Coalitional Skill Games

Yoram Bachrach Jeffrey S. Rosenschein
School of Engineering and Computer Science
The Hebrew University of Jerusalem, Israel
{yori, jeff}@cs.huji.ac.il

ABSTRACT

We consider *Coalitional Skill Games (CSGs)*, a simple model of cooperation among agents. This is a restricted form of coalitional games, where each agent has a set of skills that are required to complete various tasks. Each task requires a set of skills in order to be completed, and a coalition can accomplish the task only if the coalition's agents cover the set of required skills for the task. The gain for a coalition depends only on the subset of tasks it can complete.

We consider the computational complexity of several problems in CSGs, for example, testing if an agent is a dummy or veto agent, computing the core of the game or testing whether the core is empty, and finding the Shapley value or Banzhaf power index of agents.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*;
J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Theory, Economics

Keywords

Computational complexity, Cooperative Game Theory

1. INTRODUCTION

Game theory has implications and uses for many real-world domains, including those involving automated agents; these domains encompass electronic commerce, auctions, and general resource allocation scenarios. As a result of the desire to embed game theoretic principles into agent systems, computational aspects of game theory and social choice have been extensively studied in recent years.

Cooperation is a key issue in many automated agent scenarios. When agents are self-interested, a stable coalition can only be formed if the gains won as a result of the cooperation are distributed in an appropriate way. Coalitional game theory considers the question of how these gains should be distributed, and provides several *solution concepts*. Several such solutions have been offered, such as the core [8], the Shapley value [13] and the nucleolus [12].

These solution concepts are often used to analyze various types of interactions; one way in which they have been employed has been to measure the *power* that agents have in real-life domains, such as parties forming a coalition in legislative bodies. This has led to the definition of several power indices, such as the Banzhaf [1] and Shapley-Shubik [14] power indices. Solutions offered by cooperative game theory have been adopted by computer scientists, who have explored their attendant computational considerations [15, 3, 4, 5].

In this paper, we consider a specific model of cooperation among agents, that of *Coalitional Skill Games (CSGs)*. In this form of coalitional games, agents must cooperate to complete certain tasks. Performing each task involves using a set of required skills, and a coalition can accomplish the task only if it covers the task's required skills. In *general* CSGs, the characteristic function of the game maps the achieved set of tasks to a real value. In the paper, we also present several natural ways to restrict the structure of this characteristic function in order to get a concise representation for these games. For example, it is possible to define the value a coalition can achieve as the *number* of tasks it accomplishes, resulting in *TCSG*—Task Count Skill Games. Another possibility is giving each task its own weight, and defining the value of a coalition as the sum of weights of the accomplished tasks, resulting in *WTSG*—Weighted Task Skill Games.

CSG is a very straightforward model, yet it is highly expressive, and can model many real-world situations. Consider, for example, several communication companies that own cellular transmitters. These transmitters allow the companies to send information to various clients. In some cases, clients may be covered by more than one company; for example, two companies may have transmitters in the same area, so the clients in this area could be covered by either company. The profits of a coalition of such companies might then depend on the set of clients to which the coalition can transmit information; the coalition might gain a certain amount of money for each such client (this amount might depend on how much the client is willing to pay), or the coalition might only be awarded a contract if it covers a certain subset of the clients.

Cite as: Coalitional Skill Games, Yoram Bachrach and Jeffrey S. Rosenschein, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons(eds.),May,12-16.,2008,Estoril,Portugal,pp.1023-1030.

Another example is voting. Consider a voting domain where a decision is carried out based on the choices of voters, and where certain agents may affect how these voters vote. Each such agent may affect a certain subset of the voters, and a coalition of agents may affect all the voters that can be affected by a member of the coalition. A coalition's utility might then depend on the number of voters it can affect; for example, the coalition may have to affect more than a certain number of voters to win an election.

Yet another example is cooperative knowledge sharing. Here, each agent has access to information regarding various propositional variables, and a coalition wins if it can ascertain the value of a certain subset of these variables.

All of the above domains can easily be described as CSGs. A task in these domains might be transmitting data to a certain client, affecting a certain voter, or finding out the value of a certain variable. Questions in these domains include: 1) how to divide the total gains of a coalition among its members; 2) finding which members of the coalition are more powerful; 3) checking whether any profit can be made without a certain member of the coalition. Game-theoretic analysis of CSGs allows for the answering of such questions.

One of the advantages of the CSG model is that its simplicity allows us to tractably solve complex problems that cannot be efficiently solved in richer models. However, despite its simplicity, the model remains combinatorially quite complex, so several questions remain computationally hard. In this paper, we explore the computational complexity of solving various problems in CSGs.

The paper proceeds as follows. In Section 2 we give some background concerning coalitional games, and define the CSG model. In Section 3 we present the main algorithms and complexity results of the paper. In Section 4 we discuss some related work regarding similar questions, and we conclude in Section 5.

2. PRELIMINARIES

In this section, we define the CSG model and game theoretic concepts that are examined in the context of CSGs.

2.1 Coalitional Game Theory Solution Concepts

A transferable utility coalitional game is composed of a set of n agents, I, and a characteristic function mapping any subset (coalition) of the agents to a real value $v : 2^I \to \mathbb{R}$, indicating the total utility these agents achieve together.

Two common assumptions about coalitional games are that they are increasing and super-additive. A coalitional game is *increasing* if for all coalitions $C' \subset C$ we have $v(C') \leq v(C)$, and is *super-additive* when for all *disjoint* coalitions $A, B \subset I$ we have $v(A) + v(B) \leq v(A \cup B)$. In super-additive games, it is always worthwhile for two subcoalitions to merge, so eventually the *grand coalition* containing all the agents will form.

In a *simple* coalitional game, v only gets values of 0 or 1 ($v : 2^I \to \{0, 1\}$). We say a coalition $C \subset I$ *wins* if $v(C) = 1$, and say it *loses* if $v(C) = 0$. An agent i is *critical* in a winning coalition C if the agent's removal from that coalition would make it a losing coalition: $v(C) = 1$, $v(C \setminus \{i\}) = 0$.

