Proof Systems and Transformation Games

Yoram Bachrach[1], Michael Zuckerman[2], Michael Wooldridge[3], Jeffrey S. Rosenschein[2]

[1] Microsoft Research, Cambridge, UK
[2] The Hebrew University of Jerusalem, Israel
[3] University of Liverpool, UK

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 its combined set of initial resources to the target resource. After presenting the TG model, and discussing its interpretation, we consider possible restrictions on the transformation chain, resulting in different coalitional games. After presenting 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.

1 Introduction

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. 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 is time restrictions, in the form of deadlines. 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 model restrictions on these chains, and consider game theoretic notions and the complexity of computing them under these different restrictions. We consider 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 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 simply allows building an output resource from *one* input resource, and *complex* transformations, where a transformation may require a *set* of input resources to generate a certain output resource.

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 co-operative process, with different agents contributing their domain expertise (premises) and capabilities (proof rules). 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 [8, 10], as well as cooperative problem solving systems in general [12]. (We also believe that TGs can provide a first step towards providing a cooperative game-theoretic treatment of supply chains, although we do not explore this issue further within the present paper.)

2 Preliminaries

We briefly discuss basic game theoretic concepts that are later applied in the context of TGs (see, e.g., [13] for a detailed introduction). A *transferable utility coalitional game* is composed of a set I of n players and a characteristic function mapping any subset (coalition) of the players to a real value $v : 2^I \to \mathbb{R}$, indicating the total utility these players 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* games, v only gets 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 a *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 these gains to 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 $p(C) = \sum_{i \in \{i | a_i \in C\}} p_i$.

Game theory offers 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 $p_i \geq v(\{a_i\})$. 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 prominent solution concept focusing on 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 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. Another 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 $\beta_i(v) = \frac{1}{2^{n-1}}\sum_{S\subseteq I|a_i\in S}[v(S) - v(S\setminus\{i\})]$.

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 $r_g \in R$. In these domains, 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 abilities 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. The set of all such possible transformations (over R) is 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 $|B| > 1$. Some caveats are worth highlighting:

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 modeling choice, but it is very natural in many settings. For example, consider that derivations as corresponding to *logical proofs*. In classical logic proofs, when we derive a lemma ϕ from premises Δ, we do not "consume" Δ: both ϕ and premises Δ that were used to derive it can be used as often as required in the subsequent proof.

Formally, then, 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; R is a set of resources; for each $a_i \in I$, R_i is the set of resources with which that 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 player a_i needs in order to perform the transformation as $t_i(d) \in \mathbb{N}$.

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 as $has(C, r)$, 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 $\langle\{r_{b_1}, r_{b_2}, \ldots, r_{b_m}\}, r\rangle \in D_i$.

Definition 1. *Unrestricted-TG: An unrestricted TG (UTG) with the goal resource r_g is the game where a coalition C wins if it can derive r_g and loses otherwise: $v(C) = 1$ if $C \Rightarrow r_g$ and $v(c) = 0$ otherwise.*

We now 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 $S = \langle R_1, R_2, \ldots R_k\rangle$, such that each R_i contains one additional resource over the previous R_{i-1} (so $R_i = R_{i-1} \cup \{r'_i\}$). 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 R_{i-1} (so C is capable of a transformation $d = \langle A, r'_i\rangle$, where $A \subseteq R_{i-1}$). A sequence $S = \langle R_1, R_2, \ldots R_k\rangle$ (with k subsets) that C allows is called a $k-1$-*transformation sequence for resource r by coalition C* if $r \in R_k$ and the first subset in the sequence is the subset of resources the coalition C is endowed with, $R_1 = R_C$ (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) 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: $v(C) = 1$ if both $C \Rightarrow r_g$ and $d(C, r_g) \leq k$, and otherwise $v(c) = 0$.*

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*.[1] 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 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 $C \Rightarrow^{s+t} r$. Again, different ways of obtaining the target resource result in different time

[1] For example, 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.

bounds, and we consider the optimal way of obtaining the target resource (the *minimal* time a coalition C requires to derive r). If $C \Rightarrow r$ we denote the minimal transformation time that C needs to derive r as $t(C,r) = \min\{b \mid C \Rightarrow^b r\}$, and if C cannot derive r we denote $t(C,r) = \infty$. 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: $v(C) = 1$ if both $C \Rightarrow r_g$ and $t(C, r_g) \leq t$, and otherwise $v(c) = 0$.*

