Cooperative Solution Concepts in Coalitional Skill Games

Yoram Bachrach†, David C. Parkes‡, Jeffrey S. Rosenschein§

† Microsoft Research, Cambridge, UK
‡ Harvard University, Cambridge, MA, USA
§ Hebrew University, Jerusalem, Israel

Abstract

We consider a simple model of cooperation among agents called Coalitional Skill Games (CSGs). 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, such as testing if an agent is a dummy or veto agent, computing the core and core-related solution concepts, and computing power indices such as the Shapley value and Banzhaf power index.\footnote{A preliminary version of this paper was presented at the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008). This extended version includes an analysis of problems related to the \( \epsilon \)-core and least-core, showing that such problems are NP-hard in general, but can be solved in polynomial time for certain restricted classes.}

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. In order to consider these issues, such domains are modeled as a multi-agent system, consisting of self-interested agents, with possibly conflicting preferences. As a result of the desire to embed game theoretic principles into artificial multi-agent systems, computational aspects of game theory and social choice have been extensively studied in recent years [34, 31, 26].

A specific domain on the border between game theory and computation is that of cooperative domains. 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. Cooperative 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 [23], the $\epsilon$-core and least-core [40], the Shapley value [38], and the nucleolus [36]. The simplest of these concepts is the core, where no coalition of agents can find an outcome that they all prefer (at least one strictly) by breaking away from the game and working amongst themselves.

One way in which these solution concepts 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 [11] and Shapley-Shubik [39] power indices.

Constructing real-world applications requires taking computational considerations into account, and considering the challenges of engineering efficient and robust multi-agent systems. Solutions offered by cooperative game theory have been adopted by computer scientists, who have explored their attendant computational considerations [47, 41, 14, 15, 13]. For example, agents should be able to concisely express their preferences and capabilities when interacting, and the procedures used by these agents to decide how to operate must be computationally tractable.

1.1 Motivation

This paper defines a class of games called Coalitional Skill Games, which model collaboration and are based on the abstract notions of tasks and the skills required to complete them. We begin with a few examples of domains that can be naturally modeled using our class of games.

Our first example is that of several companies which are attempting to drill and refine oil at four different locations. The first oil patch is worth $200 million, and is located under ice. The second oil patch is located under the sea, and is worth $500 million, and the third is located in a remote desert, and is worth $400 million. To obtain these values, the oil first needs to be extracted. The three oil patches mentioned above contain crude oil, and that oil requires refining to be worth any money. One additional oil patch is also located in the desert, but is a pure oil patch that requires no refining, and is worth $10 million. There are four companies that may attempt to engage in activities related to the oil—Alice’s Refineries Inc., Bob’s Oil Industries Ltd., Charlie’s Petroleum Ltd., and Dana’s Gasoline Inc., or A, B, C, D for short. Only A and C can refine oil, while B and C have ice drilling facilities. Only B and D can drill in the desert, and the only company with sea drilling capability is D. Obviously, together all the companies can exploit all the patches, with a total worth of $200 + 500 + 400 + 10 = $1110 million.

In this example, A cannot achieve any value on its own, as it cannot drill any oil, and D cannot achieve any value on its own as it cannot refine the oil or drill the pure oil in the desert. However A and D together can drill and refine all the oil in the desert and sea patches, worth $500 + 400 + 10 = $910 million. On the other hand, without D it is impossible to drill the most lucrative sea
patch, worth $500 million of the total $1110 million available in these patches, so D certainly helps in achieving the total gain. How should this total reward be split between the companies? What is the fair share each company should get, assuming they work together to bring the oil to market? What would be a stable allocation of the gains? For example, B might suggest allocating $400 million to A, $100 million to B and C, and $410 million to D. Companies A and D are unlikely to find this satisfying as they can achieve $910 million on their own, as opposed to the total $810 million they get under this proposed solution. Are there stable payoff allocations in this domain?

Another example is voting. Consider a voting domain where an alternative is selected based on the choices of voters, and where certain agents may control how these voters vote. Each such agent may control a certain subset of the voters, and a coalition of agents may control all the voters that can be controlled by any member of the coalition. A coalition’s utility might then depend on the number of voters it can control; for example, the coalition may have to control more than a certain number of voters to win an election. A similar example is cooperative knowledge sharing [18, 21]. In such domains, 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.

Yet another example, considers robotic agents and human and rescuers in a disaster zone with victims trapped in multiple locations, where each location requires various skills in order to gain access [28, 35]. For example, in the case of a fire, rescuing a trapped victim may require a firefighter to reach the victim, a stretcher carrier to evacuate the victim outside of the building, and a medic to give initial treatment. Disaster budgets may be allocated to several agencies in charge of handling such situations (for example, different provinces or states may be allocated such budgets). In case of a disaster, an agency may request the assistance of fellow agencies. For example, one agency may request additional firefighters, medics or robots from a fellow agency. How should the budget or costs be allocated to the agencies in such cases?

Our last example focuses on the domain of multi-sensor networks [42, 16]. Consider distributed array facilities, each with several sensors of different types, where each sensor covers a certain geographical area. The goal of the array is to track multiple objects, each traveling through a different path over time. Each sensor can cover a part of the area at different times, and although no single sensor can fully track the objects, some coalitions of such sensors can track an object or even all the objects. When faced with different rewards or budgets for tracking the various objects, which coalition is likely to form? How would this reward be shared?

Domains such as the ones above can be characterized naturally through the Coalitional Skill Games (CSG) model. We examine game theoretic solution concepts that allow answering questions regarding fair and stable payoff allocations in such domains.
1.2 Related Work

There are many related papers that consider solution concepts from cooperative game theory and other forms of coalitional games. Some specific closely related models are compared and contrasted in detail with our proposed CSG model in Section 4, once CSGs have been described.

The problems examined under the CSG model relate to various cooperative game theory solutions. The concept of the core originated with Gillies [23], the ϵ-core and least-core were introduced by Shapley and Shubik [40], and the Shapley value was introduced by Shapley [38].

Values similar to the Shapley value have been used to measure power, e.g., via the Shapley-Shubik [39] and Banzhaf [11] power indices. Computational aspects of these solution concepts have also been studied. Deng and Papadimitriou [17] showed that computing the Shapley value in weighted voting games is #P-complete, and that calculating both Banzhaf and Shapley-Shubik indices in weighted voting games is NP-complete [30].

Our results show that computing power indices is hard in many CSG domains. Although computing power indices in CSGs can be hard, these power indices can be approximated [5]. Several related papers deal with computing, comparing, and approximating power indices, in general and in restricted domains [32, 17, 13, 19]. Our results in this paper regarding power indices are also hardness results, so this related research is especially important, as their methods allow using power indices in practice by approximating them, rather than exactly computing them.

A key issue in computational cooperative game theory is the tradeoff between expressivity and computational complexity. Any transferable utility cooperative game (in which side payments can be made among agents to balance out value) can be represented as a long table, mapping any coalition to its value. However, this representation is exponential in the number of agents. Restricted representations may be much more concise, and much effort has been devoted to developing such languages. Although rich languages allow modeling more domains and concisely representing them, computing solution concepts and properties such as power indices can be hard given a compact representation.

There has been significant research on the trade-offs between expressiveness and computation in many representations of cooperative games. For example, Bilbao [12] studies cooperative games on several combinatorial structures. Another model considers 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 that it contains [17]. Yet another representation [15] 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. Several works deal with representing coalitional games played over networks [9, 8, 2]. Several papers also examine network domains [33, 1], studying the Cost of Stability (CoS) cooperative game solution concept [7, 10] in such games. Another key problem in cooperative games is finding the optimal coalition structure, and
it has been studied in several domains [41, 16], including the CSG domain [6].

