG	TTG
CV	P	P (NPC)	P
VETO	P	P (co-NPC)	P
DUMMY	co-NPC	co-NPC (co-NPH)	co-NPC
CORE	P	P (co-NPC)	P
SHAPLEY	#P-Hard	#P-Hard	#P-Hard
BANZHAF	#P-Hard	#P-Hard	#P-Hard

If the results differ for simple and complex transformations, the results for complex transformations are given in parentheses. Key: P = polynomial algorithm; NPH = NP-hard; NPC = NP-complete co-NPC = co-NP-complete; co-NPH = co-NP-hard

Definition 4 COALITION-VALUE (CV): Given a coalition $C \subseteq I$, compute $v_\Gamma(C)$ (i.e., test whether a coalition is successful or not).

Definition 5 VETO (VET): Given a player a_i, check if it is a veto player, so for any winning coalition C, we have $a_i \in C$.

Definition 6 DUMMY (DUM): Given a player a_i, check if it is a dummy player, so for any coalition C, we have $v_\Gamma(C \cup \{a_i\}) = v_\Gamma(C)$.

Definition 7 CORE: Compute the set of payoff vectors that are in the core, and return a representation of all payoff vectors in it.

Definition 8 SHAPLEY (SH): Compute a_i's Shapley value $sh_i(v_\Gamma)$.

Definition 9 BANZHAF (BZ): Compute a_i's Banzhaf power index $\beta_i(v_\Gamma)$.

We now summarise the results of the present paper, and prove them in the remainder of the paper:

- We provide polynomial algorithms for testing whether a coalition wins or loses (CV) for UTGs, DTGs, and TTGs with simple transformations, and for UTGs and TTGs with complex transformations, but show that testing whether a coalition wins is NP-complete for DTGs with complex transformations.
- We provide polynomial algorithms for testing for veto players and computing the core in all domains where CV is computable in polynomial time, but show the problem is co-NP-hard in DTGs with complex transformations.
- We show that testing for dummy players and computing the Shapley value are co-NP-hard in all the TG domains defined, and provide a stronger result for the Banzhaf power index, showing that it is #P-hard in all these domains.[8]

Table 1 summarises our results relating to TGs with *simple* transformations.

We now present proofs for these results.

[8]The complexity class #P expresses the hardness of problems that "count solutions". Informally, NP deals with whether a solution to a combinatorial problem exists, while #P deals with calculating the *number* of solutions. Counting solutions generalizes the checking of their existence, so we usually regard #P-hardness as a more negative result than NP-hardness.

Theorem 1 *CV is in P, for all the following types of TGs with simple transformations: UTG, DTG, TTG. CV is in P for UTGs and TTGs with complex transformations.*

Proof First consider UTG. Denote the set S of resources with which C is endowed, $S = \{r \mid has(C, r)\}$. Denote the set of transformations of the players in C as $D_C = \cup_{a_i \in C} D_i$. We say that a set of resources S matches a transformation $d = \langle B, r\rangle \in D$ if $B \subseteq S$. If S matches d then using the resources in S the coalition C can also produce r through transformation d. Consider a basic step of iterating through all transformations in D_C. When we find a transformation $d = \langle B, r\rangle$ that S matches, we add r to S. A test to see whether a transformation d matches S can be done in time at most $|R|^2$ (where R is the set of all resources), so the basic step takes at most $|D_C| \cdot |R|^2$ time. If after performing a basic step no transformation in D_C matches S, S holds all the resources that C can generate, and we stop performing basic steps. If S has changed during a basic step, at least one resource is added to it. Thus, we perform at most $|R|$ basic steps to compute the set of all resources C can generate, so S can be computed in polynomial time. We can then check whether S contains r_g. We note that the suggested algorithm works for simple as well as complex transformations. Now consider TTGs with simple transformations. We build a directed graph representing the transformations as follows. For each resource r the graph has a vertex v_r, and for each transformation $d = \langle r_x, r_y\rangle$ the graph has an edge e_d from v_{r_x} to v_{r_y}. Given a coalition C we consider G_C, the subgraph induced by C. $G_C = \langle V, E_C\rangle$ contains only the edges of the transformations available to C, so $E_C = \{\langle v_{r_x}, v_{r_y}\rangle \mid \langle r_x, r_y\rangle \in D_C\}$. The graph G_C is weighted, and the weight of each edge $e = \langle r_x, r_y\rangle$ is $w(e) = \min_{a_i \in C} t_i(\langle r_x, r_y\rangle)$, the minimal time to derive r_y from r_x across all players in the coalition. Denote the weight of the minimal path from r_a to r_g in G_C as $w_C(r_a, r_g)$. The coalition C is endowed with all the resources in R_C and can generate all of them instantly. The minimal time in which C can generate r_g is $\min_{r_a \in R_C} w_C(r_a, r_g)$. For each resource $r_a \in R_C$, we can compute $w_C(r_a, r_g)$ in polynomial time, so we can compute in polynomial time the minimal time in which C can generate r_g, and test whether this time exceeds the required deadline. For simple transformations, we can simulate a DTG domain as a TTG domain, by having each transformation require 1 time unit (and setting the threshold to be the threshold number of transformations[9]).

Finally, we show how to adapt the algorithm used for UTGs (with either simple or complex transformations) to be used for TTGs with complex transformations. For the TTG CV algorithm for a coalition C, for each resource r we maintain $m(r)$, a bound from above on the minimal time required to produce r. All the $m(r)$ of resources endowed by some player in the coalition C are initialized to 0, and the rest are initialized to ∞. Our basic step remains iterating through all the transformations in D. When we find a transformation $d = \langle B, r\rangle$ which S matches, where the transformation requires $t(d)$, we compute the time in which the transformation can be completed, $c(d) = max_{b \in B} m(b) + t_C(d)$ (if S does not match a transformation d, we denote $c(d) = \infty$). During each basic step, we compute the possible completion times for all the matching transformations, and apply the smallest one,

[9]With complex transformations, this is no longer possible, since if a transformation requires several base resources, the shortest time to produce each of them may be different.

$argmin_{d \in D} c(d)$. To apply a transformation $d = \langle B, r \rangle$, we simply add r to S, and update $m(r)$ to be $c(d)$. During each basic step we only apply *one* transformation (although we scan all the possible transformations). A simple induction shows that after each basic step, for any resource r such that $m(r) \neq \infty$ the value $m(r)$ is indeed the minimal time required to generate r. Again, the algorithm ends if no transformations were applied during a basic step. As before, a basic step requires time of $|D_C| \cdot |R|^2$ time, and we perform at most $|R|$ basic steps, so the algorithm requires polynomial time. We can then check whether S contains r_g, and whether $m(r_g)$ is smaller than the required time threshold. □

Corollary 1 *VETO is in P, for all the following types of TGs with simple transformations: UTG, DTG, TTG, and for UTGs and TTGs with complex transformations.*

Proof A veto player a_i is present in all winning coalitions: TGs are increasing, so simply check whether $v(I_{-a_i}) = 0$. □

Now consider the problem of computing the core in TGs. The core focuses on stability of the payoff allocations. Under a core imputation, no subset of agents is incentivised to defect and form their own team. When the core is non-empty, it contains all the stable imputations; when it is empty, the coalition would be unstable no matter how we divide the utility among the agents. We first note that it is not always possible to represent the core in a succinct way, since it may contain an infinite number of imputations. However, in the case of TGs without costs, there does exist a succinct representation for the core.