3.1 Transformation Games and Logical Proofs

Structurally, TGs are similar to logical proof systems (see, e.g., [11, 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 $\phi_i \in \Delta$ (i.e., ϕ_i is a premise) or if 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 modeled 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. Clearly the relationship between TGs and proofs is very natural: such formal proof systems can be directly modeled 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: $\frac{\phi; \quad \phi \rightarrow \psi}{\psi}$, which says that if we have derived ϕ, and we have derived that $\phi \rightarrow \psi$, then we can derive ψ. Here, ϕ and ψ are variables, which can be instantiated with any formula. The second is that we take a *strategic* view: a proof modeled within our system is obtained through a cooperative process. TGs can be understood as a formulation both of cooperative theorem proving systems [8, 10], as well as cooperative problem solving systems in general [12]. 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.

4 Problems and Algorithms

Given a TG $\Gamma = \langle I, R, R_1, \ldots, R_n, D_1, \ldots, D_n, r_g \rangle$, the following are natural problems regarding the game. COALITION-VALUE (CV): given a coalition $C \subseteq I$, compute

$v_\Gamma(C)$ (i.e., test whether a coalition is successful or not). VETO (VET): given a player a_i, check if it is a veto player, so for any winning coalition C, we have $a_i \in C$. DUMMY: given a player a_i, check if it is a dummy player, so for any coalition C, we have $v_\Gamma(C \cup \{a_i\}) = v_\Gamma(C)$. CORE: compute the set of payoff vectors that are in the core, and return a representation of all payoff vectors in it. SHAPLEY: compute a_i's Shapley value $sh_i(v_\Gamma)$. BANZHAF: compute a_i's Banzhaf index $\beta_i(v_\Gamma)$.

We now summarize 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 the problem is NP-hard 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.[2] The following table summarizes our results regarding TGs with *simple* transformations.

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

Table 1. Complexity of TG problems. If the results differ for simple and complex transformations, the results for complex transformations are given in parentheses. Key: P = polynomial algorithm; co-NPC = co-NP-complete; co-NPH = co-NP-hard.

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. 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

[2] 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 their existence, so we usually regard #P-hardness as a more negative result than NP-hardness.

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[3]).

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, $argmin_{d \in D} c(d)$. 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.*

[3] 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.

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

Now consider the problem of computing the core in TGs with simple transformations. In simple (0,1-valued) games, a well-known folk theorem tells us that the core of a game is non-empty iff the game has a veto player. Thus, in simple games, the core can be represented as a list of the veto players in 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 and TTGs with complex transformations.*

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.*

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, 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 DUMMY is co-NP-hard in UTGs is enough to show 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. Let the SAT 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 $d_{c_j} = \left\langle r_{c_j}, r_{c_{j+1}} \right\rangle$, where for the last clause c_m we have a transformation $d_{c_m} = \langle r_{c_m}, r_g \rangle$. 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 converting 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, CV is NP-hard even for TGs with a single player, and VETO is co-NP-hard.*

Proof. We reduce VERTEX COVER to DTG CV. We are given a graph $G = \langle V, E \rangle$ with $V = \{v_1, \dots, v_n\}$, $E = \{e_1, \dots, e_m\}$ such that 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 TGs exceeds a certain threshold is co-NP-hard for all the following types of TGs: UTG, DTG, 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.

Definition 4. *#SET-COVER (#SC): We are given a collection $C = \{S_1, \ldots, S_n\}$ 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. #SC is a #P-hard problem. Counting the number of vertex covers, #VERTEX-COVER, is a restricted form of #SC.*[4]

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 $S = \{t_1, t_2, \ldots, t_k\}$. 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 $d_{pow} = \langle \{r_{t_n}\}, r_g \rangle$, 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 $C = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$ wins iff it contains both a_{pow} and players who are capable of all $d_1, d_2, \ldots, d_n$. 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 $C = \{a_{pow}\} \cup \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$, and denote $S_C = \{S_{i_1}, S_{i_2}, \ldots, S_{i_k}\}$. 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 q 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 q 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.