Conitzer and Sandholm [13] use a decomposition of a coalitional game to several domains to ease calculating the Shapley value. While this decomposition technique eases the computation of the Shapley value, the same technique cannot be directly used for the Banzhaf index, which we consider in this paper. Also, while our representation does use a decomposition technique (to tasks), the success of such tasks depends on a set of skills, rather than a complete coalitional subgame. Thus our representation is less expressive, but allows for the tractable solution of problems that cannot be easily solved in the more complex decomposition model.

Ieong and Shoham [27] propose 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, and CSGs are a very restricted form of MACGs. Again, since CSGs are very restricted, we are able to tractably compute answers to problems that cannot be tractably solved for general MACGs.

Several papers deal with manipulations in cooperative games, in an attempt by agents to increase their payoffs or power. One such work by Yokoo et al. [46] is thoroughly examined in Section 4, due to its close relation with the CSG model. It deals with false-name attacks (where an agent can participate under multiple pseudonyms) in skill-based domains. Other papers have also considered false-name attacks and other perturbations to increase power [4, 48, 3]. Another closely related model by Wooldridge and Dunne [45] presents coalitional games based on resources, and is also thoroughly examined in Section 4.

1.3 Contribution
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. Central to solution concepts for coalitional games is the idea of a characteristic function, which defines a coalitional value for every subset of agents. CSGs impose a particular structure on the characteristic function of coalitional games, where this structure depends in a precise way on a problem of task allocation. In general CSGs, the characteristic function of the game maps the achieved set of tasks to a real value.

General skill games are a very expressive way of defining coalitional games. 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, or the coalition might only be awarded a contract if it covers a certain subset of the clients.

In addition, the domains presented in Section 1.1 can easily be described as CSGs. A task in these domains might be producing oil, rescuing a victim, tracking an object, transmitting data to a certain client, controlling a certain voter, or finding out the value of a certain variable. In each of these domains, a task depends on having the necessary skills for it. 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.

However, the expressive power of CSGs comes at the cost of having a representation that is exponential in the number of tasks, which results in high computational difficulty in solving various questions related to such games. A good compromise between expressiveness and conciseness is achieved by imposing additional, natural restrictions on the structure of the underlying task allocation problems, and therefore on the characteristic function of the coalitional games. We present several such restrictions. 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.

Both TCSGs and WTSGs also have simple threshold versions, where a coalition of agents either “wins” or “loses.” These are simple skill games, where the characteristic function’s range is either 0 or 1. In $TCSG-T$, task count skill games with a threshold, a coalition wins if it manages to accomplish at least $k$ tasks, and in $WTSG-T$ a coalition wins if the sum of weights of the tasks it accomplishes is more than $k$. The simplest form of skill games is that of single task skills game ($STSG$s), where a single task has to be performed, and requires a coalition to cover all the required skills for that task in order to accomplish it.

We explore the computational complexity of computing important properties of CSGs such as power indices and payoff distributions. For power indices, we examine the complexity of calculating the Shapley value and the Banzhaf power index. The Shapley value defines a distribution of the gains of the coalition that meets certain desired fairness criteria. It can also be used to measure the power of agents. The Banzhaf index is a related power index, which focuses on slightly different fairness axioms.

For payoff distributions, we first examine the complexity of testing whether a certain agent is a dummy, which means that his addition to any coalition never increases the value that coalition can achieve, and whether an agent is a veto agent, which means that no coalition can win without that agent. We treat the question of computing the core of a CSG and testing whether the core is non-empty. This is a central problem to solve in determining basic stability properties of CSGs. If the core is not empty, it is possible to distribute the gains in a stable way, so that no subset of the agents will prefer to break away. In cases where the core is empty, any division of the payoffs is unstable and it
is then typical to consider relaxed solutions such as least core, which seeks to find an allocation of gains to agents in order to minimize the maximum possible gain that any coalition could achieve by breaking away from the game.

Our results show that computing the value of a coalition and finding veto agents is tractable in all the above domains. Indeed, in many of them it is possible to test for dummy agents and also compute payoff distributions (imputations) in the core. Problems related to the least core are also generally hard, but we present positive results for restricted domains. In regard to power indices, we show that they are typically hard to compute exactly. However, these can at least be approximated using other known techniques [5]. Thus, this work shows that the CSG representation is both expressive and generally computationally tractable.

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. Section 1.2 discusses some related work regarding similar problems, and Section 4 examines closely related models in more detail. 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 Cooperative Game Theory Solution Concepts

Transferable utility (TU) coalitional games provide a model for collaboration between agents. Such games are defined in terms of a specification for the value that each subset of agents (called a coalition) can achieve, while abstracting away details regarding how this value is achieved by the coalition. In the TU-coalitional game model, agents may also share the utility generated by the coalition in any way they choose through side payments among agents.

Definition 1. A transferable utility coalitional game \( \Gamma \) is composed of a set \( I = \{a_1, \ldots, a_n\} \) of \( |I| = n \) agents, and a characteristic function mapping any subset (coalition) of the agents to a real value \( v_\Gamma : 2^I \to \mathbb{R} \), indicating the total utility that these agents achieve together. When the game \( \Gamma \) is clear from the context, we sometimes omit the \( \Gamma \) subscript, and simply denote the characteristic function \( v \).

Two common assumptions about coalitional games are that they are increasing and super-additive. A coalitional game \( \Gamma \) is increasing if for all coalitions \( C' \subseteq C \subseteq I \) we have \( v_\Gamma (C') \leq v_\Gamma (C) \), and is super-additive when for all disjoint coalitions \( A, B \subseteq I \) we have \( v_\Gamma (A) + v_\Gamma (B) \leq v_\Gamma (A \cup B) \). In super-additive games, it is always worthwhile for two sub-coalitions to merge, so that the grand coalition has the largest total utility.
In a simple coalitional game $\Gamma$, $v_\Gamma$ only gets values of 0 or 1 ($v_\Gamma : 2^I \to \{0, 1\}$). We say a coalition $C \subseteq I$ wins if $v_\Gamma(C) = 1$, and say it loses if $v_\Gamma(C) = 0$. An agent $i$ is critical in a winning coalition $C$ if the agent’s removal from that coalition would make it a losing coalition: $v_\Gamma(C) = 1$, $v_\Gamma(C \setminus \{i\}) = 0$.

The characteristic function only defines the gains a coalition can achieve, but does not define how these gains are distributed among the agents.

**Definition 2.** An imputation (also known as 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 $p(C) = \sum_{i\in C} p_i$. An important question, obviously, is that of choosing an appropriate imputation with specific properties. Game theory offers several answers to this question.

### 2.1.1 Individual Rationality, the Core, $\epsilon$-Core, and Least-Core

A minimal requirement for an imputation 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 imputation $(p_1, \ldots, p_n)$ 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. Similarly, it is possible to define the degree by which a subcoalition is incentivized to deviate from the grand coalition.

**Definition 3.** Given an imputation $p = (p_1, \ldots, p_n)$, the excess of a coalition is $e(C) = v(C) - p(C)$, which quantifies the amount the subcoalition $C$ can gain by deviating and working on its own.

Given an imputation, a coalition $C$ is blocking iff its excess is strictly positive, $e(C) > 0$. If a blocked imputation is chosen, the coalition is unstable. A prominent solution concept focusing on such stability is that of the core [23].

