Computing cooperative solution concepts in coalitional skill games

Computing 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.[1]

Keywords: Coalitional game theory, Core, Power Indices

1. Introduction

Game theory has uses in many real-world domains, including those involving automated agents. These domains encompass electronic commerce, auctions, and general resource allocation scenarios. One way to analyze these domains is to model them as multi-agent systems, consisting of self-interested agents, with possibly conflicting preferences. Because of the desire to embed game-theoretic principles into artificial multi-agent systems, computational

[1]A preliminary version of this paper [11] was presented at the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008).

aspects of game theory and social choice have been extensively studied in recent years.

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 [40], the ϵ-core and least-core [62], the Shapley value [60], and the nucleolus [58]. 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 [14] and Shapley-Shubik [61] 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 game theory have been adopted by computer scientists, who have explored their attendant computational considerations [56, 69, 63, 22, 23, 28, 24]. Specifically, much work has been done on computational aspects of agent collaboration through cooperative game theory [19]. 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*, that 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 modeled using our class of games.

Our first example is that of several companies that 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 worth \$500 million, and the third is located in a remote desert and 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, and is a pure oil patch that requires no refining, and 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.

How should this total reward be split among 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? 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. 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, and so D certainly helps in achieving the total gain. B might suggest allocating \$400 million to A, \$150 million to B and C, and \$410 million to D. But companies A and D are unlikely to find this satisfying, as they can achieve \$910 million on their own. Are there stable payoff allocations in this domain?

Another example is *voting*. Consider a vote in which different entities control how different subsets of the voters choose to vote, and where a coalition of entities can control the votes of all the voters that can be controlled by any member of the coalition. A coalition may have to control more than a certain number of voters to win an election, and thus its utility can depend on the number of voters it controls. A similar example is *cooperative knowledge sharing* [31, 38]. In such domains, each party may have access to information regarding various propositional variables, and a coalition wins if it can determine the value of a particular subset of the variables that are of interest.

Yet another example is the domain of robots and human rescuers in a disaster zone with victims trapped in multiple locations, where each location requires various skills in order to gain access [46, 57]. For example, in the case of a fire, rescuing a trapped victim may require a robot for an initial visual inspection, 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 final example focuses on the domain of *multi-sensor networks* [64, 27]. Consider multiple sensor-array facilities, each with several sensors of different types, where each sensor covers a certain geographical area. The goal of the sensor 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 sensor arrays can track an object all the time, 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?

1.2. 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.

CSGs are highly expressive. For example, CSGs are able to easily express the domains presented in Section 1.1. A task in these domains can include producing oil, rescuing a victim, tracking an object, transmitting data to a certain client, controlling a certain voter, and so forth. In each of these domains, completing a task depends on having the necessary skills. Questions of interest in these domains include how to divide the total gains of a coalition among its members, and finding which members of the coalition are powerful.

1.2.1. Restricted Classes of CSGs and Threshold Versions of CSGs

The expressive power of general CSGs comes at the cost of having a representation that is exponential in the number of tasks, which results in high computational complexity in solving various questions related to such games. To achieve a good compromise between expressiveness and conciseness we can impose additional, natural restrictions on the structure of the underlying task allocation problems, and therefore on the characteristic function of the coalitional 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. TCSGs easily allow one to express the fact that the utility of a coalition of agents depends on the number of tasks that they can jointly perform, while WTSG allows for weighting such tasks according to their importance.

Any STSG can be expressed as a TCSG (with a single task), and any TCSG can be expressed as a WTSG (with equal weights for the tasks). We can denote this as STSG $\subseteq$ TCSG $\subseteq$ WTSG. We explore the issue of the expressivity of these representation language in Section 3.

We also consider *threshold* versions of skill games, that capture the notion of "acceptable performance," where there is a discontinuity in utility beyond a certain performance level. Such discontinuities are very common in many real-world scenarios. For example, agents may be under a *contract* to perform at least a certain number of tasks or to achieve a certain target in sales or profits, as is the case with many service level agreements and employment contracts with sales quotas [21, 49, 26]. Another example is voting and lobbying effort. In many voting situations a decision is made based on the opinion of the majority of voters, so in such cases passing a decision requires convincing at least half of the decision makers. In this case, no utility is gained by convincing less than the required quota of the decision makers. Similarly, once half of the decision makers are convinced, no additional utility is gained by convincing additional decision makers. This has typically been modeled as weighted threshold games (sometimes called weighted voting games) or as lobbying games [33, 14, 15, 53].

Both TCSGs and WTSGs also have simple threshold versions, where a coalition of agents either "wins" or "loses." These CSGs 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 skill games (*STSG*s), where a single task has to be performed, and requires a coalition to cover all the required skills for that task. These threshold versions can easily express lobbying efforts where agents must complete various tasks to persuade decision makers, or where they must complete different tasks in order to sell products to meet a sales quota.

Our threshold versions of CSGs are somewhat similar in spirit to threshold versions of other cooperative games [33, 12, 2, 5], which are discussed in Section 1.3 and Section 5.

1.2.2. Main Results

We first show that even very restricted forms of CSGs are still very expressive. One of the most general classes of cooperative games is that of increasing games, where the value generated is monotonically increasing in the set of agents in the coalition. Even the most restrictive class of STSGs can represent any increasing game where a coalition either wins or loses, and WTSGs can represent any increasing game. We then explore the 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. The problem of computing the core of a CSG, and testing whether the core is non-empty, is central to determining the basic stability properties of a CSG. If the core exists, then it is possible to distribute the gains in a *stable* way, so that no subset of the agents will prefer to break away. If the core is empty, then any division of the payoffs is unstable and it is typical to consider relaxed solutions such as the *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.

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 (also called *imputations*) in the core. Problems related to the least core are generally hard, but we present positive results for restricted domains. With regard to power indices, we show that they are typically hard to compute precisely. However, these can be approximated by sampling player permutations or coalitions and averaging the marginal contributions of the target agent [8]. Thus, this work shows that the CSG representation is expressive, and computationally tractable in restricted forms.