The characteristic function only defines the gains a coalition can achieve, but does not define how these gains are distributed among the agents. A payoff vector $(p_1, \ldots, p_n)$ is a division of the gains of the grand coalition among the agents, where $p_i \in \mathbb{R}$, such that $\sum_{i=1}^{n} p_i = v(I)$. We call p_i the payoff of agent a_i, and denote the payoff of a coalition C as $p(C) = \sum_{i \in \{i | a_i \in C\}} p_i$. An important question, obviously, is that of choosing the appropriate payoff vector. Game theory offers several answers to this question.

2.1.1 Individual Rationality and the Core

A minimal requirement for the payoff vector is that of *individual rationality*, which states that for any agent $a_i \in C$, we have that $p_i \geq v(\{a_i\})$—otherwise, some agent is incentivized to work alone. Similarly, we say a coalition B *blocks* the payoff vector $(p_1, \ldots, p_n)$ if $p(B) < v(B)$, since the members of B can split from the original coalition, derive the gains of $v(B)$ in the game, give each member $a_i \in B$ its previous gains p_i—and still some utility remains, so each member can get more utility. If a blocked payoff vector is chosen, the coalition is unstable. A prominent solution concept focusing on such stability is that of the core [8].

DEFINITION 1. *The core of a coalitional game is the set of all payment vectors $(p_1, \ldots, p_n)$ that are not blocked by any coalition, so that for any coalition C, we have $p(C) \geq v(C)$.*

Having a value distribution in the core indicates that no subset of the coalition is incentivized to split. In general, the core can be empty, so every possible value division in that case is blocked by some coalition.

2.1.2 The Shapley Value and Banzhaf Power Index

Another cooperative game theory solution, which defines a *single* value division, is that of the Shapley value [13]; this approach focuses on *fairness*, rather than on stability. The Shapley value of an agent depends on its marginal contribution to possible coalition permutations. We denote by π a permutation (ordering) of the agents, so $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ 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 $S_\pi(i) = \{j | \pi(j) < \pi(i)\}$.

DEFINITION 2. *The Shapley value is given by the payoff vector $sh(v) = (sh_1(v), \ldots, sh_n(v))$ 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 attempt to measure an agent's ability to change the outcome of a game, and are used (for example) to measure political power. One prominent power index is the Shapley-Shubik index, which is simply the Shapley value in a simple coalitional game. Since in such a game the value of a coalition is either 0 or 1, the formula for $sh_i(v)$ simply counts the fraction of all orderings of the agents in which agent i is critical for the coalition of its predecessors and itself. Another prominent power index, defined for any simple game, is the Banzhaf power index. The Banzhaf index depends on the number of coalitions in which an agent is critical, out of all the possible coalitions.

DEFINITION 3. *The Banzhaf power index is given by $\beta(v) = (\beta_1(v), \ldots, \beta_n(v))$ where*

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

2.2 Coalitional Skill Games (CSGs)

A *coalitional skill domain* is composed of a set of agents, $I = \{a_1, \ldots, a_n\}$, a set of tasks $T = \{t_1, \ldots, t_m\}$, and a set of skills $S = \{s_1, \ldots, s_k\}$. Each agent a_i has a set of skills $S(a_i) \subset S$,[1] and each task t_j requires a set of skills $S(t_j) \subset S$. We denote the set of skills a coalition C has by $S(C) = \cup_{a_i \in C} S(a_i)$. We say a coalition of agents $C \subset I$ can perform a task t_j if every skill required to perform the task is owned by some agent in the coalition, so $S(t_j) \subset S(C)$, and denote this by $perform(C, t_j)$. We denote the set of tasks a coalition C can perform as $T(C) = \{t_j \in T | perform(C, t_j)\}$.

By a slight abuse of notation we denote the set of skills required to perform a set of subtasks $T' \subset T$ by $S(T') = \cup_{t_j \in T} S(t_j)$. A *task value function* maps a subset of the tasks a coalition achieves to a real value: $u : 2^T \to \mathbb{R}$. We generally assume that we can freely dispose of tasks by not performing them. Thus, u is increasing; so if $T1 \subset T2$, we have $u(T1) \leq u(T2)$. Consider a coalitional skill domain; we define the coalitional skill game for that domain as follows:

Definition 4. *CSG: A CSG is the coalitional game in a coalitional skill domain, where the players are the agents of the coalitional skill domain, and the characteristic function of a coalition is the value of the tasks that coalition can perform: $v(C) = u(T(C))$.*

Lemma 1. *All CSGs are increasing coalitional games.*

Proof. Adding agents to a coalition only adds skills to that coalition, so if $C' \subseteq C$, we have $S(C') \subseteq S(C)$ and thus $T(C') \subseteq T(C)$, and $u(T(C')) \leq u(T(C))$. Therefore, if $C' \subseteq C$ we get that $v(C') \leq v(C)$, so CSGs are increasing. □

The ability to tractably answer questions regarding CSGs depends on how they are represented. A naive representation of a CSG can be exponential in $|T|$. We now define several restricted forms of CSGs, which have concise representations.

2.3 Restricted Forms of CSGs

One restricted form of CSGs expresses the value of a coalition as the number of tasks that coalition can accomplish. This restricted form of CSGs is called *TCSG*—Task Count Skill Games. A representation of the characteristic function w in a TCSG simply contains a list of the tasks and a list of required skills for each task.

Definition 5. *TCSG: Let T' be a subset of tasks. A TCSG is a CSG where $u(T') = |T'|$.*

A representation that is more general than TCSG but still has a concise representation is that of *WTSG*—Weighted Task Skill Games. In a WTSG, each task t_j has a certain weight w_j, and the characteristic function is the sum of the weights of the accomplished tasks.