**Definition 4.** The core of a coalitional game $\Gamma$ is the set of all imputations $(p_1, \ldots, p_n)$ that are not blocked by any coalition, so that for any coalition $C$, we have $p(C) \geq v_\Gamma(C)$.

Having an imputation (payoff distribution) in the core indicates that no subset of the coalition is incentivized to split. The core can be empty, which occurs when every possible imputation in that case is blocked by some coalition. In such cases we must relax the requirement of the solution concept. For example, deviating from the current coalition structure and forming an alternative coalition may be costly. Thus, coalitions that only have a small incentive to drop out of the grand coalition would not do so. A relaxed solution concept that follows this rationale is the $\epsilon$-core [40]:

**Definition 5.** The $\epsilon$-core is the set of all imputations $(p_1, \ldots, p_n)$ such that the following holds: for any coalition $C \subseteq I$, $p(C) \geq v_\Gamma(C) - \epsilon$. 

---

8
Under an imputation in the $\varepsilon$-core, the excess $e(C) = v_\Gamma(C) - p(C)$ of any coalition $C$ is at most $\varepsilon$. If $\varepsilon$ is large enough, the $\varepsilon$-core is guaranteed to be non-empty. A natural question is of course finding the smallest value of $\varepsilon$ that makes the $\varepsilon$-core non-empty. This solution concept is known as the least core. Formally, consider the game $G$ and the set $\{\varepsilon \mid \text{the } \varepsilon \text{-core of } G \text{ is not empty}\}$. It is easy to see that this set is compact, and thus has a minimal element $\varepsilon_{min}$.

**Definition 6.** The least-core of the game $G$ is the $\varepsilon_{min}$-core of $G$.

The imputations in the least-core distribute the gains in a way that minimizes the worst-case deficit, or in other words, minimize the incentive of a coalition to drop out of the grand coalition. Although the least-core minimizes the worst-case deficit, it can still leave multiple possible imputations. It does not, for example, minimize the number of coalitions with this worst-case deficit, nor does it minimize the second-worst deficit. The nucleolus [36] refines the least core, selecting the lexicographically minimal core.

### 2.1.2 The Shapley Value and Banzhaf Power Index

Another cooperative game theory solution, which defines a single imputation, is that of the Shapley value [38]. This approach focuses on fairness rather than on stability. This value is the only imputation (payoff distribution) that fulfills certain fairness axioms [38]. We denote by $\pi$ a permutation of the agents, so $\pi : \{1,\ldots,n\} \to \{1,\ldots,n\}$ and $\pi$ is onto, and by $\Pi$ the set of all possible such permutations. Denote by $S_\pi(i)$ the predecessors of $i$ in $\pi$, so $S_\pi(i) = \{j \mid \pi(j) < \pi(i)\}$.

**Definition 7.** The Shapley value of a game $\Gamma$ (with characteristic function $v_\Gamma$) is given by the imputation $sh(v_\Gamma) = (sh_1(v_\Gamma),\ldots,sh_n(v_\Gamma))$ where

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

The Shapley value can be interpreted as the marginal contribution an agent makes, averaged across all possible permutations of agents that may occur. The marginal contribution of an agent in a permutation is the amount of additional utility generated when that agent is added to its predecessors in the permutation.

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, again defined for any simple coalitional game, is the Banzhaf power index [11]. The Banzhaf index depends on the number of coalitions in which an agent is critical, out of the possible coalitions.
Definition 8. The Banzhaf power index of a game $\Gamma$ (with characteristic function $v_\Gamma$) is given by $\beta(v_\Gamma) = (\beta_1(v_\Gamma), \ldots, \beta_n(v_\Gamma))$ where

$$\beta_i(v_\Gamma) = \frac{1}{2^{n-1}} \sum_{S \subseteq I \mid a_i \in S} |v_\Gamma(S) - v_\Gamma(S \setminus \{i\})|.$$  

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 $S(a_i) \subseteq S$ and each task $t_j$ requires a set of skills $S(t_j) \subseteq S$. 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 $C \subseteq I$ can perform a task $t_j$ if every skill required to perform the task is owned by some agent in the coalition, so $S(t_j) \subseteq S(C)$, and denote this by $\text{perform}(C, t_j)$. We denote the set of tasks a coalition $C$ can perform as $T(C) = \{t_j \in T \mid \text{perform}(C, t_j)\}$.

By a slight abuse of notation we denote the set of skills required to perform a set of tasks $T' \subseteq T$ 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: $u : 2^T \rightarrow \mathbb{R}$. We normalize the function $u$ such that the utility of the empty task set is zero, so $u(\emptyset) = 0$. We also generally assume that we can freely dispose of tasks by not performing them. Thus, $u$ is increasing; so if $T_1 \subseteq T_2$, we have $u(T_1) \leq u(T_2)$.

Consider a coalitional skill domain; we define the coalitional skill game for that domain as follows:

Definition 9 (CSG). A CSG is the coalitional game $\Gamma$ 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: $v_\Gamma(C) = u(T(C))$.

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 $v_\Gamma(C') \leq v_\Gamma(C)$, so CSGs are increasing. \qed

The ability to tractably answer questions regarding CSGs depends on how they are represented. A naive representation of a CSG is exponential in $|T|$ since every subset of tasks is associated explicitly with the value. We now define several restricted forms of CSGs with 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

\footnote{When the context is clear, we sometimes use $S_i$ for $S(a_i)$.}
in a TCSG simply contains a list of the tasks and a list of required skills for each task.

**Definition 10 (TCSG).** Let \( T' \subseteq T \) be a subset of tasks. A TCSG is a CSG where task value function \( u(T') = |T'| \).

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 an associated weight \( w_j \), and the characteristic function is the sum of the weights of the accomplished tasks.

**Definition 11 (WTSG).** Let \( T' \subseteq T \) be a subset of tasks. A WTSG is a CSG where each task \( t_j \in T \) has a weight \( w_j \in \mathbb{R}_+ \). The task value function is defined as \( u(T') = \frac{1}{\max\{1, |T'| - k\}} \). CSGs where the task value function \( u \) can obtain only values 0 and 1, so \( u : 2^T \rightarrow \{0,1\} \), 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 \( v_1(C) = u(T(C)) \), in simple skill games \( v_1 \)'s range is also \( \{0,1\} \) and we have \( v_1 : 2^I \rightarrow \{0,1\} \).

Both TCSG and WTSG have versions that are simple skill 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 12 (TCSG-T).** Let \( T' \subseteq T \) be a subset of tasks. TCSG-T is a CSG with a threshold \( k \) where the task value function is \( u(T') = 1 \) if \( |T'| \geq k \) and \( u(T') = 0 \) otherwise. Thus, the game has the characteristic function \( v_1(C) = 1 \) if \( |T(C)| \geq k \) and \( v_1(C) = 0 \) otherwise.

**Definition 13 (WTSG-T).** Let \( T' \subseteq T \) be a subset of tasks. WTSG-T is a CSG where each task \( t_j \in T \) has a non-negative weight \( w_j \in \mathbb{R} \) and with a threshold \( k \), where the task value function \( u \) is defined as \( u(T') = 1 \) if \( w(T') > k \) and \( u(T') = 0 \) otherwise. Thus, the game has the characteristic function \( v_1(C) = 1 \) if \( w(T(C)) \geq k \) and \( v_1(C) = 0 \) otherwise.