1.3. Related Work

The concept of the core originated with Gillies [40], the ϵ-core and least-core were introduced by Shapley and Shubik [62], and the Shapley value was introduced by Shapley [60]. The problems of finding the optimal coalition structure and determining whether a coalition structure is stable have been studied in a number of domains [63, 27], including the CSG domain [9]. Values similar to the Shapley value have been used to measure power, e.g., via the Shapley-Shubik [61] and Banzhaf [14] power indices. Deng and Papadimitriou [30] 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 [50]. Several papers deal with computing, comparing, and approximating power indices, in general and in restricted domains [51, 30, 24, 36, 8].

Conitzer and Sandholm [24] use a decomposition of a coalitional game to several domains to ease the calculation of the Shapley value. While this decomposition technique makes the computation of the Shapley value easier, the same technique cannot be directly used for the Banzhaf index, which we consider in this paper. Also, while our representation employs a decomposition technique via tasks, the success of such tasks depends on a set of skills, rather than on 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.

There has been significant research on the trade-offs between expressiveness and computation for cooperative games. We examine the relation between the CSG model and other representation languages for cooperative games in Section 5, after discussing our technical results. We compare with linear production games [52], where agents pool together resources in order to manufacture finished goods that can be sold at a given market price. The utility of a coalition in this model is its maximal revenue, achieved when manufacturing the optimal achievable bundle of finished goods, as given by the solution to a certain linear program. Also closely related is the *Coalitional Resource Games* model [67], in which agents wish to achieve various goals, and are endowed with certain amounts of resources required to achieve these goals. Wooldridge and Dunne [67] examined the reasonable outcomes of agent interaction in such settings. Also similar is a representation adopted by Yokoo et al. [68] for anonymous environments where an agent may use false names and distribute its ability among these identities.[2]

Additional work on representation includes Bilbao [16], who studies cooperative games on several combinatorial structures. The *marginal contribution network* language [44] represents the value of a coalition using a set of rules regarding the participating agents. Another model, cooperative graph games, 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 [30]. The *sparse synergy* representation [23] relies on super-additivity, and is concise when the number of synergies between coalitions is low. The sparse synergy representation allows for efficient checking of whether a given outcome is in the core, but determining whether the core is nonempty remains NP-complete. The *Multi-Attribute Coalitional Games* (MACG) [45] provides a representation of coalitional games where the value of a coalition is described by a set of agent attributes, along with functions that aggregate the attributes of agents to a single number. Other work has considered coalitional games played over networks, such as network flow games, connectivity games and spanning connectivity games [12, 3, 13]. Several papers have examined the Cost-of-Stability and other solution concepts in network domains [10, 7, 54, 2].

1.4. Outline

In Section 2 we give some background concerning coalitional games, and define the CSG model. Section 3 examines the expressiveness of the CSG representation of cooperative games. In Section 4 we present the main al-

[2]False-name attacks and similar perturbations that change the agent power distribution have also been studied in [6, 70, 4].

gorithms and complexity results of the paper. Section 5 examines closely-related models in more detail, and we conclude in Section 6.

2. Preliminaries

In this section, we define the CSG model and the 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** Γ *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* Γ *is clear from the context, we sometimes omit the* Γ *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 Γ is *increasing* if for all coalitions $C' \subset C \subseteq I$ we have $v_\Gamma(C') \leq v_\Gamma(C)$, and is *super-additive* when for all *disjoint* coalitions $A, B \subset I$ we have $v_\Gamma(A) + v_\Gamma(B) \leq v_\Gamma(A \cup B)$. In some cases increasing games are referred to as *monotone* games. 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 Γ, v_Γ 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")* $(p_1, \dots, p_n)$ *is a division of the gains of the grand coalition among the agents, where* $p_i \in \mathbb{R}$, *such that* $\sum_{i=1}^{n} p_i = v(I)$.[3]

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 is of course how to determine which imputations are likely to be chosen by agents sharing their joint gains, or to characterize imputations with desirable properties such as fairness. Game theory offers several answers to this issue.

2.1.1. Individual Rationality, the Core, ϵ-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 C blocks the imputation $(p_1, \dots, p_n)$ if $p(C) < v(C)$, since the members of C can split from the original coalition, derive the gains of $v(C)$ in the game, give each member $a_i \in C$ 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* $p = (p_1, \dots, p_n)$*, 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.*

Definition 4. *A coalition* C **blocks** *the imputation* $p = (p_1, \dots, p_n)$ *if its excess under this imputation is strictly positive,* $e(C) > 0$*. If a coalition* C *blocks the imputation* p*, we say that* C *is* **unstable** *under* p*.*

A prominent solution concept focusing on stability is that of the core [40].

Definition 5. *The* **core** *of a coalitional game* Γ *is the set of all imputations* $(p_1, \dots, p_n)$ *that are not blocked by any coalition, so that for any coalition* C*, we have* $p(C) \geq v_\Gamma(C)$*.*

[3]Some existing literature refers to a payoff vector $p = (p_1, \dots, p_n)$ as a *pre-imputation* if $\sum_{i=1}^{n} p_i = v(I)$, and only considers it an *imputation* if for any agent $i \in I$ we have $p_i \geq v(\{i\})$. We use the simplified terminology, and call any payoff vector where $\sum_{i=1}^{n} p_i = v(I)$ an imputation.

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 choose to remain. A relaxed solution concept that follows this rationale is the ϵ-core [62]:

Definition 6. *The* **ϵ-core** *is the set of all imputations* $(p_1, \ldots, p_n)$ *such that the following holds: for any coalition* $C \subseteq I$, $p(C) \geq v_\Gamma(C) - \epsilon$.

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

Definition 7. *The* **least-core** *of the game* Γ *is the* ϵ_{min}*-core of* Γ.

The imputations in the least-core distribute the gains in a way that minimizes the worst-case deficit, or in other words, minimizes 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. The least-core does not, for example, minimize the *number* of coalitions with this worst-case deficit, nor does it minimize the second-worst deficit. The *nucleolus* [58] refines the least core, selecting the lexicographically minimal core.