All forms of TGs without costs (in Definitions 1–3) are *simple cooperative games*. In a simple cooperative game, the value of a coalition is either 1 or 0 (i.e., a coalition either wins or loses). Our TGs are also *monotone games*—adding players to a winning coalition never makes it lose, as we only allow the use of more resources and transformations. The core is a very demanding concept in simple cooperative games. A veto player is a player that is present in all winning coalitions, so if $a_i \notin C$ we have $v(C) = 0$. It is a well-known fact that in monotone simple coalitional games, the core is non-empty if and only if there is at least one veto player in the game, and that if the core is non-empty it allocates all the reward only to the veto players [25] (further, if there are veto players, any imputation that does not allocate any reward to the non-veto players is in the core). Consider a simple coalitional game that has no veto players, so for every agent a_i we have a winning coalition C that does not contain a_i. Take an imputation $p = (p_1, \ldots, p_n)$ where $p_i > 0$. Since $\sum_{i=0}^{n} p_i = 1$ and since $p_i > 0$ we know that $p(C) \leq \sum_{p_j \in I_{-a_i}} p_j < 1$, so $p(C) < v(C) = 1$. This makes C a blocking coalition. On the other hand, examine any imputation p where non-veto players get nothing. Such an imputation must be in the core: any coalition C that can potentially block p must have $v(C) = 1$ (if $v(C) = 0$ then it cannot block), and must contain all the veto players, so $\sum_{p_j \in C} p_j = 1$, and thus cannot block p.

Due to the above characterization of the core in simple cooperative games, in such games the core can be represented as a set I_{veto}, consisting of all the veto players in that game. This set represents all core imputations: an imputation $p = (p_1, \ldots, p_n)$ is in the core if $\sum_{i \in I_{veto}} p_i = 1$ (note that it must be the case that $\sum_{i \in I} p_i = 1$ for p to be an imputation). Thus, given this representation we can find a core imputation in polynomial time (if such an imputation exists), for example by equally sharing the utility among the veto players. Given this representation we can also answer core membership queries (i.e., testing if a given imputation p is in the core), by making sure the utility is distributed only to the veto players. From Corollary 1 we know we can test all the agents and find the set of veto agents I_{veto} in polynomial time, allowing us to return the veto agent representation of the core of the game. This gives the following:

Corollary 2 *CORE is in P, for all the following types of TGs with simple transformations: UTG, DTG, TTG, and for UTGs, DTGs and TTGs with complex transformations.*

Computing dummy players, however, is more complex.

Theorem 2 *DUMMY is co-NP-complete, for all the following types of TGs with simple transformations: UTGs, DTGs, TTGs, and for UTGs and TTGs with complex transformations. For DTGs with complex transformations, DUMMY is co-NP-hard.*[10]

Proof Due to Theorem 1, we can verify in polynomial time whether a_i is beneficial to C by testing if $v(C \cup \{a_i\}) - v(C) > 0$. Thus DUMMY is in co-NP for UTGs, DTGs, and TTGs with simple transformations, and for UTGs and TTGs with complex transformations. We reduce SAT to testing if a player in a UTG with simple transformations is not a dummy (TG-NON-DUMMY). Showing that DUMMY is co-NP-hard in UTGs is enough to show that it is co-NP-hard for DTGs and TTGs, since it is possible to set the threshold (of the maximal allowed transformations or allowed time) so high that the TG is effectively unrestricted. Hardness results also apply to complex transformations as well, since the restricted case of simple transformations is hard.

Assume w.l.o.g. that the SAT instance is in CNF, and let this instance be $\phi = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ over propositions $x_1, \ldots, x_n$, where $c_i = l_{i_1} \vee \cdots \vee l_{i_k}$, where each such l_j is a positive or negative literal, either x_k or $\neg x_k$ for some proposition x_k. The TG-NON-DUMMY query is regarding the player a_y. For each literal (either x_i or $\neg x_i$) we construct a player (a_{x_i} and $a_{\neg x_i}$). These players are called the literal players. The generated TG game has a resource r_y, and only a_y is endowed with that resource. The game also has the resource r_z, with which all the literal players are endowed. For each proposition x_i we also have a resource r_{x_i}. For each clause c_j in the formula ϕ we have a resource r_{c_j}. The goal resource is the resource r_g.

For each positive literal x_i we have transformation $d_{x_i} = \langle r_z, r_{x_i} \rangle$. For each negative literal we have transformation $d_{\neg x_i} = \langle r_{x_i}, r_g \rangle$. For each clause c_j we have transformation $d_{c_j} = \langle r_{c_j}, r_{c_{j+1}} \rangle$, where for the last clause c_m we have a transformation

[10]We could not prove that DUMMY is even in co-NP for DTGs with complex transformations. To show membership in coNP we must show the existence of a proof that an agent a_i is a non-dummy which is verifiable in polynomial time. Such a proof can include a coalition C such that $C \cup \{a_i\}$ wins but C loses, and a sequence of transformations through which $C \cup \{a_i\}$ generates the target resource. However, this does not suffice, as we must also be able to verify that C loses—we could not find a way to verify this in polynomial time.

$d_{c_m} = \langle r_{c_m}, r_g \rangle$. Player a_y is only capable of $d_0 = \langle r_y, r_{c_1} \rangle$. Player a_{x_i} is capable of d_{x_i}, and player $a_{\neg x_i}$ is capable of $d_{\neg x_i}$. If x_i occurs in its positive form in c_j (i.e., $c_j = x_i \vee l_{i_2} \vee \cdots$) then a_{x_i} is capable of d_{c_j}. If x_i occurs in its negative form in c_j (i.e., $c_j = \neg x_i \vee l_{i_2} \vee \cdots$) then $a_{\neg x_i}$ is capable of the d_{c_j}.

We identify an assignment with a coalition, and identify a coalition with an assignment candidate (which possibly contains both a positive and a negative assignment to a variable, or which possibly does not assign anything to a variable). Let A be an assignment to the variables in ϕ. We denote the coalition that A represents as $C_A = \{a_{x_i} \mid A(x_i) = T\} \cup \{a_{\neg x_i} \mid A(x_i) = F\}$. There are only two resources with which players are endowed: r_y and r_z. It is possible to generate r_g either through a transformation chain starting with r_z, going through r_{x_i} (for some variable x_i) and ending with r_g, or through a transformation chain starting with r_y, going through r_{c_1}, through r_{c_2}, and so on, until r_{c_m}, and finally deriving r_g from r_{c_m} (no other chains generate r_g).

Given a valid assignment A, C_A does not allow the conversion of r_z to r_g, since to do so C_A needs to be able to generate r_{x_i} from r_z (for some variable x_i) and needs to be able to generate r_g from r_{x_i}. However, the only player who can generate r_{x_i} from r_z is a_{x_i}, and the only player who can generate r_g from r_{x_i} is $a_{\neg x_i}$, and C_A can never contain both a_{x_i} and $a_{\neg x_i}$ (for any x_i) by definition of C_A.

Suppose A is a satisfying assignment for ϕ. Let c_j be some clause in ϕ. A satisfies ϕ, so it satisfies c_j through at least one variable x_i. If x_i occurs positively in ϕ, $A(x_i) = T$ so $a_{x_i} \in C_A$, and if x_i occurs negatively in ϕ, $A(x_i) = F$ so $a_{\neg x_i} \in C_A$, so we have a player $a \in C$ capable of d_{c_j}. Thus, C_A can convert r_{c_1} to r_{c_2}, can convert r_{c_2} to r_{c_3}, and so on. Thus, given r_{c_1}, C_A can generate r_g. Player a_y is endowed with r_y, and can generate r_{c_1} from r_y, so $C_A \cup \{a_y\}$ wins. However, $a_y \notin C_A$, and C_A cannot generate r_{c_1}. Since A is a valid assignment, C_A cannot generate r_g through a chain starting with r_z, so C_A is a losing coalition. Thus, a_y is not a dummy, as $v(C_A \cup \{a_y\}) - v(C_A) = 1$.