The most restricted form of CSGs is that of STSG—a Single Task Skill Game. In a STSG, there is only one task \( t \), whose set of required skills \( S(t) \) are all the skills in the domain, so we have \( S(t) = S \). In STSGs, the task value function is \( u(\{t\}) = 1 \), and \( u(\emptyset) = 0 \). A coalition \( C \) wins if it manages to cover all the skills, so \( v_1(C) = 1 \) if and only if \( S(t) \subseteq S(C) \), and since \( S(t) \) is the set of all skills, we can simply say that a coalition wins if it covers all the skills in the domain, so \( S(C) = S \).

**Definition 14 (STSG).** A STSG is a TCSG where there is only a single task \( t \), so \( v_1(C) = 1 \) if \( S(C) = S \) and \( v_1(C) = 0 \) otherwise.

---

3In this paper we assume that for any task \( t_j \) we have a strictly positive weight \( w_j > 0 \). However, it is also possible to define games where a task may have a negative weight.
All these restricted forms of skill games have concise representations, since we can find a short representation for the task value function, and thus also have a short representation for the characteristic function. Such restricted representations can model many situations. For example, TCSGs may allow expressing the fact that our goal is to save as many victims as possible in a coordinated rescue problem. WTSGs can express the fact that each oil patch has a different monetary worth, and our goal is to maximize profits. TCSG-T can express the fact that to be successful, we must track at least a certain number of objects in the cooperative sensor domain.

The restricted CSG models defined above are very natural, and they allow us to tractably find solutions to several questions regarding these games. However, some questions remain computationally hard even with these restrictions. For example, certain problems regarding power indices and the \( \epsilon \)-core and least-core are generally hard even in the simple case of STSG. Below, we either suggest approximation algorithms (for computing power indices), or focus on restricted domains. Least-core related problems become tractable under further restrictions on the domain. The restrictions include having a tree-like skill hypergraph structure, having a bound on the maximal number of skills an agent may have, or having a bound on the number of tasks. The exact definitions of these restrictions, as well as the algorithmic and hardness results, are given in Section 3.

### 3 Algorithms for 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. We now define the specific problems examined in this paper. All of these problems are with regard to a CSG \( \Gamma \), and sometimes with regard to a target agent \( a_i \).

**Definition 15** (COALITION-VALUE (CV)). Given a coalition \( C \subseteq I \) and CSG \( \Gamma \), compute \( v_\Gamma(C) \).

**Definition 16** (VETO:). In a simple CSG \( \Gamma \), check if \( a_i \) is a veto player, so for every winning coalition \( C \), we have \( a_i \in C \). In a general CSG, test if \( a_i \) is present in every coalition \( C \) where \( v_\Gamma(C) > 0 \).

**Definition 17** (DUMMY:). Check if \( a_i \) is a dummy player in CSG \( \Gamma \), such that for every coalition \( C \) (with \( a_i \notin C \)), we have \( v_\Gamma(C \cup \{a_i\}) = v(C) \).

**Definition 18** (CORE-NON-EMPTY (CNE):). Decide whether the cost of CSG \( \Gamma \) is non-empty.

**Definition 19** (CORE:). Return a representation of all imputations in the core of CSG \( \Gamma \).
**Definition 20** (ε-CORE-MEMBERSHIP (ECM)). Given an imputation \( p = (p_1, \ldots, p_n) \), and CSG \( \Gamma \), decide whether it is in the \( \epsilon \)-core of the game.

**Definition 21** (SHAPLEY). Compute agent \( a_i \)’s Shapley value \( sh_i(v \Gamma) \) in CSG \( \Gamma \).

**Definition 22** (BANZHAF). Compute agent \( a_i \)’s Banzhaf power index \( \beta_i(v \Gamma) \) in CSG \( \Gamma \).

We summarize the results from this paper in Table 1.\(^4\)

<table>
<thead>
<tr>
<th></th>
<th>STSG</th>
<th>TCSG</th>
<th>WTSG</th>
<th>TCSG-T</th>
<th>WTSG-T</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>VETO</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>DUMMY</td>
<td>P</td>
<td>co-NP</td>
<td>P</td>
<td>co-NP</td>
<td>co-NP</td>
</tr>
<tr>
<td>CNE</td>
<td>P</td>
<td>co-NPC</td>
<td>co-NPC</td>
<td>co-NPC</td>
<td>co-NPC</td>
</tr>
<tr>
<td>ECM</td>
<td>co-NPC</td>
<td>co-NPC</td>
<td>co-NPC</td>
<td>co-NPC</td>
<td>co-NPC</td>
</tr>
<tr>
<td>CORE</td>
<td>P</td>
<td>N/A</td>
<td>N/A</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>SHAPLEY</td>
<td>NPH</td>
<td>NPH</td>
<td>NPH</td>
<td>NPH</td>
<td>NPH</td>
</tr>
<tr>
<td>BANZHAF</td>
<td>#P-C</td>
<td>#P-C</td>
<td>#P-C</td>
<td>#P-C</td>
<td>#P-C</td>
</tr>
</tbody>
</table>

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; N/A—depends on the core representation.

Some results in the above table warrant further discussion. First, we assume that the CSG is given in its concise representation (polynomial in the number of agents, tasks and skills), and not in a full characteristic function representation (which is exponential in the number of agents). Thus, polynomial algorithms must run in time polynomial in the number of agent tasks and skills. Second, some results are not provided regarding the core (denoted “N/A”). The core may sometimes contain infinitely many imputations, so it is unclear how it should be represented. Where there is a concise representation for the core, we provide the complexity. When no such representation is known, we simply enter “N/A” in the table. Further, regarding the Shapley value, we have shown it is NP-hard, but have not shown it is in NP, nor shown that it is not in NP, so this issue remains open.

The most basic problem regarding the \( \epsilon \)-core and least-core is ECM, which tests if an imputation is \( \epsilon \)-stable. More demanding problems are finding an imputation in the \( \epsilon \)-core given a game and a target stability value \( \epsilon \), computing the minimal \( \epsilon \) that admits a non-empty \( \epsilon \)-core (finding the least-core value), or

\(^4\)Despite the hardness results for ECM, positive results for restricted domains are discussed below. Hardness of computing the Shapley value in STSGs, TCSGs and WTSGs is examined in [2] where it is shown to be NP-hard, as a consequence of the results in the conference version of this paper (also presented in this extended journal version), showing #P-completeness for computing the Banzhaf index in these domains. The remainder of the results are presented in this paper.
finding a least-core imputation. The above Table 1 shows that ECM is hard (co-NP-complete) even in the most restricted case of STSG. Examining the more demanding problems of finding an ε-core or least-core imputation becomes relevant for restricted domains where ECM is tractable.

We also develop results regarding restricted domains where the ECM problem, as well as finding ε-core imputations and computing the least-core are tractable. We formally define these more demanding problems:

**Definition 23** (ε-CORE-FIND-IMPUTATION (ECF)). Given ε, find an imputation \( p = (p_1, \ldots, p_n) \) in the ε-core of a CSG game \( \Gamma \) if one exists, and reply that no such imputation exists otherwise.

**Definition 24** (LEAST-CORE-VALUE (LCV)). Compute the least-core value in CSG \( \Gamma \), which is the minimal value \( \epsilon_{\text{min}} \) such that the \( \epsilon_{\text{min}} \)-core is non-empty.\(^5\)

We show that for tree-like STSGs with a bounded number of skills per agent, ECM, ECF and LCV are all solvable in polynomial time. Further, these problems are also solvable in polynomial time in TCSGs and WTSGs (as well as their threshold versions TCSG-T and WTSG-T) under the additional restriction that the number of tasks is bounded.