2.1.2. The Shapley Value and Banzhaf Power Index

Another solution concept, which defines a *single* imputation, is the Shapley value [60]. This approach focuses on *fairness* rather than on stability. This value is the only imputation (payoff distribution) that fulfills certain fairness axioms [60]. [4] We denote by π a permutation of the agents, so

[4]Shapley discussed three axioms [60]. These axioms roughly state that: dummy agents who contribute nothing to all coalitions should get a payoff of zero; Equivalent agents,

$\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and π is onto, 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 8. *The* **Shapley value** *of a game* Γ *(with characteristic function* v_Γ*) is given by the imputation* $\phi(v_\Gamma) = (\phi_1(v_\Gamma), \ldots, \phi_n(v_\Gamma))$ *where*

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

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 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 $\phi_i(v)$ 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 [14]. The Banzhaf index depends on the number of coalitions in which an agent is critical, out of all possible coalitions.

Definition 9. *The* **Banzhaf power index** *of a game* Γ *(with characteristic function* v_Γ*) is given by* $\beta(v_\Gamma) = (\beta_1(v_\Gamma), \ldots, \beta_n(v_\Gamma))$ *where*

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

who contribute the same amount to any coalition containing neither of them, should get an equal payoff; The payoff of any agent in a composed game, where the value of any coalition is the sum of the values of that coalition in two composing games, should be the sum of payoffs of that agent in the two composing games.

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 $S(a_i) \subseteq S$,[5] 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 $S(t_j) \subseteq S(C)$. We denote the set of tasks a coalition C can perform as $T(C)$. The set $T(C)$ consists of all the tasks $t_j \in T$ such that C can perform t_j.

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: $u : 2^T \to \mathbb{R}$. 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 $T1 \subset T2 \subseteq T$, we have $u(T1) \leq u(T2)$. The free disposal of tasks has important implications, as it means a coalition can never lose utility by being able to perform more tasks. Having the skills required to perform a certain task does not mean the coalition must perform the task. Rather, the coalition may decide not to make use of these skills.

An alternative definition may associate some tasks with a negative utility, but our assumption throughout this work is that the utility gained by a coalition is monotone in the set of tasks the coalition can achieve.[6]

We define the coalitional skill game for a coalitional skill domain as follows:

Definition 10 (CSG)**.** *A* **CSG** *is the coalitional game* Γ *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.*

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

[6]Such an alternative definition of skill games which allows negative weights to be associated with tasks is examined in [9], which analyzes the complexity of finding optimal coalition structure in skill 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. $\square$

2.3. Restricted Forms of CSGs

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. In order to make progress we consider restricted forms of CSGs, that remain expressive (e.g., allowing use to represent any monotone characteristic function) while allowing for computational analysis.

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 in a TCSG simply contains a list of the tasks and a list of required skills for each task.

Definition 11 (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 concise 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 12 (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 $w_j \in \mathbb{R}_+$.*[7] *The task value function $u(T') = \sum_{j \in \{j | t_j \in T'\}} w_j$.*

CSGs where the task value function u can only obtain the values 0 and 1, 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 $v_\Gamma(C) = u(T(C))$, in simple skill games v_Γ's range is also $\{0, 1\}$ and we have $v_\Gamma : 2^I \to \{0, 1\}$.

[7]In 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. In the case where negative weights are allowed, the free disposal of tasks becomes more controversial.

Both TCSG and WTSG have special cases 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 13 (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 $v_\Gamma(C) = 1$ if $|T(C)| \geq k$ and $v_\Gamma(C) = 0$ otherwise.*

Definition 14 (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 function u is defined as $u(T') = 1$ if $w(T') > k$ and $u(T') = 0$ otherwise. Thus, the game has the characteristic function $v_\Gamma(C) = 1$ if $w(T(C)) \geq k$ and $v_\Gamma(C) = 0$ 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 $v_\Gamma(C) = 1$ 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 15 (STSG). *An* **STSG** *is a TCSG where there is only a single task t, so $v_\Gamma(C) = 1$ if $S(C) = S$ and $v_\Gamma(C) = 0$ otherwise.*

Example 1. *Consider a STSG, in a domain with two skills: $S = \{s_1, s_2\}$, the single task t_1 which requires both skills (i.e. $S(t_1) = \{s_1, s_2\}$), and three agents, $I = \{a_1, a_2, a_3\}$. Agent a_1 has the skill $S(a_1) = \{s_1\}$, and agents a_2, a_3 have the skill $S(a_2) = S(a_3) = \{s_2\}$. Since in order to complete the task a coalition needs both skills, a winning coalition must contain a_1 and one of the two agents a_2, a_3. Thus, the winnings coalitions are $\{a_1, a_2\}, \{a_1, a_3\}$ and $\{a_1, a_2, a_3\}$.*

It is easy to verify that the imputation $p = (1, 0, 0)$ is a core imputation. On the one hand, any coalition C such that $a_1 \notin C$ is losing and cannot block this imputation; on the other hand any coalition C' such that $a_1 \in C'$ has a payoff of $p(C') \geq p_1 = 1$ and cannot block the imputation. Since p is not blocked by any coalition, it is in the core. Similarly, a direct computation using the formula of Definition 8 shows that the Shapley value is $\phi = (\frac{2}{3}, \frac{1}{6}, \frac{1}{6})$.

All these restricted forms of skill games have concise representations, since we can find a succinct representation for the task value function, and thus also have a succinct representation for the characteristic function.

These restricted representations can also model many situations. For example, TCSGs can express 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-Ts can express the fact that to be successful, we must track at least a certain number of objects in the cooperative sensor array domain.

Some questions remain computationally hard even with these restrictions. For example, certain problems regarding power indices and the ϵ-core and least-core are generally hard even for STSGs. Below, we either suggest approximation algorithms (for computing power indices) or focus on restricted domains (for least-core related problems).

3. Expressiveness of Coalitional Skill Games

In this section we explore the ability of CSGs to express various characteristic functions.