On the other hand, suppose a_y is not a dummy, and is beneficial to coalition C, so C is losing but $C \cup \{a_y\}$ is winning. Since C loses and cannot contain both a_{x_i} and $a_{\neg x_i}$ (for any x_i), as this would allow it to generate r_{x_i} from r_z and to generate r_g from r_{x_i} (and C would win without a_y). Consider the assignment A: if C contains a_{x_i} we set $A(x_i) = T$, and if C contains $a_{\neg x_i}$ we set $A(x_i) = F$ (if C contains neither a_{x_i} nor $a_{\neg x_i}$ we can set $A(x_i) = T$). Since $C \cup \{a_y\}$ wins, but cannot generate r_g through a chain starting with r_z, it must generate r_g through the chain starting with r_y and going through the r_{c_j}'s. Thus, for any clause c_j, C contains a player capable of transformation $d_{c_j} = \langle r_{c_j}, r_{c_{j+1}} \rangle$. That player can only be a_{x_i} or $a_{\neg x_i}$ for some proposition x_i. If that player is $a_{x_i} \in C$ then c_j has the literal x_j (in positive form) and $A(x_i) = T$, so A satisfies c_j, and if it is $a_{\neg x_i} \in C$ then c_j has the literal $\neg x_j$ (negative form) and $A(x_i) = F$, so again A satisfies c_j. Thus A satisfies all the clauses in ϕ. □

Theorem 3 *For DTGs with complex transformations, determining if a coalition C wins (i.e., the CV problem) is NP-complete. The problem is hard even in TGs with a single player. For DTGs with complex transformations, VETO and CORE are co-NP-complete.*

Proof We first show the problems are in NP / coNP. A proof that a coalition C wins consists of a valid sequence of transformations to generate the target resource, such that the base resources for every transformation are either resources the members of C are endowed with, or the result products of previous transformations. We can verify if a transformation sequence is valid in polynomial time by keeping track of the resources generated by the transformations. TGs are monotone, and adding more agents to a winning coalition never makes it lose. Thus if a_i is a non-veto player, the coalition $I \setminus \{a_i\}$ must be winning. Again, a proof for this can be a valid transformation sequence for generating the target resource, which can be verified in polynomial time. Thus, for DTGs with complex transformations, the problem of testing if a coalition C wins is in NP, and VETO is in co-NP. CORE is also in co-NP in DTGs with complex transformations. To show that an imputation is *not* in the core, we have to find a non-veto agent which receives a non-zero reward. This can be demonstrated by a polynomially verifiable proof that a specific agent is a non-veto agent (as it is easy to check in constant time whether that agent receives a non-zero reward in a given imputation), as discussed above.

To show NP-hardness, we reduce VERTEX COVER to DTG CV. In the VERTEX COVER problem, we are given a graph $G = \langle V, E \rangle$ and an integer K, and are asked if we can select a vertex subset $X \subseteq V$ of size k such that each edge $e \in E$ is connected to some vertex $u \in X$ (i.e., if the edge is $e = (x, y)$ then either $x \in X$ or $y \in X$). VERTEX COVER is a prominent NP-hard problem [43].

Examine the VERTEX COVER instance on the graph $G = \langle V, E \rangle$ with vertices $V = \{v_1, \dots, v_n\}$ and edges $E = \{e_1, \dots, e_m\}$ where e_i is from $v_{i,a}$ to $v_{i,b}$ and a target cover size of k. We construct the following DTG. We have a resource r_t and goal resource r_g, a resource r_{e_i} for each edge e_i, and a resource r_{v_i} for each vertex. We have a transformation from r_t to each vertex resource r_{v_i}. If e_i is from $v_{i,a}$ to $v_{i,b}$ we have two transformations: from $r_{v_{i,a}}$ to r_{e_i}, and from $r_{v_{i,b}}$ to r_{e_i}. We have a complex transformation from $\{r_{e_1}, \dots, r_{e_m}\}$ to r_g. A single player has r_t and all the above transformations. The target maximal number of transformations for the DTG is $k + m + 1$. Now, $G = \langle V, E \rangle$ has a vertex cover of size k iff the player wins in the game so defined. □

Corollary 3 *Testing whether the Shapley value or Banzhaf index of a player in a TG exceeds a certain threshold is co-NP-hard for all the following types of TGs: UTG, DTG, and TTG, with simple or complex transformations.*

Proof Theorem 2 shows DUMMY is co-NP-hard in these domains. However, the Shapley value or Banzhaf index of a player can only be 0 if the player is a dummy player. Thus, computing these indices in these domains (or the decision problem of testing whether they are greater than some value) is co-NP-hard. □

We now show a stronger result for the Banzhaf index, using a reduction from #SET-COVER (#SC).

Definition 10 #SET-COVER (#SC): We are given a collection $C = \{S_1, \dots, S_n\}$ of subsets. We denote $\cup_{S_i \in C} S_i = S$. A set cover is a subset $C' \subseteq C$ such that $\cup_{S_i \in C'} = S$. We are asked to compute the number of covers of S.

The problem #VERTEX-COVER, where we are asked to count to the number of different vertex covers in a graph, is a restricted form of #SC. Vadhan [74] showed that #VERTEX-COVER is #P-hard,[11] so #SC is of course also #P-hard. To prove that CG-BANZHAF is #P-complete, we show a reduction from #SC to CG-BANZHAF.[12]

Theorem 4 *Computing the Banzhaf index in UTGs, DTGs, and TTGs (with simple or complex transformations) is #P-hard.*

Proof We reduce a #SC instance to checking the Banzhaf index in a UTG. Consider the #SC instance with $C = \{S_1, \ldots, S_n\}$, so that $\cup_{S_i \in C} S_i = S$. Denote the items in S as $S = \{t_1, t_2, \ldots, t_q\}$. Denote the items in S_i as $S_i = \{t_{(S_i,1)}, t_{(S_i,2)}, \ldots, t_{(S_i,k_i)}\}$.

For each subset S_i of the #SC instance, the reduced UTG has a player a_{S_i}. For each item $t_i \in S$ the UTG instance has a resource r_{t_i}. The reduced instance also has a player a_{pow}, the resources r_0, r_{pow} and the goal resource r_g. For each item $t_i \in S$ there is a transformation $d_i = \langle\{r_{t_{i-1}}\}, r_{t_i}\rangle$. Another transformation is $d_{pow} = \langle\{r_{t_q}\}, r_g\rangle$, of which only a_{pow} is capable. All players have resource r_0.

Each player is capable of the transformation in her subset—for the subset $S_i = \{t_{i_1}, t_{i_2}, \ldots, t_{i_k}\}$, the player a_i is capable of $d_{i_1}, d_{i_2}, \ldots, d_{i_k}$. The query regarding the power index is for player a_{pow}.

Note that a coalition C wins iff it contains both a_{pow} and players who are capable of all $d_1, d_2, \ldots, d_q$. However, to be capable of d_i the coalition must contain some a_j such that $t_i \in S_j$.

Consider a winning coalition $C = \{a_{pow}\} \cup \{a_{j_1}, a_{j_2}, \ldots, a_{j_k}\}$, and denote $S_C = \{S_{j_1}, S_{j_2}, \ldots, S_{j_k}\}$. A coalition C wins iff $a_{pow} \in C$ and S_C is a set cover of S. The Banzhaf index in the reduced game is $\frac{q}{2^{n-1}}$, where n is the number of players and h is the number of winning coalitions that contain a_{pow} that lose when a_{pow} is removed from the coalition. No coalition can win without a_{pow}, so h is the number of all winning coalitions, which is the number of set covers of the #SC instance.

Thus we reduced #SC to BANZHAF in a UTG with simple transformations (a restricted case of complex transformations). We can do the same with DTGs and TTGs with a high-enough threshold. Thus, BANZHAF is #P-hard in all considered TG domains. □

A recent paper [3] shows that for any *reasonable* representation of a cooperative game, if computing the Banzhaf index is #P-hard, then computing the Shapley value is also #P-hard. A representation language is said to be reasonable if it is possible to represent the game with an additional dummy agent using the same language.[13] Our TG representation is reasonable—to add an additional dummy agent we simply add an agent who is capable of no transformations and who is endowed with no resources. Using the result of [3], we obtain the following corollary.

[11] He also showed that the problem remains #P-hard even in very restricted classes of graphs.

[12] The papers [18, 20] consider a related domain (Coalitional Skill Games and Connectivity Games), and also use #SC to show that computing the Banzhaf index in that domain is #P-complete.

[13] To be more precise, a language is reasonable if for any game v that it can represent, it can also represent the game v' defined as follows. The game v' has an additional agent x that is not present in the original game v. For any coalition C such that $x \notin C$ we have $v'(C) = v(C)$. For any coalition C such that $x \in C$ we have $v'(C) = v(C \setminus \{x\})$.