We now present the analysis of the above defined problems in the various CSG domains. Since we focus on a given game \( \Gamma \), from here on we drop the subscript \( \Gamma \) from our notation.

### 3.1 Coalition Value

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

**Proof.** Given a 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 \( S(t_j) \subseteq S(C) \). 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). \( \square \)

### 3.2 Veto Agents

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

**Proof.** A veto agent \( a_i \) in \( \Gamma \) is present in all winning coalitions (with \( v(C) = 1 \) for simple games and \( v(C) > 0 \) for general CSGs.) Consider \( C = I \setminus \{a_i\} \). If \( v(C) = 0 \) then \( a_i \) is veto since \( v(C') = 0 \) for all \( C' \subseteq C \) by Lemma 1. If \( v(C) > 0 \)

\(^5\)Obviously, a computer can represent a non-integer value up to a certain degree of accuracy, so the LCV problem requires computing \( \epsilon_{\text{min}} \) up to a certain accuracy. A tractable algorithm must be polynomial in the input size and logarithmic in the desired accuracy (i.e., polynomial in the number of bits describing the accuracy level).
then \(a_i\) is not veto. As seen in Theorem 1, we can compute \(v(C)\) in polynomial time.

### 3.3 Dummy Agents

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 that 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 depend 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 \(S \setminus S_i\).

**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) lacks some skill \(s \in S_i\). 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, \(I_s\) 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 there is such an \(s\), then \(a_i\) 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 TCSGs and WTSGs.

**Proof.** Let \(\Gamma\) be a WTSG, with tasks \(t_1, \ldots, t_m\). Let \(\Gamma_j\) be a STSG with the single task \(t_j\), with the same agents and skills as \(\Gamma\). Suppose \(a_i\) is not a dummy in \(\Gamma\), so for some \(C \subseteq I \setminus \{a_i\}\) we have \(v(C \cup \{a_i\}) > v(C)\). Then for at least one task \(t_j\), \(C\) cannot achieve \(t_j\) without \(a_i\), and \(a_i\) is not a dummy in \(\Gamma_j\). Going the other way, if \(a_i\) is not a dummy in some \(\Gamma_j\), there is some coalition \(C'\) which cannot achieve \(t_j\) without \(a_i\), so that in \(\Gamma\) we also have \(v(C' \cup \{a_i\}) > v(C)\), and \(a_i\) is not a dummy in \(\Gamma\). Thus, in order to test if an agent is not a dummy in a WTSG \(\Gamma\), 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 \(\Gamma\), and if he is a dummy in all of them, he is a dummy in \(\Gamma\) as well. Since TCSG is a restricted class of WTSG, the same algorithm works for TCSGs as well.

While DUMMY is polynomial in TCG and WTSG, it is co-NP-complete in TCSG-T and WTSG-T.\(^6\) We show this by a reduction from 3SAT, a well-known

\(^6\)This is easy to show for WTSG-T. In Matsui and Matsui [30] 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.
one more task, as a reduction from 3SAT to COMPLETE-K-TASKS, and set the threshold to be of WTSG-T, so it is enough to show this for TCSG-T. We do this by showing co-NP; it remains to show that it is co-NP hard. TCSG-T is a restricted form of agents in the created TCSG-T game that completes exactly covers all the clauses tasks in the created TCSG-T game. The covered tasks cannot include any of the

Proof. Theorem 5. DUMMY is co-NP-complete for TCSG-T and WTSG-T.

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 $\Gamma$ wins if it covers at least $k+1$ tasks. Thus, $a_i$ is a non-dummy in $\Gamma$ if and only if there is a coalition of agents (without $a_i$) that covers exactly $k$ tasks (denoted COMPLETE-K-TASKS).

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

Given the 3SAT formula $\psi = c_1 \land c_2 \land \ldots \land c_m$ over $n$ propositional variables $y_1, \ldots, y_n$ (where $c_i = l_{i,1} \lor l_{i,2} \lor l_{i,3}$), we construct a TCSG-T game with threshold $m$. For every propositional variable in $\psi$, the game has two skills, $s_{yi}$ and $s_{-yi}$. For every clause $c_j$ in $\psi$ the game has a skill $s_{cj}$ and three agents, $a_{cj,1}, a_{cj,2}, a_{cj,3}$. The skills of $a_{cj,x}$ depend on the literal $x$ of $c_j$, and $S(a_{cj,x}) = \{s_{cj}, s_{l_{j,x}}\}$. For example, if we have $c_1 = y_1 \lor \neg y_2 \lor \neg y_3$, we create 3 agents: agent $a_{c_1,1}$ with skills $S(a_{c_1,1}) = \{s_{c_1}, s_{y_1}\}$, agent $a_{c_1,2}$ with skills $S(a_{c_1,2}) = \{s_{c_1}, s_{-y_2}\}$ and agent $a_{c_1,3}$ with skills $S(a_{c_1,3}) = \{s_{c_1}, s_{-y_3}\}$. For each clause $c_i$ we also create a task $t_{ci}$, which requires the skill $s_{ci}$, so $S(t_{ci}) = \{s_{ci}\}$. For each propositional variable $y_i$ we create $m+1$ tasks $t_{(y_i, \neg y_i, 1)}, \ldots, t_{(y_i, \neg y_i, m+1)}$, each of which requires the skills $S(t_{(y_i, \neg y_i, j)}) = \{s_{y_i}, s_{-y_i}\}$. The purpose of these tasks is to eliminate covers where both $y_i$ and $\neg y_i$ are chosen.

Suppose there is a satisfying truth assignment $A$ for $\psi$, in which the variables assigned true are $y_1, \ldots, y_{lx}$ and the variables assigned false are $y_{f1}, \ldots, y_{fy}$. We construct a winning coalition that covers exactly $m$ tasks 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 $a_{cj,x}$ to $C$. Coalition $C$ covers all the clauses $c_j$ of $\psi$, since $A$ is a satisfying truth assignment. On the other hand, $C$ does not cover any of the tasks $t_{(y_i, \neg y_i, j)}$ (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 that 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.
to $(y_i, \neg y_i, t_j)$ 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 that cover the skills $s_{c_j}$ for all $m$ clauses $c_j$, and for no $y_i$ does it cover both $s_{y_i}$ and $s_{\neg y_i}$. We build the following truth assignment $A$: for each $s_{y_i}$ covered by $C$, set $y_i$ 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 $s_{c_j}$).

### 3.4 The Core

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 an imputation $p = (p_1, \ldots, p_n)$ where $p_i > 0$. Since $\sum_{i=1}^n p_i = 1$ 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 imputation $p$ that 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. As a consequence, 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.\(^7\)

**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. Since the representation of the core is simply a list of veto agents, we can compute the core in polynomial time. \(\Box\)

**Theorem 7.** CORE-NON-EMPTY is in co-NP for STSG, TCSG, TCSG-T and WTSG, and WTSG-T.\(^8\)

**Proof.** Malizia et al. [29] show 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.\(^9\) \(\Box\)

---

\(^7\)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 imputation 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.

\(^8\)This result shows that CNE is in co-NP, but of course does not show that it is co-NP-complete.