Lemma 1 has shown that CSGs are increasing games, so obviously CSGs cannot represent non-monotone characteristic functions. We show that the CSG representation is *fully expressive* for increasing games: STSGs can represent any simple increasing cooperative game, and WTSGs can represent any increasing cooperative game.[8]

Theorem 1. *STSGs are fully expressive for the class of simple increasing games: it is possible to represent any simple increasing game as an STSG.*

Proof. Consider a given simple increasing cooperative game over the set I of players and with the characteristic function $v : 2^I \to \{0, 1\}$. Given this

[8]Related work on with coalition structure generation in skill games provides a proof that WTSGs can express any cooperative game [9]. However, it examines *non-monotone* skill games which do not allow the free disposal of tasks, whereas in this work we do make the assumption of free disposal of tasks. The CSG model examined by Bachrach et al. [9] allows tasks to be associated with negative weights, so performing an additional task can diminish a coalition's utility. Our results are in regard to the monotone model, which respects the free-disposal assumption, and are not entailed by the results in Bachrach et al. [9].

original game, we construct an STSG representation for it as follows. For any *losing* coalition $L \subseteq I$ in the original game, we create a skill s_L. This skill is possessed by $I \setminus L$, i.e., all the agents who are *not* in L, and by no other agents. The single task t in the constructed game requires all the skills, so this is a proper STSG. First note that all the agents in L do not possess the skill s_L (as by construction this skill is only possessed by agents who are not in L), so L cannot achieve the task t which requires s_L. On the other hand, consider a winning coalition $W \subseteq I$ in the original game. The original game is increasing, so since W is winning in that game, it *cannot* be the case that W is *contained* in any losing coalition L. Thus for any losing coalition L, there is an agent $j \in W$ such that $j \notin L$. By our construction, since $j \notin L$, player j is endowed with the skill s_L. Thus, for any skill s_L the coalition W covers s_L, so W covers all the skills and is a winning coalition. Therefore, any losing coalition in the original game is also losing in the constructed game, and any winning coalition in the original game is also winning in the constructed game, so both games have the same characteristic function. □

Theorem 1 shows that STSGs can represent any simple monotone game. We now show that WTSGs can represent any monotone game (not necessarily a simple game). Our method is based on game composition. Given two games over the same set I of agents, v_1 and v_2 the composed game $(v_1 + v_2)$ over the agents I is defined as follows. For any coalition $C \subseteq I$ we have $(v_1 + v_2)(C) = v_1(C) + v_2(C)$.

Lemma 2. *If v_1 and v_2 have a WTSG representation, then there is a WTSG representation for $(v_1 + v_2)$.*

Proof. Let I be the set of agents in v_1 and v_2. Let S_1, S_2 be the skills of the agents in v_1 and v_2, accordingly. Let T_1, T_2 be the set of tasks in v_1, v_2 accordingly. Name the tasks and skills such that $S_1 \cap S_2 = \emptyset$ and that $T_1 \cap T_2 = \emptyset$. Construct a WTSG G over the set I of agents as follows. The set of tasks in G is $T = T_1 \cup T_2$. The set of skills is $S = S_1 \cup S_2$. Each task requires exactly the same skills it originally required in v_1 or v_2. Every agent $a_i \in I$ in the game G is endowed with all the skills she had in *both* v_1 and v_2. Denote G's function as v. If a coalition C achieves a task $t \in T_1$ in v_1, it can also achieve it in v, as the task requires the same skills it did in v_1, and if the agents of C covered them in v_1 they also cover them in v. Similarly, if C achieves a task $t \in T_2$ in v_2, it can also achieve it in v. However, if C does not achieve t in v_1 it would not achieve it in v

(the same skills are still missing). Similarly if C does not achieve t in v_2 it would not achieve it in v. Denote the set of tasks C can achieve in v_1 as W_1 and the tasks C can achieve in v_1 as W_2. The value C achieves in v is thus $v(C) = \sum_{t \in W_1} w(t) + \sum_{t \in W_2} w(t) = v_1(C) + v_2(C)$. Thus, given a WTSG representation for v_1 and v_2 we have built a WTSG representing $(v_1 + v_2)$. $\square$

We build a WTSG representation for a game by iterating through the possible values the game's function v can take, from lowest to highest. In each step we examine the coalitions that have the next highest value in v, and construct a WTSG where these coalitions get the difference between the values. We then take the WTSG representation of the composed game.

Theorem 2. *WTSGs are fully expressive for the class of general monotone games: it is possible to represent any monotone cooperative game as a WTSG.*

Proof. A game G over the n agents in I, with function v has 2^n different coalitions, so v has at most 2^n different values. Order these values from smallest to largest to obtain $u_0, \ldots, u_m$ (where $m < 2^n$). Denote the set of coalitions which have a value of u_i as C_{u_i} (so if $C \in C_{u_i}$ then $v(C) = u_i$). Denote $w_i = u_i - u_{i-1}$. Since v is monotone, u_0 is obtained by the empty coalition $\emptyset$, and w.l.o.g. we assume $u_0 = 0$.

Consider the game v'_1 over the agents I, where $v'_1(C) = 0$ if $v(C) = u_0$ and $v'_1(C) = 1$ if $v(C) > u_0$. This is a simple monotone game, so due to Theorem 1 it has an STSG representation G'_1. We can take this representation and modify the weight of the single task from 1 to w_1. This is a WTSG representation G_1 with characteristic function v_1 where $v_1(C) = v(C) = u_1$ if $C \in C_{u_0} \cup C_{u_1}$, and where if $C \in C_{u_i}$ for some $i > 1$ we have $v_1(C) = u_1$. Now consider the game v'_2 over the agents I, where $v'_2(C) = 0$ if $v(C) \leq u_1$ and $v'_2(C) = 1$ if $v(C) > u_1$. Again, this is a simple monotone game, and due to Theorem 1 has an STSG representation G'_2. Again we take this representation and modify the weight of the single task from 1 to w_2. This is a WTSG representation G_2, with characteristic function v_2. Due to Lemma 2 we can construct a WTSG representation H_2 for $(v_1 + v_2)$. Denote the characteristic function of H_2 as $h_2 = (v_1 + v_2)$. Note that if $C \in C_{u_0} \cup C_{u_1} \cup C_{u_2}$ we have $h_2(C) = v(C)$, and if $C \in C_{u_i}$ for some $i > 2$ we have $h_2(C) = u_2$. We can continue this process to construct a WTSG representation for H_3 then for H_4 and so on, until we obtain the game H_m with characteristic function v_m. However, $v_m(C) = v(C)$ for any coalition C, so this is a WTSG representation for G, as required. $\square$