Definition 6. *WTSG: Let T' be a subset of tasks. A WTSG is a CSG where each task $t_j \in T$ has a weight $w_j \in \mathbb{R}$. We denote the weight of a subset of tasks $T' \subset T$ as $w(T') = \sum_{j \in \{j | t_j \in T'\}} w_j$. In a WTSG the characteristic function u is defined as $u(T') = w(T')$.*

[1]When the context is clear, we may use S_i for $S(a_i)$.

CSGs where the function u gets only 0 and 1 values, so $u : 2^T \to \{0, 1\}$, are called *simple skill games*. We say a task subset T wins the game if $u(T) = 1$, otherwise we say T loses the game. Since for any coalition C we have $v(C) = u(T(C))$, in simple skill games v's range is also $\{0, 1\}$ and we have $v : 2^I \to \{0, 1\}$, so this is indeed a simple game.

Both TCSG and WTSG have versions that are simple games. These games require the number of completed tasks or the total weight of completed tasks to exceed a certain threshold value k for a coalition to win. These versions are called *TCSG-T* (Task Count Skill Games with Threshold) and *WTSG-T* (Weighted Task Skill Games with Threshold).

Definition 7. *TCSG-T: Let T' be a subset of tasks. TCSG is a CSG with a threshold k where $v(C) = 1$ if $|T(C)| \geq k$ and $v(C) = 0$ otherwise.*

Definition 8. *WTSG-T: Let T' be a subset of tasks. WTSG is a CSG where each task $t_j \in T$ has a weight $w_j \in \mathbb{R}$ and with a threshold k, where the characteristic function u is defined as $u(T') = 1$ if $w(T') > k$ and $w(T') = 0$ otherwise.*

The most restricted form of skill games is that of *STSG*—a Single Task Skill Game. In this case, u is defined by a single task t that requires some skill subset $S(t)$, that the agents must cover to win the game. Dropping all irrelevant skills (not required to perform t), a coalition C wins if it manages to cover all the skills, so $v(C) = 1$ if and only if $S(C) = S$.

Definition 9. *STSG: An STSG is a TCSG where there is only a single task t, so $v(C) = 1$ if $S(C) = S$ and $v(C) = 0$ otherwise.*

All these restricted forms of skill games have concise representations, since we can find a short representation for the characteristic function u. In some cases, these restrictions allow us to tractably find solutions to several questions regarding these games. However, some questions remain computationally hard even with these restrictions. These results are given in Section 3.

3. ALGORITHMS FOR CSGS

Section 2.2 discussed representations of CSGs. With general CSGs, the representation of the characteristic function may be exponential in the number of tasks. However, restricting it as is done in TCSG, WTSG, STSG (and in TCSG-T and WTSG-T) gives a representation that is always polynomial. This small representation may allow (but of course, does not guarantee) polynomial algorithms for various problems. We now define the specific problems examined in this paper. All of these problems are with regard to a CSG Γ, and sometimes with regard to a target agent a_i.

Definition 10. *COALITION-VALUE (CV): Given a coalition $C \subset I$, compute $v(C)$ (in simple games, test whether $v(C) = 1$ or $v(C) = 0$).*

Definition 11. *VETO (VET): In a simple CSG, check if a_i is a veto player, so for any winning coalition C, we have $a_i \in C$. In a general CSG, test if a_i is present in all coalitions C where $v(C) > 0$.*

Definition 12. *DUMMY (DUM): Check if a_i is a dummy player, so for any coalition C, we have $v(C \cup \{a_i\}) = v(C)$.*

Definition 13. *CORE-NON-EMPTY (CNE): Decide whether the game's core is non-empty, so there is some payoff vector in the core.*

Definition 14. *CORE (COR): Compute the set of payoff vectors that are in the core, and return a representation of all payoff vectors in it.*

Definition 15. *SHAPLEY (SH): Compute a_i's Shapley value $sh_i(v_\Gamma)$.*

Definition 16. *BANZHAF (BZ): Compute a_i's Banzhaf power index $\beta_i(v_\Gamma)$.*

We summarize the results from this paper in Table 1, which gives the computational class of the above problems for each CSG restriction defined above. We prove the results in the rest of the section.

	STSG	TCSG	WTSG	TCSG-T	WTSG-T
CV	P	P	P	P	P
VET	P	P	P	P	P
DUM	P	P	P	co-NPC	co-NPC
CNE	P	co-NP	co-NP	P	P
COR	P	N/A	N/A	P	P
SH	?	?	?	NPH	NPH
BZ	#P-C	#P-C	#P-C	#P-C	#P-C

Table 1: Complexity of CSG problems
P—polynomial algorithm; NPC/co-NPC—NP-complete/co-NP-complete; co-NP—in co-NP; NPH—NP-hard; #P-C—#P-complete; ?—unknown; N/A—depends on the core representation.

3.1 Proofs

3.1.1 Coalition Value

Theorem 1. *COALITION-VALUE is in P, for all the following types of CSGs: STSG, TCSG, WTSG, TCSG-T, WTSG-T.*

Proof. Given a certain coalition C, it is simple to compute $S(C)$ in polynomial time, as the union of all the skills of the agents in C. Thus, we can compute the set of tasks $T(C)$ accomplished by that C: for each $t_j \in T$ we check if $S(t_j) \subset S(C)$. Given $T(C)$ in all these game forms, we can easily calculate $v(C)$ (as $|T(C)|$ or $w(T(C))$, or by checking if these are above the threshold). □

3.1.2 Veto

Theorem 2. *VETO is in P for all the following types of CSGs: STSG, TCSG, WTSG, TCSG-T, WTSG-T.*