\(^9\)We thank an anonymous reviewer of the earlier conference version of this paper for directing us to [29].
3.5 The $\epsilon$-Core and Least-Core

We now consider $\epsilon$-core and least-core related problems. Determining whether a certain imputation $p = (p_1, \ldots, p_n)$ is in the $\epsilon$-core is equivalent to testing whether the maximal excess of any coalition is at most $\epsilon$. We can thus focus on the complexity of finding the maximal excess of any coalition given an imputation.\(^{10}\) For this, it is sufficient to study ECM.

Even in the simple domain of STSG, the ECM problem is coNP-Complete and equivalent to the weighted set-cover problem.

**Definition 26** (Weighted Set-Cover (WSC)). We are given a set of elements $E$ and a collection of subsets $S = \{S_1, \ldots, S_n\}$ where $S_i \subseteq E$ and $\bigcup S_i = E$, and positive weights $c_1, \ldots, c_n$, and are asked to find a subset $S' \subseteq S$ that covers $E$ (i.e., such that $\bigcup_{S_i \in S'} S_i = E$) of minimal weight (i.e., minimizing $\sum_{i \in S'} c_i$).

The restricted version of WSC where all the weights are identical is the SET-COVER problem. SET-COVER is a prominent NP-complete problem [22].

**Theorem 8.** ECM is co-NP-Complete in any of the CSG forms: STSGs, TCSG, WTSG, TCSG-T, and WTSG-T.

**Proof.** A coalition $C$ with an excess greater than $\epsilon$ (so $e(C) = v(C) - p(C) \geq \epsilon$) violates the $\epsilon$-core constraints. Thus, the decision version of ECM requires making sure that under a given imputation there does not exist a coalition with an excess of at least $\epsilon$ for some $\epsilon \geq 0$. Equivalently, ECM requires making sure that the maximal excess across all coalitions is at most $\epsilon$. Due to Theorem 1, given a coalition $C$ we can compute the value $v(C)$ and its payoff $p(C)$ in polynomial time, and thus can also compute its excess $e(C) = v(C) - p(C)$. Thus, ECM is in coNP for all of the above CSG classes.

We now show that computing the maximal excess $e_{\text{max}} = \max\{e(C) | C \subseteq I\}$ even in the restricted class of STSG is equivalent to a weighted set-cover problem. Any losing coalition $C$ has a negative excess (i.e., $p(C) \geq v(C)$) under any imputation. Thus, the maximal excess occurs for some winning coalition, and we have $e_{\text{max}} = \max\{e(C) | v(C) = 1\}$. Any winning coalition $C$ has $v(C) = 1$, so the maximal excess occurs for a winning coalition $C$ that minimizes $p(C)$. Thus, ECM is in coNP for all of the above CSG classes.

\[\square\]

\(^{10}\)Obviously, given a polynomial algorithm to compute the maximal excess of any coalition given an imputation, we can test $\epsilon$-core membership: if the maximal excess is at most $\epsilon$ then this is an $\epsilon$-core imputation and otherwise it is not. On the other hand, given the ability to test for $\epsilon$-core membership, we can also compute the maximal excess, by performing a “binary search”, querying whether the imputation is in the $\epsilon_1$-core, $\epsilon_2$-core, $\epsilon_3$-core and so on, where $\epsilon_i$ is chosen in a binary search for the “correct” maximal excess value. Thus, a polynomial algorithm for ECM allows computing the maximal excess up to any desired degree of accuracy in polynomial time.

18
The above result holds even if the imputation is always the equal imputation where \( p_i = p_j \) for any two agents \( i, j \), in which case ECM is equivalent to SET-COVER (rather than WSC). The result holds even for the equal imputation when each skill is shared by exactly two agents (i.e., for any skill \( s \) we have exactly two agents who own that skill), in which case the ECM problem translates to the prominent NP-complete VERTEX-COVER problem [22].

Unfortunately, WSC is also hard to approximate, and unless NP has slightly super-polynomial algorithms (which is highly unlikely), the best polynomial-time approximation algorithm for it achieves an approximation factor of \( \Theta(\ln n) \) [20].

A simple algorithm for computing the maximal excess (or for solving ECM) through approximating WSC requires a FPTAS for SET-COVER. Thus, to provide tractable algorithms for the ECM problem, we focus on restricting the inputs.

Turning to positive results, the problem of WSC has been studied for a class of problems in which the inputs are tree-like and have a constant bounded maximal subset size [25]. Guo and Niedermeier [25] discuss a fixed-parameter tractable approach to WSC. In domains where each of the subsets has a size at most \( b \) (where \( b \) is constant), so \( b = \max_{S_i \in S}|S_i| \) and where the subset collection is tree-like, then WSC can be solved in polynomial time.\(^{12}\) We first give the definition of a “tree-like” subset collection.

**Definition 27** (Tree-like subset collection). A collection \( S = \{S_1, \ldots, S_n\} \) of subsets over the elements \( X \) (such that \( S_i \subseteq X \)) is tree-like if it is possible to arrange the subsets of \( S \) in an acyclic undirected graph (unrooted tree) \( T \) such that there is a one-to-one correspondence between the vertices of the tree and the subsets, and such that for every element \( x \in X \) all the nodes in \( T \) corresponding to the subsets that contain \( x \) induce a subtree of \( T \).

The subset collection is tree-like if it is possible to create edges between the subsets so that for any element \( x \in X \), the induced subgraph on the subsets that contain \( x \) is a tree. The results of Tarjan and Yannakakis [43] show that it is possible to test whether a subset collection is tree like, and if so to construct a subset-tree for it, in linear time.

Since ECM is equivalent to WSC, and since Guo and Niedermeier [25] provide a polynomial algorithm for WSC where the subset collection is tree-like and each subset contains at most \( b \) elements (for a constant \( b \)), we obtain the following corollary:

**Corollary 1.** Given a STSG where each agent has at most \( b \) skills and the collection of the agents’ skill subsets is tree-like, it is possible to solve ECM in polynomial time.

---

\(^{11}\)To see this equivalence, consider each vertex to be an agent, and each edge to be a skill connecting the two agents that share the skill.

\(^{12}\)The algorithm in [25] is fixed parameter tractable. Fixed parameter tractable algorithms with parameter \( k \) run on input \( I \) of size \( |I| \) in time \( f(k) \cdot |I|^{O(1)} \), where \( f \) is any computable function (typically exponential). In the case of [25] the parameter \( k \) is the size of the largest subset in the input.
For solving ECM in the other CSG classes, we need yet another restriction: the number of tasks must also be bounded by a constant $q$.

**Definition 28** (Bounded tree-like CSG domain). We say a CSG domain is a bounded tree-like CSG domain if each of the agents’ skill subsets has at most $b$ skills (where $b$ is a constant), where there are at most $q$ tasks (where $q$ is a constant), and where the agents’ skill subsets are tree-like.

There is no limitation on the total number of skills or the total number of agents. We now provide the equivalent result to Corollary 1 for the other CSG classes.

**Theorem 9.** In bounded tree-like CSGs domains of the classes STSG, TCSG, WTSG, TCSG-T, and WTSG-T it is possible to solve ECM in polynomial time.