Example 2. *We now provide an example for the construction in Theorem 2. Consider the game v over 3 agents $I = \{a_1, a_2, a_3\}$ with the characteristic function $v(\emptyset) = v(\{a_1\}) = v(\{a_2\}) = v(\{a_3\}) = 0$, $v(\{a_1, a_2\}) = v(\{a_1, a_3\}) = v(\{a_2, a_3\}) = 1$ and $v(\{a_1, a_2, a_3\}) = 5$. In this game no single agent makes any value, any two agents generate a value of 1, and the grand coalition of all agents has the value of 5.*

The construction of Theorem 2 generates a WTSG representation for the game v'_1 where any two agents have a value of 1, and the grand coalition has a value of 1 as well, and composes it with a representation for the game v'_2 where any coalition short of the grand coalition has a value of 0, and the grand coalition has a value of $5 - 1 = 4$.

The construction for the game v'_1 is done using Theorem 1. Since in the game v'_1 the losing *coalitions are the singleton coalitions $\{a_1\}, \{a_2\}, \{a_3\}$, the construction creates three skills: the skill s^1_1, owned by a_2 and a_3, the skill s^1_2, owned by a_1 and a_3, and the skill s^1_3, owned by a_1 and a_2. The task for the game v'_1 requires all these three skills, s^1_1, s^1_2, s^1_3.*

By generating a representation for v'_2, again using the construction of Theorem 1, and composing the representations for the games v'_1 and v'_2 using Lemma 2, we obtain a representation for the required game v.

Not every increasing cooperative game has a *succinct* representation as a CSG. For example, the construction in Theorem 1 involved generating a skill for every losing coalition, so the number of skills in this construction may be exponential in the number of players. However we believe that many domains, such as the examples in Section 1.1, can be described as CSGs in a concise way. In the reminder of the paper, we show that CSGs also have desirable computational characteristics.

4. Algorithms for Coalitional Skill Games

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 TCSG-T and WTSG-T) gives a representation that is always polynomial. We now define the specific computational problems of interest. All of these problems are stated with regard to a given CSG, Γ, and sometimes with regard to a target agent a_i.

Definition 16 (COALITION-VALUE (CV))**.** *Given a coalition $C \subseteq I$ and CSG Γ, compute $v_\Gamma(C)$.*

Definition 17 (VETO)**.** *In a simple CSG Γ, 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 18 (DUMMY)**.** *Check if a_i is a dummy player in CSG Γ, such that for every coalition C (with $a_i \notin C$), we have $v_\Gamma(C \cup \{a_i\}) = v(C)$.*

Definition 19 (CORE-NON-EMPTY (CNE))**.** *Decide whether the core of CSG Γ is non-empty.*

Definition 20 (CORE)**.** *Return a representation of all imputations in the core of CSG Γ.*

Definition 21 (ϵ-CORE-MEMBERSHIP(ECM))**.** *Given an imputation $p = (p_1, \dots, p_n)$, and CSG Γ, decide whether it is in the ϵ-core of the game.*

Definition 22 (SHAPLEY)**.** *Compute agent a_i's Shapley value $\phi_i(v_\Gamma)$ in CSG Γ.*

Definition 23 (BANZHAF)**.** *Compute agent a_i's Banzhaf power index $\beta_i(v_\Gamma)$ in CSG Γ.*

We summarize our results in Table 1.[9] Complexity results are stated in regard to the succinct representations that are achieved through the restricted forms of CSGs. Thus, our polynomial algorithms run in time that is polynomial in the number of agent tasks and skills. Because the core may sometimes contain infinitely many imputations, it is unclear how it should be represented. When no compact representation is known we enter "N/A" in the table. We have shown that testing whether the Shapley value exceeds a given threshold is an NP-hard problem. However, we have not established whether or not it is in NP. This remains open.

The most basic problem regarding the ϵ-core and least-core is ECM, which tests if an imputation is ϵ-stable. Table 1 shows that ECM is hard (co-NP-complete) even in the most restricted case of STSG. In domains where ECM

[9]Hardness of computing the Shapley value in STSGs, TCSGs and WTSGs is examined in Aziz et al. [3], where it is shown to be #P-complete, as a consequence of intractability results in the earlier version of the present paper in regard to the Banzhaf index.

	STSG	TCSG	WTSG	TCSG-T	WTSG-T
CV	P	P	P	P	P
VETO	P	P	P	P	P
DUMMY	P	P	P	co-NPC	co-NPC
CNE	P	co-NP	co-NP	P	P
ECM	co-NPC	co-NPC	co-NPC	co-NPC	co-NPC
CORE	P	N/A	N/A	P	P
SHAPLEY	#P-C	#P-C	#P-C	#P-C	#P-C
BANZHAF	#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; #P-C—#P-complete; N/A—depends on the core representation.

can be solved in polynomial time, one can *verify* that a certain imputation is in the ϵ-core, so it is "stable enough". In such domains, another interesting problem is *finding* such a "sufficiently stable" imputation (i.e., an imputation in the ϵ-core). Further, one may want to determine the best "degree of stability" that can be achieved. We formally define these problems:

Definition 24 (ϵ-CORE-FIND-IMPUTATION (ECF)). *Given ϵ, find an imputation $p = (p_1, \dots, p_n)$ in the ϵ-core of a CSG game Γ if one exists, and reply that no such imputation exists otherwise.*

Definition 25 (LEAST-CORE-VALUE (LCV)). *Compute the least-core value in CSG Γ, which is the minimal value ϵ_{min} such that the ϵ_{min}-core is non-empty.*[10]

We provide results regarding restricted domains where the ECM, ECF and LCV are tractable, so one can test for ϵ-core membership, find ϵ-core imputations and compute the least core value. 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

[10]A non-integer value can be represented up to a certain degree of accuracy, and so the LCV problem requires computing ϵ-min up to a certain accuracy. A tractable algorithm is polynomial in the input size and logarithmic in the desired accuracy (i.e., polynomial in the number of bits describing the accuracy level).