Proof. A veto agent a_i in Γ is present in all winning coalitions (for non-simple games we call a coalition C winning if $v(C) > 0$). A non-veto player has some winning coalition C that does not contain him. Denote $I_{-a_i} = I \setminus \{a_i\}$. CSGs are always increasing (by Lemma 1), so if $v(I_{-a_i}) = 0$ then for every $C \subset I_{-a_i}$ we have $v(C) = 0$. Thus if $v(I_{-a_i}) = 0$, then a_i is a veto player, and if $v(I_{-a_i}) > 0$ then I_{-a_i} is a winning coalition that does not contain a_i, so a_i is not a veto player. As seen in Theorem 1, in all these games we can compute $v(C)$ in polynomial time, and in order to test if a_i is dummy, we can simply compute $v(I_{-a_i})$. □

3.1.3 Dummy

We now consider testing whether an agent is a dummy. First note that DUMMY is *in* co-NP for all types of games, since due to Theorem 1, given a coalition C, we can compute $v(C \cup \{a_i\})$ and $v(C)$ in polynomial time, and see if $v(C \cup \{a_i\}) > v(C)$. We denote the set of agents who do not cover the skill s by $I_{-s} = \{a_j \in I | s \notin S(a_j)\}$. I_{-s} can be calculated in polynomial time by going over each agent's skill list, and removing those whose skill list contains s. The algorithms for testing if an agent is a dummy depends on the following lemma.

Lemma 2. *If a_i is a* non-dummy *in an STSG then there is some skill $s \in S_i$ such that I_{-s} covers $S \setminus S_i$ (so $S \setminus S_i \subset S(I_{-s})$).*

Proof. Suppose a_i is not a dummy. Then it contributes to some coalition C, which means C covers $S \setminus S_i$ (so $C \cup \{a_i\}$ is winning), but lacks some skill $s \in S_i$ (so C is losing). If C covers $S \setminus S_i$, then any superset of it also covers $S \setminus S_i$. I_{-s} is a superset of C, since C lacks the skill s (which means every agent $a_j \in C$ lacks s). Thus, for that skill s we get that I_{-s} covers $S \setminus S_i$. □

Theorem 3. *DUMMY is in P for STSGs.*

Proof. We can iterate through all skills $s \in S_i$, and given each skill $s \in S_i$ calculate I_{-s} and check if it covers $S \setminus S_i$. If it does, a_i is not a dummy (it contributes to I_{-s}). If there is no skill $s \in S_i$ for which I_{-s} covers $S \setminus S_i$, then through Lemma 2, a_i is a dummy player. □

Theorem 4. *DUMMY is in P for TCSG and WTSG.*

Proof. Let Γ be a WTSG, with tasks $t_1, \ldots, t_m$. Let Γ_i be the STSG with the single task t_i, with the same agents and skills of Γ. Suppose a_i is not a dummy in Γ, so for some $C \subset I$ we have $v(C \cup \{a_i\}) > v(C)$. Then for at least one task t_j, C cannot achieve t_j without a_i, so a_i is not a dummy in Γ_j. If a_i is not a dummy in some Γ_j, there is some coalition C which cannot achieve t_j without a_i, so for that coalition in Γ we also have $v(C \cup \{a_i\}) > v(C)$, so that agent is not a dummy in Γ. Thus, in order to test if an agent is not a dummy in a WTSG Γ, it is enough to test this for $\Gamma_1, \ldots, \Gamma_m$. If the agent is not a dummy in *any* of them, he is not a dummy in Γ, and if he is a dummy in *all* of them, he is a dummy in Γ as well. Since TCSG is a restricted class of WTSG (where the weight of each task is 1), the same algorithm works for TCSGs as well. □

While DUMMY is polynomial in TCSG and WTSG, it is co-NP-complete in TCSG-T and WTSG-T.[2] We show this by a reduction from 3SAT, a well-known NP-hard problem.

Definition 17. *3SAT: We are given a propositional formula over n propositional variables $p_1, \ldots, p_n$, denoted $\psi = c_1 \wedge c_2 \wedge \ldots \wedge c_m$, where c_i is a disjunction of three literals $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$ (each literal is the propositional variable p_j or its negation $\neg p_j$). We are asked if there is an assignment that satisfies ψ.*

[2]This is easy to show for WTSG-T. In [11] it is shown that testing if an agent is a dummy is hard in weighted voting games. When for each agent a_i there is a single task t_i which requires a skill s_i that only a_i owns, the WTSG-T becomes a weighted voting game—so we get a natural reduction from weighted voting games. Proving the same for TCSG-T requires a different reduction.

We show that DUMMY in TCSG-T is co-NP-complete by showing that a restricted case of testing whether an agent is a non-dummy is NP-hard. Consider the restricted case of a TCSG-T Γ with a threshold $k+1$, that has a certain task t which only requires one skill s (so $S(t) = \{s\}$); and where an agent a_i is the *only* agent with a certain skill s; and where no task other than t requires the skill s. Adding a_i to any coalition C makes that coalition able to complete exactly one more task, t. A coalition in Γ wins if it covers at least $k+1$ tasks. Thus, a_i is a *non-dummy* in Γ if and only if there is a coalition of agents (without a_i) that covers exactly k tasks (denoted COMPLETE-K-TASKS). Thus, testing if a subset of the agents covers exactly k tasks is a restricted case of testing if an agent is a *non-dummy* in TCSG-T.

THEOREM 5. *DUMMY is co-NP-complete for TCSG-T and WTSG-T.*

PROOF. We have noted that DUMMY, both in TCSG-T and in WTSG-T, is in co-NP; it remains to show that it is co-NP hard. TCSG-T is a restricted form of WTSG-T, so it is enough to show this for TCSG-T. We do this by showing a reduction from 3SAT to COMPLETE-K-TASKS.