Corollary 4 *Computing the Shapley value in UTGs, DTGs, and TTGs (with simple or complex transformations) is #P-hard.*

4.1 Discussion of the complexity results

The complexity results summarised in Table 1 reveal an interesting picture regarding the computational complexity of solving TGs. First, in UTGs, TTGs and DTGs (with simple transformations), computing the value of a coalition, testing for veto agents and computing the core can be done in polynomial time. Therefore, in such games it is possible to find *stable* divisions of the rewards, such that no subset of agents is incentivised to defect and form an alternative coalition. Our positive results are significant for the supply chain domain, as such distributions of the profits would preserve the structure of the supply chain intact, with no subset of the participants deviating to form an alternative chain.

The picture is a bit less encouraging when it comes to DTGs with complex transformations. In such games calculating any solution concept we examined or testing any of the agent properties we examine is computationally hard. In fact, for such games it is even NP-hard to compute the value of a coalition. Our negative result highlight the fact that although it is appealing to apply game theoretic analysis to predict agent agreements in this domain, the combinatorial complexity of the domain makes it difficult to apply such solutions in practice. Of course, NP-hardness is a "worst-case" notion, so it is quite possible that some games could be efficiently solved in practice: NP-hardness means that there can be no *guarantee* of obtaining a solution efficiently for *all* problem instances.

The results in Table 1 also indicate that computing power indices and testing for dummy agents are also hard in all the TG forms we have examined. At a first glance, one might be tempted to suggest that this makes power indices unsuitable for solving TGs. In other words, one might claim that despite the good axiomatic properties of such indices as *fair* allocations of the rewards, agents would not be able to use them in practice to reach such fair agreements due to the computational barrier. We wish to emphasize that this is not the case. Although our results show that computing power indices exactly is hard for TGs, they can certainly be *approximated* to a high degree of accuracy [13, 27, 31, 40, 61].

One method for approximating power indices is [13]. It allows approximately computing both the Shapley value and the Banzhaf power index, and has a running time quadratic in the required accuracy. This is a randomized approach, and admits a small error probability δ, which can be made as small as desired, at the expense of increasing the runtime by $O(\log \frac{1}{\delta})$. This approach works for any game where the value of a coalition can be computed in polynomial time, so it is suitable to our TGs.

Using the power index approximation approaches discussed above, we can tractably compute a fair allocation of the utility in TGs. This allows agents to find fair agreements on dividing utility in supply chains, or to determine the importance of experts in a proof system domain. The method of [13] for approximating power indices is quite simple. Rather than examining all the possible coalitions, it randomly selects many coalitions, and averages the contribution of an agent across these sampled coalitions. For example, in the proof system domain, a naive method for computing power indices requires examining every expert subset and determining whether they can generate a proof, effectively enumerating over the many possible alternative proofs of a certain theorem. Alternatively, the approximation method randomly *samples* expert subsets, and examines the number of such subsets where an expert is critical in generating the proof (or in other words, the proportion of sampled expert coalitions that can generate the proof but cannot do so without the expert in question).

As Table 1 indicates there are some computational barriers to applying game theoretic solutions on TGs. However, we believe that such barriers do not hinder the significance of the TG model. As the table indicates, some game theory constructs, such as coalition values, veto agents and the core, can be computed exactly and tractably using the approaches we present. Other constructs, such as the Shapley value and Banzhaf index, may be hard to compute exactly, but can at least be accurately estimated using existing approaches.

5 Bounded transformation games

The results in Table 1 indicate that some problems in TGs are hard, even if we only allow simple transformations—testing for dummy agents is co-NP-complete, and computing the Shapley value or Banzhaf power index is #P-hard. This makes it very unlikely that we could find an algorithm that solves these problems and is polynomial in the number of agents, resources and transformations. We now show that these problems become tractable if the number of resources and transformations is bounded by a constant, even if the number of agents is unbounded.

In the rest of this section, we assume that the number of resources is bounded from above by a certain constant c_{res} and that the number of transformations is bounded from above by a certain constant c_{trans}. We call such a domain a *bounded* TG domain. We show that in bounded TG domains it is possible to solve the DUMMY, SHAPLEY and BANZHAF problems in polynomial time.

Two agents $i, j \in I$ in a coalitional game Γ with the characteristic function $v : 2^I \to \mathbb{R}$ are called *equivalent* if for any coalition C such that $i \notin C$ and $j \notin C$ we have $v(C \cup \{i\}) = v(C \cup \{j\})$. In other words, two agents are equivalent if we can add either one to a coalition that contains neither of them and obtain the same value. We can partition the set of agents in any game to disjoint equivalence classes $A_1, \ldots, A_k \subseteq I$ such that all the agents in each of the A_i's are equivalent.

We show that in bounded TG domains, the number of different agent equivalence classes is bounded by a certain constant.

Lemma 1 *Consider a bounded TG domain, where the number of resources is bounded by c_{res} and the number of transformations is bounded by c_{trans} (c_{res} and c_{trans} are constants). Then the agents in the game can be partitioned into a constant number q of agent equivalence classes. The number q of agent equivalence classes is bounded from above by the constant $b_{ec} = 2^{c_{res}} \cdot 2^{c_{trans}}$.*[14]

[14] b_{ec} stands for the bound on agent equivalence classes.

Proof We note that the value of a coalition in any of the TG forms we've defined only depends on the resources and transformations endowed by the coalition members. In other words, if two coalitions are endowed with the same set of initial resources and the same set of transformations, then they obtain the same value in the game. Therefore, if two agents $i, j \in I$ are endowed with the same set of resources and the same set of transformations they can perform, they must be equivalent agents. Since the number of resources is at most c_{res}, there are at most $2^{c_{res}}$ different sets of initial resources an agent can have. Similarly, there are at most $2^{c_{trans}}$ sets of transformations an agent may have. Since two agents who have both the same set of resources they are endowed with and the same set of transformations they can perform must be equivalent, there are at most $b_{ec} = 2^{c_{res}} \cdot 2^{c_{trans}}$ equivalence classes of agents. □

A recent paper [72] provides complexity results regarding coalitional games where the number of different equivalence classes is bounded.[15] They describe algorithms for determining whether an agent is a dummy and for computing power indices, which work in time polynomial in the number of agents and the bound on the number of equivalence classes. These algorithms use an oracle for computing the value of a coalition. The approach of [72] falls in the general framework of fixed-parameter tractable approaches [34], where the running time of the algorithm is polynomial in the input size, but may be exponential in a certain parameter of the input (the inputs are restricted and the only inputs allowed are ones where this parameter is smaller than a certain constant). In our case, the parameter of the input is the number of agent equivalence classes, which Lemma 1 shows to be smaller than a certain constant b_{ec}. Applying the complexity results of [72] we obtain the following theorem:

Theorem 5 *In bounded TG domains, DUMMY, SHAPLEY and BANZHAF are in P for UTGs, DTGs and TTGs with simple transformations, and for UTGs and TTGs with complex transformations.*

Proof In bounded TG domains the number of different equivalence classes is at most the constant b_{ec}. In all these TG forms we can compute the value of a coalition in polynomial time (see Theorem 1), which provides the polynomial oracle for computing the value of a coalition. Therefore the results of [72] provide the required algorithm. □

The above positive result shows that the complexity of the hard problems in TGs stems from the number of different resource and transformation combinations, and not from the number of the agents. This indicates that these problems can be tractably solved in TG domains where there are only few resources and transformations.

[15]They use the term "agent types" instead of agent equivalence classes.

6 Transformation games with costs

An important assumption in the work presented above is that transformations can be carried out at no cost. In many scenarios, this is not the case. For example, suppose we wish to derive a resource r_g from base resources R, and can do this either using a powerful but expensive computer or using a slower but cheaper one. Such trade-offs are ubiquitous in real-world problem solving. We model TGs with costs as follows. Every transformation t has cost $c(t) \in \mathbb{R}^+$. Given a coalition C and a resource r, we denote by $h(C, r)$ the minimum cost needed to obtain r from R_C, which is the sum of transformation costs in the minimal sequence of transformations from R_C to r. If r cannot be obtained from R_C, we define $h(C, r) = \infty$. The goal resource r_g has the value $v(r_g) \in \mathbb{R}^+$.