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. Unfortunately, we could not provide a polynomial algorithm for solving ECF and LCV in general CSGs, nor provide a hardness result. Thus, this remains an interesting open problem. Since ECM is hard in these domains, we conjecture that these problems are hard in general STSGs, TCSGs and WTSGs.

We now present the analysis of each problem in the various CSG domains. Since we focus on a given game Γ, from here on we drop the subscript Γ from our notation.

4.1. Coalition Value

Theorem 3. *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 $S(t_j) \subset S(C)$. 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). $\square$

4.2. Veto Agents

Theorem 4. *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 Γ 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 $C' \subset C$ by Lemma 1. If $v(C) > 0$ then a_i is not veto. As seen in Theorem 3, we can compute $v(C)$ in polynomial time. $\square$

4.3. Dummy Agents

We now consider testing whether an agent is a dummy. Due to Theorem 3, given a coalition C, we can compute $v(C \cup \{a_i\})$ and $v(C)$ in polynomial time, and see if $v(C \cup \{a_i\}) > v(C)$. Thus, given a specific coalition, we can check if a target agent contributes to that coalition. If this is the case for some coalition C, then our target agent is a non-dummy. Thus, DUMMY is *in* co-NP for all CSG classes we have defined.

We say that a coalition C *covers* a set of skills S if for any skill $s \in S$ there exists at least on agent in C who possesses the skill s. 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 3. *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, so there exists a coalition C such that $C \cup \{a_i\}$ is winning but C is losing. This only happens if C covers $S \setminus S_i$ and fails to cover 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 5. *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 3, a_i is a dummy player. □

Theorem 6. *DUMMY is in P for TCSGs and WTSGs.*

Proof. Let Γ be a WTSG, with tasks $t_1, \ldots, t_m$. Let Γ_j be the STSG with the single task t_j, with the same agents and skills as Γ. Suppose a_i is not a dummy in Γ, 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 Γ_j. Going the other way, 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 that in Γ we also have $v(C' \cup \{a_i\}) > v(C)$, and a_i 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, 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. This is easy to show for WTSG-T. Matsui and

Matsui have shown that testing if an agent is a dummy is hard in weighted voting games [50]. 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 this by reducing from 3SAT, a well-known NP-hard problem.

Definition 26 (3SAT)**.** *We are given a propositional formula in conjunctive normal form (CNF) over n propositional variables $y_1, \ldots, y_n$, 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 y_j or its negation $\neg y_j$). We are asked if there is an assignment that satisfies ψ.*

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 the 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).

Theorem 7. *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, and set the threshold to be $k = m$ tasks.

Given the 3SAT formula $\psi = c_1 \wedge c_2 \wedge \ldots \wedge c_m$ over n propositional variables $y_1, \ldots, y_n$ (where $c_i = l_{i,1} \vee l_{i,2} \vee l_{i,3}$), we construct a TCSG-T game with threshold m. For every propositional variable in ψ, the game has two skills, s_{y_i} and $s_{\neg y_i}$. For every clause c_j in ψ the game has a skill s_{c_j} and 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 $c_1 = y_1 \vee \neg y_2 \vee \neg y_3$, 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 $S(a_{c_1,2}) = \{s_{c_1}, s_{\neg y_2}\}$ and agent $a_{c_1,3}$ with skills $S(a_{c_1,3}) = \{s_{c_1}, s_{\neg y_3}\}$. 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 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 $S(t_{(y_i,\neg y_i,j)}) = \{s_{y_i}, s_{\neg y_i}\}$. 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 ψ, in which the variables assigned true are $y_{t1}, \ldots, y_{tx}$ 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 $a_{c_j,x}$ to C. Coalition C covers all the clauses c_j of ψ, 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 $t_{(y_i,\neg y_i,j)}$ 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 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}). $\square$

4.4. The Core

We now consider the complexity of calculating the core of CSGs, or checking if it is empty. We denote the set of all agents except a_i as $I_{-a_i} = I \setminus \{a_i\}$.

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 $\sum_{i=0}^{n} p_i = 1$ and since $p_i > 0$ we get that $p(C) \leq \sum_{a_j \in I_{-a_i}} p_j < 1$, 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 $\sum_{a_j \in C} p_j = 1$, 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.[11]

Theorem 8. *CORE and CORE-NON-EMPTY is in P for STSG, TCSG-T and WTSG-T.*

Proof. Due to Theorem 4, for these games we can find all the veto agents in polynomial time. One representation of the core in any simple game is a list of veto agents, as in such games any imputation that allocates all the payoff among the veto agents, in any way, is in the core and any other imputation is blocked (if there are not veto agents in such games the core is empty). Since we can generate the list of veto agents in polynomial time, we can compute the core in polynomial time. $\square$

Theorem 9. *CORE-NON-EMPTY is* in *co-NP for STSG, TCSG, TCSG-T and WTSG, and WTSG-T.*[12]

Proof. Malizia et al. [48] show that CORE-NON-EMPTY is *in* co-NP for any coalitional game where the coalitional function can be computed in polynomial time. The result follows by Theorem 3.[13] $\square$

4.5. The ϵ-Core and Least-Core

We now consider ϵ-core and least-core related problems. Determining whether a certain imputation $p = (p_1, \ldots, p_n)$ is in the ϵ-core is equivalent to testing whether the maximal excess of any coalition is at most ϵ. We can thus focus on the complexity of finding the maximal excess of any coalition given an imputation.[14] For this, it is sufficient to study ECM.

[11]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.

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

[13]We thank an anonymous reviewer of the earlier conference version of this paper for directing us to Malizia et al. [48].

[14]Given a polynomial algorithm to compute the maximal excess of any coalition given an imputation, we can test ϵ-core membership: if the maximal excess is at most ϵ then

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

Definition 27 (Weighted Set-Cover (WSC))**.** *We are given a set E of elements and a collection of subsets $S = \{S_1, \ldots, S_n\}$ where $S_i \subseteq E$ and $\cup S_i = E$, 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 $\cup_{S_i \in S'} S_i = E$) 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 [39].