Given the 3SAT formula $\psi = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ over n propositional variables $p_1, \ldots, p_n$ (where $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$), we construct a TCSG-T game. For every propositional variable in ψ, the game has two skills s_{p_i} and $s_{\neg p_i}$. For every clause c_j in ψ the game has a skill s_{c_j}. For every clause c_j in ψ the game has three agents, $a_{c_j,1}, a_{c_j,2}, a_{c_j,3}$. The skills of $a_{c_j,x}$ depend on the literal x of c_j, and $S(a_{c_j,x}) = \{s_{c_j}, s_{l_{j,x}}\}$. For example, if we have $c_1 = p_1 \vee \neg p_2 \vee \neg p_3$, we create 3 agents: agent $a_{c_1,1}$ with skills $S(a_{c_1,1}) = \{s_{c_1}, s_{p_1}\}$, agent $a_{c_1,2}$ with skills $S(a_{c_1,2}) = \{s_{c_1}, s_{\neg p_2}\}$ and agent $a_{c_1,3}$ with skills $S(a_{c_1,3}) = \{s_{c_1}, s_{\neg p_3}\}$. For each clause c_i we also create a task t_{c_i}, which requires the skill s_{c_i}, so $S(t_{c_i}) = \{s_{c_i}\}$. For each propositional variable p_i we create $m+1$ tasks $t_{(p_i,\neg p_i,1)}, \ldots, t_{(p_i,\neg p_i,m+1)}$, each of which requires the skills $S(t_{(p_i,\neg p_i,j)}) = \{s_{p_i}, s_{\neg p_i}\}$. The purpose of these tasks is to eliminate covers where both p_i and $\neg p_i$ are chosen.

Suppose there is a satisfying truth assignment A for ψ, in which the variables assigned true are $p_{t1}, \ldots, p_{tx}$ and the variables assigned false are $p_{f1}, \ldots, p_{fy}$. We construct a coalition from the truth assignment A as follows: each clause c_j is satisfied through at least one of the literals, say literal x in c_j, denoted $l_{j,x}$. We add the agent $a_{c_j,x}$ to C. Coalition C covers all the clauses c_j of ψ, since A is a satisfying truth assignment. On the one hand, C does not cover any of the tasks $t_{(p_i,\neg p_i,j)}$ (again, A is a valid truth assignment). Thus, if there is a satisfying truth assignment, there is a coalition of agents in the created TCSG-T game which completes exactly m tasks.

On the other hand, suppose there is a coalition C which covers exactly m tasks in the created TCSG-T game. The covered tasks cannot include any of the $t_{(p_i,\neg p_i,j)}$ tasks, since each of these have m more identical copies, and covering one of these means covering all $m+1$ of them. Thus the covered m tasks are the t_{c_j} tasks. This means C holds agents who cover the skills s_{c_j} for all m clauses c_j, and for no p_i does it cover both s_{p_i} and $s_{\neg p_i}$. We build the following truth assignment A: for each s_{p_i} covered by C, set p_i to true, and set all the other variables to false.

This truth assignment satisfies every clause, since for each c_j we have some literal in c_j matching the value in the truth assignment (or C would not cover s_{c_j}). Thus, in the reduced TCSG-T there is a coalition of agents who cover m tasks if and only if the 3SAT formula is satisfiable, so we have shown a reduction from 3SAT to COMPLETE-K-TASKS. $\square$

3.1.4 Core Not Empty and Core

We now consider the complexity of calculating the core of CSGs, or checking if it is empty. Obviously, it is harder to compute the core and return a concise representation of it. This cannot always be done, in general situations, since the core may contain infinitely many payoff vectors. Fortunately, it can be done in simple games.

Consider a simple game with no veto players. For every agent a_i there is a winning coalition that does not contain a_i. Consider a payoff vector $p = (p_1, \ldots, p_n)$ where $p_i > 0$. Since $\sum_{i=0}^{n} p_i = 1$ and since $p_i > 0$ we get that $p(C) \leq \sum_{p_j \in I_{-a_i}} p_j < 1$, so $p(C) < v(C) = 1$ and C is a blocking coalition. On the other hand, any payoff vector p which gives nothing to non-veto players is in the core, since any coalition C that can block must have $v(c) = 1$, so it must contain all the veto players; thus, it also has $\sum_{p_j \in C} p_j = 1$, and therefore is not blocking. Thus, calculating the core of simple games simply requires returning a list of veto players in that game, and checking if the core is non-empty simply requires testing if the game has any veto players.[3]

THEOREM 6. *CORE and CORE-NON-EMPTY is in P for STSG, TCSG-T and WTSG-T.*

PROOF. Due to Theorem 2, for these games we can find all the veto agents in polynomial time (by checking if each agent is a veto agent). Since the representation of the core is simply a list of veto agents, we can compute the core in polynomial time. $\square$

THEOREM 7. *CORE-NON-EMPTY is* in *co-NP for STSG, TCSG/TCSG-T and WTSC/WTSG-T.*[4]

PROOF. [10] shows that CORE-NON-EMPTY is *in* co-NP for any coalitional game where the coalitional function can be computed in polynomial time. Theorem 1 shows that this is indeed the case.[5] $\square$

3.1.5 Shapley Value and Banzhaf Power Index

We now consider calculating the Shapley value and Banzhaf power index in CSGs. As can be seen from the formulas for Shapley and Banzhaf values in Section 2.1, dummy players have a Shapley value and Banzhaf index of 0. Thus, computing either of them allows testing if an agent is a dummy player.

[3]We can also present a polynomially testable sufficient condition for emptiness of the core of the non-threshold version WTSG/TCSG. Consider an agent a_i such that I_{-a_i} cannot complete all the tasks. Such an agent must have a unique skill s required for some task $t \in T$ (so $s \in S(t)$) that no other agent has, so $s \in S(a_i)$, but $s \notin S(I_{-a_i})$. We call such an agent a *unique-skill agent*. Suppose there are no unique-skill agents, and consider some agent a_i. I_{-a_i} covers all the skills and completes all the tasks. Thus, I_{-a_i} blocks any payoff vector where $p_i > 0$, since $v(I_{-a_i}) = \sum_{t \in T} w(t)$. a_i was any agent, so for all i we have $p_i = 0$, so the core is empty.

[4]This result shows that CNE is *in* co-NP, but of course does not show that it is co-NP-complete.

[5]We thank an anonymous reviewer for directing us to [10].