**Proof.** We say a task subset $T' \subseteq T$ is achievable by the coalition $C$ if $C$ covers the set of skills $S(T') = \cup_{t' \in T'} S(t_j)$. If $T'$ is achievable by coalition $C$, then we have $T' \subseteq T(C)$ where $T(C)$ is the set of all tasks the coalition $C$ can achieve. We have assumed free disposal of tasks, so $u(T(C)) \geq u(T')$ and the value of a coalition $C$ is $v(C) = u(T(C)) \geq u(T')$. Thus, if for any coalition we have $p(C) \leq u(T') - \epsilon$ then $C$ has a deficit $\epsilon(C) = v(C) - p(C) \geq \epsilon$ and $p(C)$ is not in the $\epsilon$-core. On the other hand, $T(C) \subseteq T$, so if for any $T' \subseteq T$ we have $p(C) > u(T') - \epsilon$, then all coalitions have a deficit of at most $\epsilon$, and the imputation $p$ is in the $\epsilon$-core. Thus, to check if the maximal deficit under an imputation is at most $\epsilon$ (or in other words, to solve ECM), it suffices to test every task subset $T' \subseteq T$ and test whether any coalition $C$ that achieves $T'$ has a payoff $p(C) > u(T') - \epsilon$.

We now note that in bounded CSG domains, there is a constant number of tasks, so there is a constant number of task subsets. Thus we must only examine a constant number of task subsets $T'$ and test whether any coalition $C$ that achieves $T'$ has a payoff $p(C) > u(T') - \epsilon$. Such a test for a specific task subset $T'$ requires polynomial time. In all the above CSG domains, it is possible to compute $u(T')$ in polynomial time. Also, to achieve $T'$, a coalition must cover $S(T')$. Since the domain is bounded and tree-like, the method of Guo and Niedermeier [25] allows testing the minimal payoff $p(C)$ of any coalition that covers $S(T')$: similarly to the proof of Theorem 8, this is simply a Weighted Set-Cover problem and can be solved in polynomial time in tree-like domains where the agents’ skill subsets have at most $b$ skills. \hfill \Box

We now show that in these restricted domains, the more demanding problems of finding $\epsilon$-core imputations and computing the least-core are also in P. We first show that ECF can be solved in polynomial time in these restricted domains, using a separation oracle.

**Theorem 10.** In bounded tree-like CSG domains of the classes STSG, TCSG, WTSG, TCSG-T, and WTSG-T, ECF is solvable in polynomial time.
**Proof.** We first consider an exponential-size feasibility linear program for computing an imputation in the \(\epsilon\)-core. The program simply considers all the possible 2\(^n\) coalitions over the \(n\) players, \(C_1, \ldots, C_{2^n}\). The \(\epsilon\)-core can be written directly as a linear program over the variables \(p_1, \ldots, p_n\) (representing the agents’ payoffs in the imputation), with a constraint for each such coalition.

<table>
<thead>
<tr>
<th>Feasible ((p_1, \ldots, p_n))</th>
<th>s.t.:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. (v(C_1) - \sum_{i</td>
<td>a_i \in C_1} p_i &lt; \epsilon)</td>
</tr>
<tr>
<td>1.2. (v(C_2) - \sum_{i</td>
<td>a_i \in C_2} p_i &lt; \epsilon)</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>1.2(^n). (v(C_{2^n}) - \sum_{i</td>
<td>a_i \in C_{2^n}} p_i &lt; \epsilon)</td>
</tr>
<tr>
<td>2. (\sum_{i=1}^n p_i = v(I))</td>
<td>(Imputation constraint)</td>
</tr>
</tbody>
</table>

The above program is a feasibility linear program, and any solution for it, \(p = (p_1, \ldots, p_n)\), is an imputation in the \(\epsilon\)-core. However, its size is exponential in the number of agents, so even writing this program requires exponential time. Nonetheless, we can use a separation oracle to either find an \(\epsilon\)-core imputation or show the \(\epsilon\)-core is empty.\(^{13}\) Our proofs of Corollary 1 and Theorem 9 used the relation between ECM and Weighted Set-Cover, and taking a candidate imputation \(p = (p_1, \ldots, p_n)\) as input, find a coalition \(C\) with payoff \(p(C)\) that achieves a task subset \(T'\) such that \(p(C) \leq u(T') - \epsilon\), if such a subset exists. Such coalitions are exactly the ones for which the constraint is violated in the above linear program, and provide a separation oracle. Given this separation oracle, the above linear program can be solved in polynomial time and without the need to specify it explicitly.

Theorem 10 shows that under the above restrictions regarding the CSG domain, given a certain \(\epsilon\) it is possible to find an imputation in the \(\epsilon\)-core in polynomial time if one exists, or determine that the \(\epsilon\)-core is empty. We now consider the problem of finding the least-core.

**Corollary 2.** In bounded tree-like CSG domains of the classes STSG, TCSG, WTSG, TCSG-T, and WTSG-T, LCV is solvable in polynomial time, and it is also possible to find an imputation in the least-core in polynomial time.\(^{14}\)

**Proof.** First note that the maximal value any coalition can achieve is \(v(I)\), so the \(v(I)\)-core is always non-empty. We can perform a binary search on the minimal value of \(\epsilon\) such that the \(\epsilon\)-core is non-empty, applying the algorithm of Theorem 10 on each tested value. \(\Box\)

\(^{13}\)A separation oracle is an algorithm that, given a candidate feasible solution either returns a violated constraint, or confirms that the solution is feasible. A linear program can be solved in polynomial time by the ellipsoid method as long as it has a polynomial-time separation oracle, and it is not necessary to explicitly write down the program [37, 24].

\(^{14}\)The least-core value is computed up to a desired degree of accuracy, so the methods here compute \(\epsilon'_{min}\) such that the distance between the true \(\epsilon_{min}\) value and this value is at most \(|\epsilon_{min} - \epsilon'_{min}| < \delta\), and the running time is polynomial in the accuracy (or logarithmic in the number of bits to represent it). The imputation found is in the \(\epsilon'_{min}\)-core.
Thus, although Theorem 8 shows that in general (non-tree-like) CSGs, even testing for an \( \epsilon \)-core imputation is hard, the key questions regarding the least-core and \( \epsilon \)-core are tractably solvable for more restricted domains.

### 3.6 The Shapley Value and Banzhaf Power Index

We now consider calculating the Shapley value and Banzhaf power index in CSGs. Dummy players have a Shapley value and Banzhaf index of 0. Thus, computing either index allows testing for whether an agent is a dummy player.

**Corollary 3.** SHAPLEY and BANZHAF 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 29.** \#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 \( \bigcup_{S_i \in C'} \supseteq S \). Given \( S \) and \( C \), we are asked to compute the number of different set covers of \( S \).

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

**Definition 30.** \#SET-COVER-K (\#SC-K): A set-cover with size \( k \) is a set cover \( C' \) such that \( |C'| = k \). 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 \( G = \langle V, E \rangle \) the complement graph \( G^c = \langle V, E' \rangle \) is the graph where \( (u, v) \in E', u \neq v \), if and only if \( (u, v) \notin E \). Three problems related to \#SC are the following:

**Definition 31.** \#VC: We are given an undirected graph \( G = \langle V, E \rangle \), and are asked to count the number of vertex covers in the graph. A vertex cover is a subset \( V' \subseteq V \) such that for every edge \( e = (u, v) \in E \), either \( u \in V' \) or \( v \in V' \).
Definition 32. \#CLIQUE: We are given an undirected graph \( G = \langle V, E \rangle \), and are asked to count the number of cliques in the graph.

Definition 33. \#CLIQUE-K: We are given an undirected graph \( G = \langle V, E \rangle \), and are asked to count the number of cliques with size at least \( k \) in the graph.

Problems \#SC-K and \#CLIQUE-K are known \#P-complete problems [22].