Theorem 10. *ECM is co-NP-Complete in any of the CSG forms: STSGs, TCSG, WTSG, TCSG-T, and WTSG-T.*[15]

Proof. A coalition C with an excess greater than ϵ (so $e(C) = v(C) - p(C) \geq \epsilon$) violates the ϵ-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 ϵ for some $\epsilon \geq 0$. Equivalently, ECM requires making sure that the *maximal* excess across all coalitions is at most ϵ. Due to Theorem 3, 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.

this is an ϵ-core imputation and otherwise it is not. On the other hand, given the ability to test for ϵ-core membership, we can also compute the maximal excess, by performing a "binary search", querying whether the imputation is in the ϵ_1-core, ϵ_2-core, ϵ_3-core and so on, where ϵ_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.

[15]Following the conference version of this paper, Aziz et al. have studied threshold versions of monotone cooperative games [2]. They have examined the *length* of simple cooperative games—the size of the smallest winning coalition. They have shown that computing the length of an STSG is an NP-hard problem. Further, they have shown that if computing the length of a simple game is NP-hard, testing for ϵ-core membership (ECM) is also hard. This provides an alternative proof to the one given in this theorem. We have chosen to keep the current proof as it ties together ECM with the weighted set cover problem (WSC). In addition to this relation resulting in certain inapproximability results, our polynomial algorithm for restricted cases (see Theorem 11) relies on this relation as it uses a method for solving a restricted case of WSC.

We now show that computing the maximal excess $e_{\max} = \max\{e(C)|C \subseteq I\}$ even in the restricted class of STSG is equivalent to a weighted set-cover problem. Any losing coalition C has a non-positive excess (i.e., $p(C) \geq v(C)$) under any imputation. Thus, the maximal excess occurs for some winning coalition, and we have $e_{\max} = \max\{e(C)|v(C) = 1\}$. Any winning coalition C has $v(C) = 1$, so the maximal excess occurs for a winning coalition C that minimizes $p(C)$. By definition $p(C) = \sum_{i \in C} p_i$, so finding the maximal excess coalition requires finding a subset of agents C whose skill set $S(C) = \cup_{i \in C'} S_i$ covers all skills S (i.e., $\cup_{i \in C'} S_i = S$) and whose sum of payoffs $p(C) = \sum_{i \in C} p_i$ is minimal. This is simply a weighted set-cover instance with subsets $S = \{S_1, \ldots, S_n\}$ and weights $p = \{p_1, \ldots, p_n\}$. $\square$

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 [39].[16]

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 $\Theta(\ln n)$ [37]. A simple algorithm for computing the maximal excess (or for solving ECM) through approximating WSC requires a FPTAS for SET-COVER.

To provide tractable algorithms for the ECM problem, we focus on restricting the inputs. 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 [42].[17]

Consider for example the case of sensor networks, where each agent is present in a certain geographical location, and a "skill" represents an agent's ability to cover a certain geographical point of interest with its sensors. In this case, a point of interest may only be covered by agents who reside in its

[16]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.

[17]Techniques for tree decomposition, and algorithms for problems where the input's structure is known to be somewhat "similar" to a tree, have received much attention in other parts of the literature [55, 17, 1, 29].

vicinity, so a "skill" is owned by agents who reside in a continuous geographical space. Further, in many domains the sensor network may be quite sparse, so each agent would only cover at most a few points of interest. When the coverage is sparse, it is also unlikely that there would be a cycle of agents where every two agents in the cycle share a certain point of interest that both cover. The skill structure in such domains is tree-like in the sense described below, so certain core-related problems that are generally hard can be tractably handled.

We use the approach of Guo and Niedermeier [42], who showed that when each of the subsets has a size at most b (where b is constant), so $b = max_{S_i \in S}|S_i|$, and when the subset collection is *tree-like*, then WSC can be solved in polynomial time.[18] We first give the definition of a "tree-like" subset collection (Guo and Niedermeier's work provides a few examples for tree-like subset collections [42]):

Definition 28 (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. Many skill domains such as the sensor network domain described above may exhibit these properties, so algorithms tailored for such domains can be used to solve core-related problems in these domains.[19]

Since ECM is equivalent to WSC, and since Guo and Niedermeier [42] 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:

[18]The algorithm in [42] 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 [42] the parameter k is the size of the largest subset in the input.

[19]The results of Tarjan and Yannakakis [65] 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.

Corollary 1. *Given an 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.*

For solving ECM in the other CSG classes, we need another restriction: the number of tasks must also be bounded by a constant q.

Definition 29 (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 11. *In bounded tree-like CSG domains of the classes 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 $S(T') = \cup_{t_j \in T'} S(t_j)$. 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 that can achieve T' we have $p(C) \leq u(T') - \epsilon$ then C has a deficit $e(C) = v(C) - p(C) \geq \epsilon$ and the imputation p is not in the ϵ-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 ϵ, and the imputation p is in the ϵ-core. Thus, to check if the maximal deficit under an imputation is at most ϵ (or in other words, to solve ECM), it suffices to examine 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 [42] allows testing the minimal payoff $p(C)$ of any coalition that covers $S(T')$: similarly to the proof of Theorem 10, 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. □

The above Corollary 1 and Theorem 11 indicate that in certain restricted domains, *verifying* ϵ-core imputations (or computing the maximal excess under a given imputation) can be performed in polynomial time. We now show that in these restricted domains, the problems of *finding* ϵ-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 12. *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 ϵ-core. The program simply considers all the possible 2^n coalitions over the n players, $C_1, \ldots, C_{2^n}$. The ϵ-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.