Definition 11 CTG: A TG with costs (CTG) with the goal resource r_g and the cost function $c : D \to \mathbb{R}^+$ is the game where the value of a coalition C is the value of the goal resource r_g minus the minimum cost needed to obtain r_g from R_C—if this latter difference is positive, and 0 otherwise. Thus, $v(C) = \max(0, v(r_g) - h(C, r_g))$.

The most basic question regarding a TG with costs is computing the value of a coalition C. This question is equivalent to finding the minimal cost for the coalition C to achieve the target resource r_g. TGs where each transformation takes a *single* resource and transforms it into another resource can be expressed as a graph where resources and products are the vertices, and directed edges represent the transformations, connecting the base resource of a transformation to the resource generated by the transformation. In such a representation we can weight the edges according to the cost of the transformation. In such domains, given a coalition C we can eliminate all edges representing transformations not owned by the coalition members. In this induced graph, the minimal cost path from any resource owned by the coalition members to the target resource is $h(C, r_g)$, the minimum cost for C to obtain r_g. We can compute the minimal cost path from the coalition's resources in polynomial time using Dijkstra's algorithm or using the Floyd-Warshall algorithm (see [29] for details regarding these algorithms). The above approach is not suitable for TGs where a transformation may convert *several* base resources into a product. We now propose a polynomial algorithm for such TGs.

Our Algorithm 1 for computing coalitions values in a CTG is somewhat similar to Dijkstra's algorithm. However, in general TGs with costs a transformation takes several base resources and generates a product. Thus, the TG can be represented as a hypergraph (rather than a graph), as follows. We define for every resource $r \in R$ a vertex in a hypergraph, v_r. We identify with every transformation $t = \langle\{r_1, \ldots, r_l\}, r\rangle$ an hyperedge $e_t = \langle\{v_{r_1}, \ldots, v_{r_l}\}, v_r\rangle$. We denote: R – resources, C – coalition, r_g – target resource, D_C – C's transformations. We first review Dijkstra's algorithm, as our approach is very similar, but adapted to handle hypergraphs rather than graphs.

Dijkstra's algorithm operates by maintaining and updating a bound $\lambda(v)$ for each vertex v. The value $\lambda(v)$ bounds the minimal cost for reaching v from above, and this bound is *relaxed* (lowered) during the algorithm's run. In Dijkstra's algorithm, given a vertex u with current bound $\lambda(u)$ and edge $e_{u,v}$ with cost c_e, relaxing the bound for edge $e_{u,v}$ simply entails checking whether $\lambda(u) + c_e < \lambda(v)$. If this is the case then $\lambda(v)$ is set (lowered) to be $\lambda(u) + c_e < \lambda(v)$. Dijkstra's algorithm initializes $\lambda(s) = 0$ for the source s and $\lambda(v) = \infty$ for any other vertex v. It then simple iterates by choosing the vertex of minimal bound (v such that $\lambda(v)$ is minimal), and performs relaxations for all the outgoing edges from that vertex, and terminates once this vertex of minimal bound is the target vertex t. As discussed in [29], in fact it suffices to simply apply a relaxation for *all* the edges (in any order), and repeat this $|E|$ times (where $|E|$ is the number of edges).

Our algorithm also maintains a bound from above on the minimal cost for generating a resource v_r, called $\lambda(v_r)$. Consider a transformation $t = \langle\{r_1, \ldots, r_l\}, r\rangle$ of cost $c(t)$, and assume the bounds for $r_1, \ldots, r_l$ are set to finite numbers $\lambda(v_1), \ldots, \lambda(v_l)$. If $\lambda(v_1), \ldots, \lambda(v_l)$ are indeed correct bounds from above (i.e., it is possible to generate v_i from the base resources of the coalition with price at most $\lambda(v_i)$), then r can be generated with cost at most $c(t) + \sum_{i=1}^{l} \lambda(v_i)$ (by generating $v_1, \ldots, v_l$ and then applying t). We can thus consider a relaxation procedure for a transformation (hyperedge) $t = \langle\{r_1, \ldots, r_l\}, r\rangle$ which checks if $\lambda(v_r) < c(t) + \sum_{i=1}^{l} \lambda(v_i)$ and if so, sets $\lambda(v_r)$ to be $c(t) + \sum_{i=1}^{l} \lambda(v_i)$. We can then initialize $\lambda(v_R) = 0$ for any resource r that the coalition members own, and initialize $\lambda(v) = \infty$ for all the other resources. Following similar arguments to those given in [29], it suffices to perform a relaxation for each of the transformations owned by the coalition members and repeat this $|T_C|$ times (where T_C is the set of transformations owned by the coalition).[16] This algorithm has polynomial time, as it performs at most $|T|^2$ relaxations (where T is the set of all transformations). Thus computing the value of a coalition can be done in polynomial time.

We now propose Algorithm 1, which is another polynomial algorithm for computing the value of a coalition, more similar to Dijkstra's algorithm. It also maintains a bound $\lambda(v_r)$ for resource r, which bounds the minimal cost of obtaining r from above. To obtain the optimal sequence of transformations as well, for each resource r we also maintain $S(v_r)$, which is the set of transformations in the minimal cost transformation path for generating v_r found so far. Our Algorithm 1 also operates by applying relaxations on the transformations, and makes use of the subsidiary procedure Total-Cost, which computes the transformations in the path from R_C to r, summing their costs to get the total path cost. Total-Cost($t = \langle\{r_1, \ldots, r_l\}, r\rangle$) returns a tuple ($cost$, S) where $cost$ is the cost bound when using transformation t, and S is the required set of transformations (according to the current λ bounds). This pair ($cost$, S) is a candidate for relaxation, and is used in the main procedure Compute-Coalitional-Value if indeed it allows for a better bound than the current $\lambda(v_r)$ (the relaxation itself). We first provide the formal description of the algorithm, then prove its correctness.

[16]To show this formally one can show that after i iterations of relaxing for all transformations, for any resource r such that a minimal cost transformation sequence for obtaining r makes use of i transformations, we have $\lambda(v_r)$ set to the correct value. This can be done using a simple induction on the number of iterations of relaxing for all transformations.

Algorithm 1 Procedure Compute-Coalitional-Value (R, C, r_g, D_C):

```
1. For all $r \in R_C$ do $\lambda(v_r) \leftarrow 0$
2. For all $r \in R \setminus R_C$ do $\lambda(v_r) \leftarrow \infty$
3. For all $r \in R$ do $S(v_r) \leftarrow \emptyset$
4. $T \leftarrow D_C$ (* $T$ initially contains all the transformations coalition $C$ has*)
5. while $T \neq \emptyset$:
   (a) $t = \langle\{r_1, \ldots, r_l\}, r\rangle \leftarrow \arg\min_{t \in T}(Total - Cost(t).first)$
   (b) $tc \leftarrow Total - Cost(t).first$, $S \leftarrow Total - Cost(t).second$
   (c) if $tc == \infty$ then (* remaining transformations unreachable from $R_C$*)
       i. return $\max(0, v(r_g) - \lambda(v_{r_g}))$
   (d) if $tc < \lambda(v_r)$ then $\lambda(v_r) \leftarrow tc$, $S(v_r) \leftarrow S$
   (e) $T \leftarrow T \setminus \{t\}$
6. return $\max(0, v(r_g) - \lambda(v_{r_g}))$
```

Procedure Total-Cost ($t = \langle\{r_1, \ldots, r_l\}, r\rangle$

```
1. if $\sum_{i=1}^{l} \lambda(v_{r_i}) == \infty$ then return $pair(\infty, \emptyset)$
2. $S \leftarrow \cup_{i=1}^{l} S(v_{r_i}) \cup \{t\}$
3. $tc \leftarrow \sum_{t_i \in S} c(t_i)$
4. return $pair(tc, S)$
```

We show that the above Algorithm 1 is correct using the following lemma.