Corollary 1. *SHAPLEY and BANZAHF are NP-hard in TCSG-T and WTSG-T.*

Proof. DUMMY is NP-hard in these domains, due to Theorem 5. Given the Shapley value or Banzhaf index, we can answer DUMMY by comparing the index to 0. Thus, computing these indices in these domains (or the decision problem of testing whether they are greater than some value) is NP-hard. □

Computing the Shapley or Banzhaf indices is NP-hard, but may not even be in NP, so the problem is not NP-complete in these domains. We show a stronger result of #P-completeness for the Banzhaf value, for all domains, by a reduction from #-SET-COVER. We first define two #P-complete problems:

Definition 18. *#SET-COVER (#SC): We are given a set S and a collection $C = \{S_1, \ldots, S_n\}$ that for all S_i we have $S_i \subset S$. A set cover is a subset $C' \subset C$ such that $\cup_{S_i \in C'} = S$. Given S and C, we are asked to compute the number of covers of S.*

A slightly different version requires finding the number of set covers *of size at most k*:

Definition 19. *#SET-COVER-K (#SC-K): A set-cover with size k is a set cover C' such that $|C'| = k$. Similar to Definition 18, we are given S and C,* and a target size k, *and are asked to compute the number of covers of S* of size at most k.

Below we prove that BANZHAF is #P-Complete by a reduction from #SC, but we first need to prove that #SC is #P-Complete. For this we consider a few definitions and problems. An *independent set* $V' \subset V$ is a subset of the vertices such that for every $u, v \in V'$ we have $(u, v) \notin E$. A *clique* $C \subset V$ is a subset of the vertices such that for every two vertices $u, v \in C$ we have $(u, v) \in E$. A clique with size k is simply a clique C such that $|C| = k$. Given a graph $G =< V, E >$ the *complement* graph $G^c =< V, E' >$ is the graph where $(u, v) \in E'$ if and only if $(u, v) \notin E$. Three problems related to #SC are the following:

Definition 20. *#VC: We are given an undirected graph $G =< V, E >$, and are asked to count the number of vertex covers in the graph. A vertex cover is a subset $V' \subset V$ such that for every edge $e = (u, v) \in E$ either $u \in V'$ or $v \in V'$.*

Definition 21. *#CLIQUE: We are given an undirected graph $G =< V, E >$, and are asked to count the number of cliques in the graph.*

This problem also has a slightly different version, which is known to be #P-complete.

Definition 22. *#CLIQUE-K: We are given an undirected graph $G =< V, E >$, and are asked to count the number of cliques* with size at least k *in the graph.*

#SC-K and #CLIQUE-K are known #P-complete problems [7].[6]

[6]As explained in [7], reductions that maintain the same number of solutions from SAT to VC and SC exist. However, the definition of VC and SC problems, which are known to be NP-complete, test whether a vertex-cover *of size at most k* or a set-cover *of size at most k* exist. Thus, VC-K and SC-K are known to be #P-complete, but we still require a proof that #SC (rather than #SC-K) is #P complete.

We will now show that #SC (rather than #SC-K) is also #P-Complete. #VC is a restricted case of #SC (where each subset S_i is the list of edges connected to vertex v_i), so it is enough to show that #VC is #P-complete. We first note that the number of vertex covers in a graph is the number of independent sets in a graph, since if V' is a vertex cover, $V \setminus V'$ is an independent set and vice versa. We also note that an independent set in a graph G is a clique in the complement graph G^c. Thus, the number of vertex covers in a graph G is the same as the number of cliques in its complement G^c. Thus, in order to show #SC is #P-complete, we only need to show #CLIQUE is #P-complete.

Theorem 8. *#SC and #CLIQUE are #P-complete.*

Proof. As explained above, it is enough to show that #CLIQUE is #P-complete. We do this by a reduction from #CLIQUE-K. Given a graph $G =< V, E >$ and k, we construct $|V|$ graphs. In the i'th graph the vertices of the original graphs are duplicated i times. We call each such duplicate a *layer*.

The first graph we build is a bipartite graph, with two copies (layers) of the original vertices, V_1 and V_2. The new graph $G_2 =< V_1, V_2, E_2 >$ is created so that for each vertex $v_i \in V$ we create two vertices $v_{i,1} \in V_1$ and $v_{i,2} \in V_2$. If $(v_i, v_j) \in E$, for every two indices $1 \leq x, y \leq 2$, we connect $v_{i,x}$ and $v_{j,y}$ so $(v_{i,x}, v_{j,y}) \in E_2$. We note that for each 2-clique (which is an edge) in G, we get 2 cliques in G_2, as we have 2 options from which to choose a layer for the first vertex of the clique, and 1 option to choose as a layer for the second vertex of the clique. Let c_2 be the number of 2-cliques in G. A call to #CLIQUE on G_2 thus returns $2 \cdot c_2$, so we can find c_2.

We can then construct $G_3 =< V_1, V_2, V_3, E_3 >$, which is created so that for each vertex $v_i \in V$ we create 3 vertices, $v_{i,1} \in V_1$, $v_{i,2} \in V_2$ and $v_{i,3} \in V_3$. If $(v_i, v_j) \in E$, for every two indices $1 \leq x, y \leq 3$, we connect $v_{i,x}$ and $v_{j,y}$ so $(v_{i,x}, v_{j,y}) \in E_3$. We note that for each 2-clique (which is an edge) in G, we get $\frac{3!}{(3-2)!} = 3!$ 2-cliques in G_2, as we have 3 layers from which to choose for the first vertex of the clique, and 2 options from which to choose the layer for the second vertex of the clique. We also note that for every 3-clique in G, we get $\frac{3!}{(3-3)!} = 3!$ 3-cliques in G_2, as we have 3 layers from which to choose a layer for the first vertex of the clique, 2 options from which to choose the layer for the second vertex of the clique, and 1 option for a layer for the third vertex of the clique. Let c_i be the number of i-cliques in G. A call to #CLIQUE on G_3 thus returns $3! \cdot c_2 + 3! \cdot c_3$. Since we know c_2 we can calculate c_3 from this result.