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 11. \#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 \( G = \langle V, E \rangle \) 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 \( G_2 = \langle V_1, V_2, E_2 \rangle \) 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 \( G_3 = \langle V_1, V_2, V_3, E_3 \rangle \), 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 \( 1 \leq x, y, z \leq 3 \), 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.

As explained in [22], 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.
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 $m! \cdot \sum_{i=1}^{m} \binom{m}{i} c_i$, so an algorithm for #CLIQUE returns $\sum_{i=1}^{m} \binom{m}{i} c_i$. 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 12.** 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. 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 $S_{n+1} = \{s_{\text{new}}\}$, such that $s_{\text{new}} \notin S$. The BANZHAF query is regarding the Banzhaf index of $a_{n+1}$. Every winning coalition must cover $s_{\text{new}}$, 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 $\frac{x}{2}$, then the #SC answer is $x$. Thus a polynomial algorithm for BANZHAF also solves #SC, so BANZHAF is #P-complete.

**4 Similar Models of Cooperative Games**

Related research deals with similar models of cooperation among agents. A model similar to CSGs was used in Yokoo et al. [46], where the concept of anonymity and false-name manipulations was presented. Another similar model
is *Coalitional Resource Games* (CRGs) [45], a restricted form of Qualitative Coalitional Games (QCGs) [44]. We now consider similarities and differences between our model and these others.

### 4.1 Anonymous Proof Solutions

Yokoo et al. [46] consider 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 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 $\forall a_i \neq a_j, \ S_i \cap S_j = \emptyset$.

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 [46] is essentially equivalent to that of general CSGs. Obviously, the model of [46] is more general, as it directly maps a subset of skills that a coalition has to the utility of that coalition, so any mapping from a skill subset to a utility is possible, whereas our CSG model requires defining the utility through tasks. However, any game represented in this model of [46] 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, our task-based CSG representation can save much space, since if a certain skill subset $S' \subset S$ allows achieving a task, the CSG model does not have to specify the utility for any skill subset $X$ such that $S' \subset X$, and this is assumed to be at least the utility of $S'$ (so a new value must only be specified if $S''$ allows achieving additional tasks).

Moreover, Yokoo et al. [46] do not consider the computational complexity of calculating solution concepts. Rather, these authors consider strategic issues; for example, they show 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.

### 4.2 Coalitional Resource Games

Wooldridge and Dunne [45] present a model of *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. One main concern that these authors address is the properties of goal subsets that are successful—both feasible and satisfying for a coalition. Wooldridge and Dunne [45] consider 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.

Formally, such games have set $I = \{a_1, \ldots, a_n\}$ of agents, a set $G = \{g_1, \ldots, g_m\}$ of possible goals, a set $R = \{r_1, \ldots, r_t\}$ of resources, an endowment function $en : I \times R \to \mathbb{N}$ mapping an agent and a resource to the amount of that resource that the agent has, and a requirement function $req : G \times R \to \mathbb{N}$ mapping a goal and a resource to the amount of that resource required to obtain that goal. The amount of resource $r_j$ available to a coalition $C$ of agents is $en(C, r_j) = \sum_{a_i \in C} en(a_i, r_j)$. Similarly, the amount of $r_j$ required for a set of goals $G'$ is $req(G', r_j) = \sum_{g_k \in G'} req(g_k, r_j)$.

A goal subset $G' \subset G$ satisfies a coalition $C \subset I$ if for every agent $a_i \in C$ there is a goal $g \in G'$ such that $g \in G_i$, and the set of goal sets that satisfy a coalition $C$ is denoted by $sat(C) = \{G' \subset G | \forall a_i \in C, G_i \cap G' \neq \emptyset\}$. A set of goals $G'$ is feasible for coalition $C \subset I$ if that coalition has sufficient resources to achieve all the goals in $G'$. We denote the set of feasible goal sets for coalition $C$ by $feas(C) = \{G' \subset G | \forall r_j \in R, req(G', r_j) \leq en(C, r_j)\}$.

Almost all the results are negative results, so although CRGs have small representations, answering many questions regarding them is computationally hard. One positive result is that of potential goal sets—a polynomial algorithm is presented that, given a goal set $G' \subseteq G$, decides whether there is some coalition $C \subseteq I$ that $G'$ satisfies and for which $G'$ is also feasible.

Our model of CSGs is somewhat similar to that of CRGs. CSGs define tasks to accomplish, and CRGs define 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.

First, completing a goal in a CRG requires different amounts of various resources. In a CSG, tasks simply require the use of a skill, and the only requirement is for an agent in the coalition to have the skill (the skill is not consumed when performing the task). As mentioned in [45], it is possible to model this situation in CRGs by requiring just 1 unit of the resource for any goal, giving an agent $2^{|I|}$ units of that resource, enough for any possible coalition. In this sense, CRGs are more general than CSGs. However, this expressiveness of CRGs comes at a price, since many questions regarding CRGs are computationally hard. For example, while computing the value of a coalition in a CSG can be done in polynomial time, testing whether there exists a feasible and satisfying goal set (a question that somewhat resembles computing the value of a coalition) is NP-hard in CRGs; many questions relating to agent properties are hard in CRGs while finding veto agents in CSGs admits a simple polynomial algorithm.

Second, the CRG model does not define a coalitional game, but rather defines for each coalition $C$ the successful goal sets for that coalition, $sf(C) \subset G$, which are both feasible and satisfying. A solution in the CRG 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. It is possible to
define a simple coalitional game, with the characteristic function defined to be 1 for coalitions that have successful goal sets and 0 for coalitions that do not, so that

\[ v_{CRG}(C) = \begin{cases} 1 & \text{if } \exists G' \subseteq G \text{ that } G' \in sf(C) \\ 0 & \text{otherwise} \end{cases} \]

However, such a definition has several drawbacks. Testing if a certain coalition has a successful goal set is NP-complete, so simply evaluating the value of the game is computationally hard, even for a single coalition. More importantly, if a goal satisfies a coalition, all the agents are satisfied as at least one of their goals is achieved. This makes the question of dividing the total utility among the agents meaningless.

In comparison, the model of CSG assumes that accomplishing different tasks generates a certain utility, which has to be divided among all the agents. An example of this situation is a common project that requires completing various tasks, that all the agents have to accomplish, and that generate a certain revenue. The agents are not simply satisfied that the project has succeeded, but rather their utility is the share of the revenue received.

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 the computational complexity of some key problems related to game theoretic solution concepts in these domains. We showed that although in all the restricted CSG forms we defined we could calculate the value of a coalition in polynomial time, some problems remain computationally hard. Some key results we have shown are tractable algorithms for testing veto agents and dummy agents (in most domains) and computing the core or testing its emptiness (in most domains), while showing that computing power indices is computationally hard.

Several questions remain open for future research. First, this paper presents several negative results, especially regarding the \(\epsilon\)-core and least-core in general domains and regarding computing power indices. Power indices, however, can be approximated using general algorithms, as discussed in Section 1.2. On the other hand, this work provided positive results regarding \(\epsilon\)-core and least-core related problems in CSGs only in quite restricted domains. It would be interesting to see if there are other restricted domains where such problems can be tackled. Second, there are game theoretic solution concepts refining the least-core, such as the nucleolus, and a further step would be to examine the complexity of computing these solution concepts. Third, it is still open to decide whether core-non-emptiness is co-NP-complete for TCSSGs and WTSGs, and whether the Shapley value is in NP. Finally, it would be interesting to see how CSG-based models could be used in real-world applications.
6 Acknowledgment

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

References