Lemma 2 *During the execution of Algorithm* 1*, for every vertex v_r, if $\lambda(v_r) < \infty$ then $S(v_r)$ contains* some *path from $V' = \{v_{r'} \mid r' \in R_C\}$ to v_r, and $\lambda(v_r)$ is equal to its length.*

Proof The proof is by induction on the order of removal of transformations from T. □

Theorem 6 *Algorithm* 1 *calculates the coalitional value of a coalition C in a CTG.*

Proof We must prove that at the end of the execution of the algorithm, $\lambda(v_{r_g})$ is equal to the minimum cost of obtaining r_g from R_C.

We prove by induction on the order of removal of the transformations from T, that when $t = \langle\{r_1, \ldots, r_l\}, r\rangle$ is removed from T, $\lambda(v_r)$ is equal to the minimum distance from $V' = \{v_{r'} \mid r' \in R_C\}$ to v_r, and $S(v_r)$ contains the corresponding shortest path (or $S(v_r) = \emptyset$ if $\lambda(v_r) = \infty$).

The base case of the induction: before removing any transformation from T, $\lambda(v_r) = 0$, and $S(v_r) = \emptyset$ for all the vertices $v_r \in V'$.

Now let us assume that the claim is correct for all the transformations removed before the transformation $t = \langle\{r_1, \ldots, r_l\}, r\rangle$, and t was removed in the n-th stage. If $\lambda(v_r)$ was not updated by t then in that stage $\lambda(v_r) < \infty$, and so it was updated in earlier stage by another transformation, and by the inductive assumption, $\lambda(v_r)$ is equal to the minimum distance from V' to v_r, and $S(v_r)$ contains the corresponding shortest path. Now suppose that $\lambda(v_r)$ was updated by t. It means that for all i, $1 \leq i \leq l$: $\lambda(v_{r_i}) < \infty$ (in the stage of removal of t), and so by the inductive assumption for all i, $1 \leq i \leq l$: $\lambda(v_{r_i})$ is equal to the minimum distance from V' to v_{r_i}, and $S(v_{r_i})$ contains the appropriate shortest path.

Suppose for the contradiction that the shortest path $T' = \{t'_1, \ldots, t'_m\} \subseteq D_C$ from V' to v_r does not pass through $S(v_{r_i})$ and t. Denote by X the vertices v which have $\lambda(v) < \infty$ at stage of removal of t (stage n).

Assume first that all the transformations in T' contain only vertices from X. For each i, $1 \leq i \leq l$, if T' passes through vertex v_{r_i}, then the sub-path of T' to v_{r_i} is minimal, and then by the inductive assumption it is of length $\lambda(v_{r_i})$. Since $\sum_{i=1}^{l} c(t'_i) < \lambda(v_r)$, $t'_m = \langle\{r'_1, \ldots, r'_w\}, r\rangle$ should have been chosen in line (a) of Algorithm 1 (the first line of the "while" loop) before t, and this is a contradiction to the fact that $\lambda(v_r)$ was updated by t. Now suppose that there are transformations in T' that contain vertices outside X. Let $t'_i = \langle\{r'_1, \ldots r'_p\}, r''\rangle$ be the first transformation in T' with $v_{r''} \notin X$. Since in the first n stages t'_i was not chosen (because in the stage n $\lambda(v_{r''}) = \infty$), it follows that $\sum_{j=1}^{i} c(t'_j) \geq \sum_{t_j \in S(v_r)} c(t_j) = \lambda(v_r)$.

On the other hand, $\sum_{j=1}^{m} c(t'_j) \geq \sum_{j=1}^{i} c(t'_j)$. And so, combining these two inequalities, we get that $\sum_{j=1}^{m} c(t'_j) \geq \lambda(v_r)$, and this is a contradiction to the fact that T' is a shorter path than $S(v_r)$. □

Proposition 7 *The DUMMY problem is co-NP-Complete for CTG. SH is co-NP-Hard, and BZ is #P-Hard for CTG.*

Proof DUMMY $\in$ co-NP for CTG, since given a coalition C and a player a_i, due to Theorem 6, it is easy to test whether $v(C) < v(C \cup \{a_i\})$ (i.e., that a_i is not a dummy player). UTG is a private case of CTG (set for all the transformations t, $c(t) = 0$, and set $v(r_g) = 1$). Therefore, all the hardness results for UTG hold for CTG as well. □

7 Related work and conclusions

This work has examined game theoretic problems in transformation games, and their relation to proof systems and supply chains. We have discussed how the Transformation Game model can capture strategic aspects of firms forming supply chains or of expert agents in a cooperative proof system, and how concepts from cooperative game theory can answer many interesting problems regarding these domains when applied to a transformation game describing the domain.

We have made use of many concepts that originated in cooperative game theory. The concept of the core originated in the work of Gillies [46], and the Shapley value was introduced by Shapley [68]. The Shapley value and similar values were used to measure power in voting games, for example through the Shapley-Shubik index [69] and the related Banzhaf power index [22]. We have examined these concepts in the context of transformation games, and have also considered computational aspects.

The computational aspects of game theoretic concepts have received much attention recently. Deng and Papadimitriou [31] showed that computing the Shapley value in Weighted Voting Games is #P-complete, and calculating both Banzhaf and Shapley-Shubik indices in weighted voting games is NP-complete [55]. Our analysis of Transformation Game domains shows similar hardness results for power indices for our model. Despite the negative results regarding computing power indices *exactly*, there are several positive results regarding computing them *approximately* in both restricted and general domains [13, 27, 31, 40, 61]. These positive results indicate that it is possible to estimate the relative importance of agents (or equivalently, facts and rules) in our Transformation Game domain. Thus, power indices can be tractably *approximated* and used to determine the criticality of facts and rules in collaborative inference.

This work is somewhat reminiscent of previous work on multi-agent supply chains. Although some attention was given to auctions or procurement in such domains, (for example for forming supply chains [5] or procurement tasks [26]), previous work gave little attention to coalitional aspects. One exception is [60], which studies stability in supply chains, but focuses on pair coalitions and situations without side payments. Models relying on transformations can also be used to analyse various kinds of combinatorial auctions, such as mixed multi-unit auctions [23, 47–49]. However, our focus in this paper is very different. Rather than examining the non-cooperative game induced by a combinatorial auction, we focus on a cooperative game and cooperative solution concepts.

7.1 Planning and planning games

Our Transformation Game model is also related to work in planning within artificial intelligence [38, 39, 45]. In planning, the goal is to obtain a plan of action that will transform some initial world state into a target world state. The components in a plan are actions from some defined action repertoire. The main differences with our work are that, in classical planning, there is no multi-agent strategic component: typically there is assumed to be a single goal to be achieved, even if there are multiple actors. Thus the questions to be addressed are different in our domain, and more closely related to solutions from cooperative game theory. One paper that is closer to ours in spirit, and studies questions of self-interested actions in planning domains, is [37]. However, that paper studies a different setting, and the analysis focuses on different solution concepts.

7.2 Related cooperative games

Previous research has considered bounded resources through threshold games, in which a coalition wins if the sum of their combined resources exceeds a stated threshold [36]. Several papers have even considered manipulations in voting domains aimed at increasing power, as measured by various power indices [2, 4, 8, 9, 76, 78–80]. Another network resource based cooperative game is defined in terms of a network flow domain [7, 19]. In this game, the agents are the edges and a coalition wins if the maximal flow that can be sent through the coalition's edges exceeds a specified threshold. In one sense such games are simpler than TGs, as they consider a single resource; in another sense they are richer, as different quantities of resource are considered. Most of these works provide hardness results regarding computing power indices, indicating that this problem is hard in many combinatorial domains.