Denote $|V| = m$. We can continue this process, and build G_4 to calculate c_4, and so on until we construct G_m to calculate c_m. Similarly to what we have seen above, in this graph, for each i-clique in G we get $\frac{m!}{(m-i)!}$ cliques in G_m, so a call to #CLIQUE returns $\sum_{i=1}^{m} (\frac{m!}{(m-i)!}) \cdot c_i$. Since at this point we know the values of $c_1, c_2, \ldots, c_{m-1}$ we can calculate c_m, and obtain the number of m-cliques in G. The answer to the #CLIQUE-K problem is the number of all the cliques of size at least k, $\sum_{i=k}^{m} c_i$. Thus, given an algorithm for $\#CLIQUE$ we have constructed a polynomial algorithm for $\#CLIQUE - K$. Thus #-CLIQUE is #P-complete, and so is #SET-COVER. □

We now show that BANZHAF is #P-complete in all the restricted versions of CSGs defined in this paper. This is done by a reduction from #SC.

Theorem 9. *BANZHAF in STSG, TCSG, WTSG, TCSG-T and WTSG-T is #P-complete.*

Proof. STSG is a restricted case of all the other types of games, so it is enough to show #P-completeness of BANZHAF in STSGs. Definition 3 of the Banzhaf power in STSGs is the proportion of coalitions where a_i is critical out of all coalitions containing a_i. Since the number of coalitions containing a_i is known to be 2^{n-1}, we only need to calculate the number of coalitions where a_i is critical. First, we note this problem is in #P, since due to Theorem 1 we have a simple polynomial procedure that can test if a_i is critical in some coalition containing a_i.

We show that BANZHAF is #P-complete in STSGs by a reduction from #SC. Let the #SC instance contain subsets $S = \{S_1, \ldots, S_n\}$. We build the following STSG, with $n+1$ agents. Agent a_i has the skill set S_i, and a_{n+1} has a single new skill, so $S_{i+1} = \{s_{new}\}$, such that $s_{new} \notin S$. The BANZHAF query is regarding the Banzhaf index of a_{n+1}. Every winning coalition must cover s_{new}, which can only be done using a_{n+1}. Consider a coalition C that does not contain a_{n+1} and covers S. While C is losing, $C \cup \{a_{n+1}\}$ is winning, and a_{n+1} is critical in $C \cup \{a_{n+1}\}$. Consider a coalition C that does not contain a_{n+1} and does not cover S. C is losing, and $C \cup \{a_{n+1}\}$ is also losing, so a_{n+1} is not critical in $C \cup \{a_{n+1}\}$. Denote by x the number of coalitions that do not contain a_{n+1}, and do cover S. Since each such coalition covers S, it is a set cover in the original problem. Since a_{n+1} is not critical to any coalition that does not contain a_{n+1}, the number of coalitions where a_{n+1} is critical is exactly x. Thus, if the BANZHAF answer is $\frac{x}{2^n}$, then the #SC answer is x. Thus a polynomial algorithm for BANZHAF also solves #SC, so BANZHAF is #P-complete. $\square$

4. RELATED WORK, SIMILAR MODELS

We now consider related work, especially models similar to CSGs. The concept of the core originated in [8], and the Shapley value in [13]. Such values have been used to measure power, e.g., via the Shapley-Shubik [14] and Banzhaf [1] power indices. Computational aspects of these solution concepts have been studied. [6] showed that computing the Shapley value in weighted voting games is #P-complete, and [11] showed that calculating both Banzhaf and Shapley-Shubik indices in weighted voting games is NP-complete. Several papers deal with coalitional game representations and using them to calculate solution concepts. [2] studies cooperative games on several combinatorial structures. [6] studies games where agents are represented as nodes of a weighted graph and a coalition's value is determined by the total weight of the edges contained in it. [4] uses a representation of coalitional games that relies on super-additivity, and is concise when the number of synergies between coalitions is low. This representation allows for efficient checking of whether a given outcome is in the core, but determining whether the core is nonempty remains NP-complete.

[5] uses a decomposition of a coalitional game to several domains to ease calculating the Shapley value. In the representation in [5], testing if a certain value division is in the core is co-NP complete. While this decomposition technique eases computing the Shapley value, the same cannot be directly used for the Banzhaf index, which we consider in this paper. Also, while our representation does use a certain decomposition to various domains (or tasks), the success of such tasks depends on a set of skills, rather then a complete coalitional subgame. Thus our representation is less expressive, but allows tractably solving problems that cannot be easily solved in the more complex model proposed in [5].

[9] proposes Multi-Attribute Coalitional Games (MACG), a representation of coalitional games where the value of a coalition is described by a set of agent attributes, and functions that aggregate the attributes of all the agents to a single number. MACGs can describe any coalitional game; CSGs are a very restricted form of MACGs. Again, since CSGs are very restricted, this allows us to tractably compute answers to problems that cannot be tractably solved for general MACGs. Also, CSG is a restricted case of MACGs, so hardness results for MACGs do not always hold when restricting the input to CSG form.

4.1 Similar Models

Related research deals with similar models of cooperation among agents. A model very similar to CSGs was used in [18], where the concept of anonymity and false-name manipulations was presented. Another similar model is *Coalitional Resource Games* (CRGs) [17], a restricted form of Qualitative Coalitional Games (QCGs) [16]. We now consider similarities and differences between our model and these others.

4.1.1 Anonymous Proof Solutions

[18] considers manipulations in open, anonymous environments, where a single agent can use multiple identifiers (or false names), pretending to be multiple agents, and distribute its ability among these identifiers. This requires a model of what abilities agents have, so they can be split among their false identities. The setting examined in [18] is similar to general CSGs: there are several skills S, and each agent a_i has some subset of skills $S_i \subset S$. The model assumes that no two agents possess the same skill, so $\forall a_i \neq a_j,\ S_i \cap S_j = \phi$. The characteristic function of the game is defined on the set of skills that a coalition has: $v : 2^T \rightarrow \mathbb{R}$.