5 TGs with Costs

In many domains, transformations have costs. 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 tradeoffs 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 set $h(C, r) = \infty$. The goal resource r_g has the value $v(r_g) \in \mathbb{R}^+$.

Definition 5. *CTG: A TG with costs (CTG) with the goal resource r_g and the cost function $c : D \rightarrow \mathbb{R}^+$ 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, $v(C) = \max(0, v(r_g) - h(C, r_g))$.*

[4] [5, 3] 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.

Algorithm 1 computes coalition values in a CTG. 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. Subprocedure Total-Cost computes the transformations in the path from R_C to r, summing their costs to get the total path cost.

Algorithm 1 Compute Coalitional Value

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) $t = \langle \{r_1, \ldots, r_l\}, r \rangle \leftarrow \arg\min_{t \in T} \mathit{Total} - \mathit{Cost}(t).\mathit{first})$
   (b) $tc \leftarrow \mathit{Total} - \mathit{Cost}(t).\mathit{first}$, $S \leftarrow \mathit{Total} - \mathit{Cost}(t).\mathit{second}$
   (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 $\mathit{pair}(\infty, \emptyset)$
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 $\mathit{pair}(tc, S)$

Theorem 5. *Algorithm 1 calculates the coalitional value of a coalition C in a CTG. The proof is omitted for lack of space.*

Proposition 1. *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 5, 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 $v(r_g) = 1$). And so all the hardness results for UTG hold for CTG as well.

6 Related Work and Conclusions

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 [1] or procurement tasks [6]), previous work gave little attention to coalitional aspects. One exception is [14], which studies stability in supply chains, but focuses on pair coalitions and situations without side payments.

Previous research considered bounded resources through threshold games, in which a coalition wins if the sum of their combined resources or maximal flow exceed a stated threshold [7, 9, 4]. 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. Coalitional Resource Games (CRGs) [15] 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 [15]. TGs can also be considered as descended from Coalitional Skill Games [3] or related to connectivity and flow games [5, 4]; the main difference is that this previous work does not consider transformation chains.

Finally, despite our hardness results, power indices can be tractably *approximated* [2] and used to determine the criticality of facts and rules in collaborative inference.

References

1. Moshe Babaioff and William E. Walsh. Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation. *Decision Support Systems*, 2005.
2. Y. Bachrach, E. Markakis, E. Resnick, A. D. Procaccia, J. S. Rosenschein, and A. Saberi. Approximating power indices: theoretical and empirical analysis. *The Journal of Autonomous Agents and Multi-Agent Systems*, 20(2):105–122, 2010.
3. Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *AAMAS-08*, 2008.
4. Y. Bachrach and J. S. Rosenschein. Power in threshold network flow games. *The Journal of Autonomous Agents and Multi-Agent Systems*, 18(1):106–132, 2009.
5. Y. Bachrach, J. S. Rosenschein, and E. Porat. Power and stability in connectivity games. In *AAMAS-08*, 2008.
6. R. Chen, R. Roundy, R. Zhang, and G. Janakiraman. Efficient auction mechanisms for supply chain procurement. *Manage. Sci.*, 51(3):467–482, 2005.
7. X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.
8. J. Denzinger and M. Kronenburg. Planning for distributed theorem proving. In *Proc. KI-96 (LNAI Volume 1137)*, pages 43–56, 1996.
9. E. Elkind, L. Goldberg, P. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *AAAI-2007*, 2007.
10. M. Fisher and M. Wooldridge. Distributed problem-solving as concurrent theorem proving. In *Multi-Agent Rationality MAAMAW-97*. Springer-Verlag: Berlin, Germany, 1997.
11. M. R. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers: San Mateo, CA, 1987.
12. D. B. Lenat. BEINGS: Knowledge as interacting experts. In *IJCAI-75*, pages 126–133, 1975.
13. M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
14. Michael Ostrovsky. Stability in supply chain networks. *American Economic Review*, 98(3):897–923, June 2008.
15. M. Wooldridge and P. E. Dunne. On the computational complexity of coalitional resource games. *Artificial Intelligence*, 170(10):853–871, 2006.