Coalitional Resource Games (CRGs) [77] are also related to our work. In CRGs, players seek to achieve individual goals, and cooperate in order to pool scarce resources in order to achieve mutually satisfying sets of goals. The main differences are that in CRGs, players have *individual* goals to achieve, which require different quantities of resources; in addition, CRGs do not consider anything like transformation chains to achieve goals. It would be interesting to combine the models presented in this paper with those of [77]. TGs can also be considered as descended from Coalitional Skill Games [15, 18], but rather then focusing on covering a set of required skills, TGs allow an elaborate interaction where a chain of transformations convert resources into products. Another class of games relying on a theme reminiscent of TGs are Boolean Games [35, 52] where the preferences of the agents are represented as boolean logical formulas. TGs are very different from Boolean Games—they do not rely on a logical formula representation, but rather on a set of transformation rules.

TGs are also somewhat related to other forms of network games such as connectivity and flow games [3, 17, 19, 20, 51, 66, 75]; the main difference is that this previous research does not consider transformation chains. However, some of the results in this line of research are akin to our results. For example, in some of the above games, the core can sometimes be computed in polynomial time if the game is a simple cooperative game, but may become computationally hard to compute in a model with costs. Finally, our analysis of reward sharing in TGs is very distantly related to some studies on coordination in supply chains and collusion in auctions [6, 12, 30, 54]. However, our focus is on the agent collaboration in the manufacturing process, rather than on information hiding or bidding manipulations.

7.3 Future work

Our model of transformation games was deliberately intended to be rather simple, leaving much room for further research. Future work might consider variations where richer models are permitted (e.g., consumable *vs* non-consumable resources). One interesting direction is taken in [24], which presents the notion of resource interfaces. The idea is to model processes that have different resource consumption profiles at different stages of execution; one might then ask, for example, whether two different processes can be executed in parallel without exceeding some stated resource bound. Similar ideas might be applied to the model in the present paper.

Additionally, we have only considered very prominent game theoretic solutions, such as the core and Shapley value. It would be interesting to examine more refined solutions such as the least-core and nucleolus. Further, our model of supply chains does not represent uncertainty. In many realistic supply chain domains, parts of the chain may fail and harming the agents' ability to generate the end-products of the chain. Several approaches have been proposed to handling uncertainty in such domains [11, 14, 42, 58, 63, 64]. Examining the impact of uncertain agent failures in TGs is a promising direction for future research. We have also ignored the possible impact of external subsidies allowing agents to cooperate even when the core is empty [10, 16, 33, 56, 57, 65, 66]. Certain supply chains provide critical products, and governments subsidise many supply chains [70, 73], so this is also an important direction for future work.

Acknowledgements This work was partially supported by Israel Science Foundation grant #898/05, by Israel Ministry of Science and Technology grant #3-6797, and by the European Research Council under Advanced Grant 291528 ("RACE").

References

1. Aumann, R.J., Hart, S. (eds.): Handbook of Game Theory, with Economic Applications, vol. 2. North-Holland, Amsterdam (1994)
2. Aziz, H., Bachrach, Y., Elkind, E., Paterson, M.: False-name manipulations in weighted voting games. J. Artif. Intell. Res. **40**(1), 57–93 (2011)
3. Aziz, H., Lachish, O., Paterson, M., Savani, R.: Power indices in spanning connectivity games. In: Algorithmic Aspects in Information and Management, pp. 55–67 (2009)
4. Aziz, H., Paterson, M.: False name manipulations in weighted voting games: splitting, merging and annexation. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 409–416. International Foundation for Autonomous Agents and Multiagent Systems (2009)
5. Babaioff, M., Walsh, W.E.: Incentive-Compatible, Budget-Balanced, yet Highly Efficient Auctions for Supply Chain Formation. Decision Support Systems (2005)
6. Bachrach, Y.: Honor among thieves: collusion in multi-unit auctions. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 617–624. International Foundation for Autonomous Agents and Multiagent Systems (2010)
7. Bachrach, Y.: The least-core of threshold network flow games. Math. Found. Comput. Sci. **1**, 36–47
8. Bachrach, Y., Elkind, E.: Divide and conquer: false-name manipulations in weighted voting games. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 975–982. International Foundation for Autonomous Agents and Multiagent Systems (2008)
9. Bachrach, Y., Elkind, E., Faliszewski, P.: Coalitional voting manipulation: a game-theoretic perspective. In: The 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 49–54 (2011)
10. Bachrach, Y., Elkind, E., Meir, R., Pasechnik, D., Zuckerman, M., Rothe, J., Rosenschein, J.S.: The cost of stability in coalitional games. In: Algorithmic Game Theory, pp. 122–134 (2009)
11. Bachrach, Y., Kash, I., Shah, N.: Agent failures in totally balanced games and convex games. In: Proceedings of the 8th Workshop on Internet and Network Economics (2012)
12. Bachrach Y., Key, P., Zadimoghaddam, M.: Collusion in VCG path procurement auctions. In: Internet and Network Economics, pp. 38–49 (2010)
13. Bachrach, Y., Markakis, E., Resnick, E., Procaccia, A.D., Rosenschein, J.S., Saberi, A.: Approximating power indices: theoretical and empirical analysis. Journal of Autonomous Agents and Multiagent Systems **20**(2), 105–122 (2010)
14. Bachrach, Y., Meir, R., Feldman, M., Tennenholtz, M.: Solving cooperative reliability games. UAI (2011)
15. Bachrach, Y., Meir, R., Jung, K., Kohli, P.: Coalitional structure generation in skill games. In: Proceedings of the 24th Conference on Artificial Intelligence (AAAI-2010) (2010)
16. Bachrach, Y., Meir, R., Zuckerman, M., Rothe, J., Rosenschein, J.S.: The cost of stability in weighted voting games. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, pp. 1289–1290. International Foundation for Autonomous Agents and Multiagent Systems (2009)
17. Bachrach, Y., Porat, E.: Path disruption games. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 1123–1130. International Foundation for Autonomous Agents and Multiagent Systems (2010)
18. Bachrach, Y., Rosenschein, J.S.: Coalitional skill games. In: AAMAS-08 (2008)
19. Bachrach, Y., Rosenschein, J.S.: Power in threshold network flow games. Journal of Autonomous Agents and Multiagent Systems **18**(1), 106–132 (2009)
20. Bachrach, Y., Rosenschein, J.S., Porat, E.: Power and stability in connectivity games. In: AAMAS-08 (2008)
21. Bachrach, Y., Zuckerman, M., Wooldridge, M., Rosenschein, J.S.: Proof systems and transformation games. Math. Found. Comput. Sci. **1**, 78–89 (2010)
22. Banzhaf, J.F.: Weighted voting doesn't work: a mathematical analysis. Rutgers Law Rev. **19**, 317–343 (1965)
23. Cerquides, J., Endriss, U., Giovannucci, A., Rodríguez-Aguilar, J.A.: Bidding languages and winner determination for mixed multi-unit combinatorial auctions. Institute for Logic, Language and Computation (ILLC), University of Amsterdam (2006)

24. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Proceedings of the 3rd International Conference on Embedded Software (EMSOFT) (LNCS vol. 2855), pp. 117–133. Springer-Verlag, Berlin, Germany (2003)
25. Chalkiadakis, G., Elkind, E., Wooldridge, M.: Computational Aspects of Cooperative Game Theory. Morgan & Claypool (2011)
26. Chen, R., Roundy, R., Zhang, R., Janakiraman, G.: Efficient auction mechanisms for supply chain procurement. Manage. Sci. **51**(3), 467–482 (2005)
27. Conitzer, V., Sandholm, T.: Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In: Proc. of AAAI-04 (2004)
28. Conitzer, V., Sandholm, T.: Complexity of determining nonemptiness of the core. In: Proceedings ACM EC-03, pp. 230–231 (2003)
29. Cormen, T.H.: Introduction to Algorithms. The MIT Press (2001)
30. de Kok, A.G., Graves, S.C.: Information sharing and supply chain coordination. In: Supply Chain Management: Design, Coordination and Operation, p. 341 (2003)
31. Deng, X., Papadimitriou, C.H.: On the complexity of cooperative solution concepts. Math. Oper. Res. **19**(2), 257–266 (1994)
32. Denzinger, J., Kronenburg, M.: Planning for distributed theorem proving. In: Proc. KI-96 (LNAI vol. 1137), pp. 43–56 (1996)
33. Dinar, A., Ratner, A., Yaron, D.: Evaluating cooperative game theory in water resources. Theory Decis. **32**(1), 1–20 (1992)
34. Downey, R.G., Fellows, M.R., Victoria University of Wellington. Mathematics Dept.: Fixed-parameter tractability and completeness I: basic results. SIAM J. Comput. **24**(4), 873–921 (1995)
35. Dunne, P.E., van der Hoek, W., Kraus, S., Wooldridge, M.: Cooperative boolean games. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1015–1022. International Foundation for Autonomous Agents and Multiagent Systems (2008)
36. Elkind, E., Goldberg, L., Goldberg, P., Wooldridge, M.: Computational complexity of weighted threshold games. In: AAAI-2007 (2007)
37. Engel, Y., Brafman, R., Domshlak, C., Tennenholtz, M.: Planning games. In: Proceedings of the 21st International Joint Conference on Artificial Intelligence (2009)
38. Ephrati, E., Pollack, M.E., Rosenschein, J.S.: A tractable heuristic that maximizes global utility through local plan combination. In: The 1st International Conference on Multiagent Systems, pp. 94–101. San Francisco, California (1995)
39. Ephrati, E., Rosenschein, J.S.: A heuristic technique for multiagent planning. Ann. Math. Artif. Intell. **20**, 13–67. Spring (1997)
40. Faliszewski, P., Hemaspaandra, L.: The complexity of power-index comparison. In: Algorithmic Aspects in Information and Management, pp. 177–187 (2008)
41. Fisher, M., Wooldridge, M.: Distributed problem-solving as concurrent theorem proving. In: Multi-Agent Rationality MAAMAW-97. Springer-Verlag, Berlin, Germany (1997)
42. Gabarro, J., Garcia, A., Clint, M., Kilpatrick, P., Stewart, A.: Bounded site failures: an approach to unreliable grid environments. In: Making Grids Work, pp. 175–187 (2008)
43. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company (1979)
44. Genesereth, M.R., Nilsson, N.: Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers, San Mateo, CA (1987)
45. Ghallab, M., Nau, D., Traverso, P.: Automated Planning: Theory and Practice. Morgan Kaufmann Publishers, San Mateo, CA (2004)
46. Gillies, D.B.: Some theorems on n-person games. PhD thesis, Princeton University (1953)
47. Giovannucci, A., Rodriguez-Aguilar, J.A., Cerquides, J., Endriss, U.: Winner determination for mixed multi-unit combinatorial auctions via petri nets. In: Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems, p. 104. ACM (2007)
48. Giovannucci, A., Rodriguez-Aguilar, J.A., Vinyals, M., Cerquides, J., Endriss, U.: Mixed multi-unit combinatorial auctions for supply chain management. ACM SIGecom Exchanges **7**(1), 58–60 (2007)
49. Giovannucci, A., Vinyals, M., Rodriguez-Aguilar, J.A., Cerquides, J.: Computationally-efficient winner determination for mixed multi-unit combinatorial auctions. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1071–1078. International Foundation for Autonomous Agents and Multiagent Systems (2008)
50. Girard, J.-Y.: Linear logic. Theor. Comp. Sci. **50**(1), 1–102 (1987)

51. Granot, D., Huberman, G.: Minimum cost spanning tree games. Math. Program. **21**(1), 1–18 (1981)
52. Harrenstein, P., van der Hoek, W., Meyer, J.J., Witteveen, C.: Boolean games. In: Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge, pp. 287–298. Morgan Kaufmann Publishers Inc. (2001)
53. Lenat, D.B.: BEINGS: knowledge as interacting experts. In: IJCAI-75, pp. 126–133 (1975)
54. Li, L., Zhang, H.: Confidentiality and information sharing in supply chain coordination. Manage. Sci. **54**(8), 1467–1481 (2008)
55. Matsui, Y., Matsui, T.: NP-completeness for calculating power indices of weighted majority games. Theor. Comp. Sci. **263**(1–2), 305–310 (2001)
56. Meir, R., Bachrach, Y., Rosenschein, J.: Minimal subsidies in expense sharing games. In: Algorithmic Game Theory, pp. 347–358 (2010)
57. Meir, R., Rosenschein, J.S., Malizia, E.: Subsidies, stability, and restricted cooperation in coalitional games. In: Proc. of 22nd IJCAI (2011)
58. Meir, R., Tennenholtz, M., Bachrach, Y., Key, P.: Congestion games with agent failures. In: 26th AAAI Conference on Artificial Intelligence (2012)
59. Osborne, M.J., Rubinstein, A.: A Course in Game Theory. MIT Press (1994)
60. Ostrovsky, M.: Stability in supply chain networks. Am. Econ. Rev. **98**(3), 897–923 (2008)
61. Owen, G.: Multilinear extensions and the Banzhaf Value. Nav. Res. Logist. Q. **22**(4), 741–750 (1975)
62. Peleg, B., Sudholter, P.: Introduction to the Theory of Cooperative Games, 2nd edn. Springer-Verlag, Berlin, Germany (2002)
63. Penn, M., Polukarov, M., Tennenholtz, M.: Congestion games with failures. In: Proceedings of the 6th ACM Conference on Electronic Commerce, pp. 259–268. ACM (2005)
64. Penn, M., Polukarov, M., Tennenholtz, M.: Congestion games with load-dependent failures: identical resources. Games. Econom. Behav. **67**(1), 156–173 (2009)
65. Qiu, L.D., Tao, Z.: Policy on international r&d cooperation: subsidy or tax? Eur. Econ. Rev. **42**(9), 1727–1750 (1998)
66. Resnick, E., Bachrach, Y., Meir, R., Rosenschein, J.: The cost of stability in network flow games. Math. Found. Comput. Sci. **1**, 636–650 (2009)
67. Schmeidler, D.: The nucleolus of a characteristic function game. SIAM J. Appl. Math. **17**(6), 1163–1170 (1969)
68. Shapley, L.S.: A value for n-person games. In: Contrib. to the Theory of Games, pp. 31–40 (1953)
69. Shapley, L.S., Shubik, M.: A method for evaluating the distribution of power in a committee system. Am. Polit. Sci. Rev. **48**, 787–792 (1954)
70. Sheu, J.B., Chou, Y.H., Hu, C.C.: An integrated logistics operational model for green-supply chain management. Transp. Res., Part E Logist. Trans. Rev. **41**(4), 287–313 (2005)
71. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press, Cambridge, England (2008)
72. Shrot, T., Aumann, Y., Kraus, S.: On agent types in coalition formation problems. In: Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, pp. 757–764. International Foundation for Autonomous Agents and Multiagent Systems (2010)
73. Simatupang, T.M., Wright, A.C., Sridharan, R.: The knowledge of coordination for supply chain integration. Bus. Process. Manag. J. **8**(3), 289–308 (2002)
74. Vadhan, S.P.: The complexity of counting in sparse, regular, and planar graphs. SIAM J. Comput. **31**(2), 398–427 (2002)
75. Voorneveld, M., Grahn, S.: Cost allocation in shortest path games. Math. Method Oper. Res. **56**(2), 323–340 (2002)
76. Wagman, L., Conitzer, V.: Optimal false-name-proof voting rules with costly voting. In: Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 190–195 (2008)
77. Wooldridge, M., Dunne, P.E.: On the computational complexity of coalitional resource games. Artif. Intell. **170**(10), 853–871 (2006)
78. Yokoo, M., Conitzer, V., Sandholm, T., Ohta, N., Iwasaki, A.: Coalitional games in open anonymous environments. In: IJCAI-05 (2005)
79. Zuckerman, M., Faliszewski, P., Bachrach, Y., Elkind, E.: Manipulating the quota in weighted voting games. In: The 23rd National Conference on Artificial Intelligence (AAAI-2008) (2008)
80. Zuckerman, M., Faliszewski, P., Conitzer, V., Rosenschein, J.: An NTU cooperative game theoretic view of manipulating elections. In: Internet and Network Economics, pp. 363–374 (2011)