The expressiveness of the model defined in [18] is essentially equivalent to that of *general* CSGs. Obviously, it is more general, as it directly maps a subset of skills that a coalition has to the utility of that coalition, rather than requiring a definition of tasks. However, any game represented in this model can also be represented as a CSG, by defining a task for each skill (which requires exactly this skill). In this way, it is possible to map any subset of skills a coalition may have to any utility for that coalition. However, if there are even a few tasks, the CSG representation is smaller than that in [18]. Also, while our CSG model is very similar to that of [18], we study the complexity of calculating solution concepts, which relies heavily on the representation used, so the restrictions on the structure of the game and its representation must be clearly defined.

[18] mostly defines anonymous proof solution concepts, and does not consider the computational complexity of calculating solution concepts. It shows that when the utility of the coalition is divided according to the Shapley value of each agent, agents may sometimes gain by splitting their skills among several false identities, pretending to be several agents. [18] suggests anonymous proof solutions, resistant to such manipulations, but does not consider these solutions' computational complexity.[7] The focus of our work is

[7]One computational question that [18] does consider is the

the computational complexity of various solution concepts; while some of these problems are computationally hard for *general* CSGs, we suggested restricted forms of CSGs where these problems can be solved by polynomial algorithms.

4.1.2 Coalitional Resource Games

[17] presents a model called *Coalitional Resource Games*. In such games, agents are interested in achieving goals. A set of different *resources* are required to reach these goals. Each agent is endowed with different amounts of each resource, and wants to achieve one of a different subset of goals. A goal subset *satisfies* a coalition if for every agent in that coalition it contains a goal desired by that agent. A goal set is *feasible* for a coalition if that coalition has sufficient resources to achieve all the goals in that set. [17] examines several questions regarding CRGs. One main concern is the properties of goal subsets that are *successful*—both feasible and satisfying for a coalition. [17] considers the complexity of several questions such as whether a coalition has a successful goal set (NP-complete); whether a certain resource r is necessary for a coalition (co-NP complete); whether a successful goal set G' for a coalition is optimal in its use of the resource r (co-NP complete), and several similar questions.

Our model of CSGs is somewhat similar to that of CRGs. CSGs define tasks to accomplish, and CRGs defines goals desired by agents. Performing a task in CSGs requires a coalition to have a certain set of skills, and achieving a goal in CRGs requires certain resources. However, significant differences exist between the models. The CRG model allows the expression of the fact that different *quantities* of various resources are required for different tasks. Also, the CRG model does not define a coalitional game; a solution in this model is simply a goal set that is both feasible and satisfying for a coalition. However, if there are several such goal sets, it is unclear which of them is chosen; even checking for the existence of such a solution is NP-hard. Similarly to [17], the current paper focuses on the computational complexity of finding appropriate solutions to the game. However, the model it uses is more similar to that of [18].

5. CONCLUSIONS

We examined a simple model of cooperation among agents, Coalitional Skill Games (CSGs), and showed that it is a rather expressive model. We considered several restricted CSG domains, and examined computational complexity of some key problems related to game theoretic solution concepts in these domains. We showed that although such games allow calculating the value of a coalition in polynomial time in all these domains, some problems remain computationally hard. We examined the computational complexity differences between these various restricted domains.

Several questions remain open for future research, such as the complexity of calculating the Shapley value in STSG, TCSG, and WTSG, and that of computing other solution concepts in CSGs, such as the nucleolus and least-core.

6. ACKNOWLEDGMENT

This work was partially supported by Israel Science Foundation grant #898/05.

question of determining whether an agent can increase its value by splitting its skills among several false identities (it is mentioned that this is NP-complete).

7. REFERENCES

[1] J. F. Banzhaf. Weighted voting doesn't work: a mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.

[2] J. M. Bilbao. *Cooperative Games on Combinatorial Structures.* Kluwer Publishers, 2000.

[3] V. Conitzer and T. Sandholm. Complexity of manipulating elections with few candidates. In *Proceedings of the National Conference on Artificial Intelligence (AAAI 2002)*, pages 314–319, 2002.

[4] V. Conitzer and T. Sandholm. Complexity of determining nonemptiness of the core. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 230–231, 2003.

[5] V. Conitzer and T. Sandholm. Computing Shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, pages 219–225, 2004.

[6] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Math. Oper. Res.*, 19(2):257–266, 1994.

[7] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[8] D. B. Gillies. *Some theorems on n-person games.* PhD thesis, Princeton University, 1953.

[9] S. Ieong and Y. Shoham. Multi-attribute coalitional games. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 170–179, 2006.

[10] E. Malizia, L. Palopoli, and F. Scarcello. Infeasibility certificates and the complexity of the core in coalitional games. In *Proc. of the Int. Joint Conference on Artificial Intelligence*, pages 1402–1407, 2007.

[11] Y. Matsui and T. Matsui. NP-completeness for calculating power indices of weighted majority games. *Theoretical Computer Science*, 263(1–2):305–310, 2001.

[12] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, 17(6):1163–1170, 1969.

[13] L. S. Shapley. A value for n-person games. *Contrib. to the Theory of Games*, pages 31–40, 1953.

[14] L. S. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

[15] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[16] M. Wooldridge and P. E. Dunne. On the computational complexity of qualitative coalitional games. *Artificial Intelligence*, 158(1):27–73, 2004.

[17] M. Wooldridge and P. E. Dunne. On the computational complexity of coalitional resource games. *Journal of Artificial Intelligence*, 170(10):835–871, 2006.

[18] M. Yokoo, V. Conitzer, T. Sandholm, N. Ohta, and A. Iwasaki. Coalitional games in open anonymous environments. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI 2005)*, pages 509–514, 2005.