Feasible	$(p_1, \ldots, p_n)$	s.t.:
1.1.	$v(C_1) - \sum_{i \mid a_i \in C_1} p_i < \epsilon$	(Coalition C_1 constraint)
1.2.	$v(C_2) - \sum_{i \mid a_i \in C_2} p_i < \epsilon$	(Coalition C_2 constraint)
...		
1.2^n.	$v(C_{2^n}) - \sum_{i \mid a_i \in C_{2^n}} p_i < \epsilon$	(Coalition C_{2^n} constraint)
2.	$\sum_{i=1}^{n} p_i = v(I)$	(Imputation constraint)

Table 2: Exponential linear program for the core

The above program is a feasibility linear program, and any solution for it, $p = (p_1, \ldots, p_n)$, is an imputation in the ϵ-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 ϵ-core imputation or show that the ϵ-core is empty.[20] Our proofs of Corollary 1 and

[20]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

Theorem 11 used the relation between ECM and Weighted Set-Cover, and taking a candidate imputation $p = (p_1, \dots, p_n)$ 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 12 shows that under the above restrictions regarding the CSG domain, given a certain ϵ it is possible to find an imputation in the ϵ-core in polynomial time if one exists, or determine that the ϵ-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.*[21]

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 ϵ such that the ϵ-core is non-empty, applying the algorithm of Theorem 12 on each tested value. □

Thus, although Theorem 10 shows that in general (non-tree-like) CSGs, even testing for an ϵ-core imputation is hard, the key questions regarding the least-core and ϵ-core are tractably solvable for more restricted domains.

4.6. 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. The problems SHAPLEY and BANZHAF have decision problem versions: testing whether the Shapley value or Banzhaf power index of an

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 [59, 41].

[21]The least-core value is computed up to a desired degree of accuracy, so the methods here compute ϵ'_{min} such that the distance between the true ϵ_{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 ϵ'_{min}-core.

agent exceeds a certain threshold K. We show that these decision problems are NP-hard as a corollary of our result regarding the DUMMY problem.

Corollary 3. *The decision versions of SHAPLEY and BANZHAF are NP-hard in TCSG-T and WTSG-T.*

Proof. DUMMY is NP-hard in these domains, due to Theorem 7. 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. □

The decision versions of SHAPLEY and BANZHAF are NP-hard, but may not even be in NP, so these problems may not be NP-complete. We show a stronger result of #P-completeness for the Banzhaf value, for all domains. We first define two #P-complete problems:

Definition 30 (#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 $\cup_{S_i \in C'} S_i \supseteq S$. Given S and C, we are asked to compute the* number *of different set covers of S.*

Definition 31 (#VERTEX-COVER (#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 $v \in V'$.*

We note that #VC is a special case of #SC, where the collection S to be covered is the set of all the edges and the collection of subsets includes a subset for every vertex, which contains all the edges in which this vertex occurs (i.e., the subset representing the vertex v contains all the edges that have v as one of their endpoints). The problem #SC is known to be #P-complete, even for some very restricted classes of graphs such as planar bipartite graphs of degree at most four [66]. Since checking if a collection of subsets of a target set S indeed covers all the elements of S, the problem #SC is also #P-complete.

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

Theorem 13. *BANZHAF in STSG, TCSG, WTSG, TCSG-T and WTSG-T is #P-complete.*[22]

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 3 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 $S_{n+1} = \{s_{new}\}$, 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. □

5. Similar Cooperative Game Representation Languages

Related research deals with similar models of cooperation among agents, and alternative cooperative game representation languages. We now examine

[22]Following the conference version of this paper, the computational complexity of computing the Shapley value in STSGs, TCSGs and WTSGs was examined in Aziz et al. [3]. Aziz et al. have shown that computing the Shapley value in these CSG versions is #P-complete, strengthening our results, which only showed NP-hardness. In fact, they show that in any representation language that allows adding a dummy agent to any game computing the Shapley value is #P-hard if computing the Banzhaf index is #P-hard.

similarities and differences between these languages and the CSG representation, both in terms of expressivity and computational aspects. We provide a general discussion regarding many representation languages, and provide a deeper discussion regarding the representations we believe are closest to the CSG representation: linear production games [52], the anonymous proof solution representation [68] and Coalitional Resource Games [67].

5.1. Expressiveness And Succinctness Of Representation Languages

Section 1.3 discussed several representation languages proposed for cooperative games. In Section 3 we showed that even STSGs, the most restrictive CSG class, can represent any simple increasing game. Further, we showed that WTSGs can represent any increasing game. This completeness in terms of expressivity is a significant advantage that CSGs have over other representation languages that are limited in expressiveness. The most prominent restricted representation is the class of weighted voting games [14, 50, 33, 35], where each agent has a weight, and a coalition of agents wins the game if the sum of the weights of its participants exceeds a certain threshold.

While some characteristic functions cannot be represented as a weighted voting game (see Elkind et al. [34] for a discussion of the expressiveness of weighted voting games), certain problems such as computing the core or finding veto agents can be solved in polynomial time in these games. Interestingly, even in this simple representation, certain problems are already computationally hard, such as testing whether an agent is a dummy, computing power indices or computing the least-core and the nucleolus.

Other examples of restricted cooperative game representations are cooperative graph games [30] and various forms of network games [12, 3, 13]. Many such incomplete representations admit polynomial algorithms for problems that are computationally hard in CSGs, such as computing power indices or core-related problems. Thus, if a game is given in one of these restricted representations, it is better to use algorithms tailored for these representations. On the other hand, these languages are not fully-expressive, and so for each of them there are some characteristic functions that cannot be represented using the language.

Recently, Aziz et al. [2] have examined the impact of certain restrictions on CSGs that allow them to be solved more efficiently. Specifically, they show that in STSGs where the total number of skills is bounded by a constant, it is possible to compute the least-core in polynomial time (as well as the Cost-of-Stability and the nucleolus [10, 58]). However, this restriction regarding the bounded number of skills is severe. Our algorithm for computing the least-core in Section 4.5 only requires that the number of skills *any single agent* may have is bounded, but allows an arbitrary number of skills in total. On the other hand, our algorithm has the additional requirement of a tree-like structure.

Previous work on computational game theory has uncovered various *fully-expressive* representation languages, which can describe *any* characteristic function. Examples include the marginal contribution network language (MC-Nets) [44], the sparse synergy representation [23], and the Multi-Attribute Coalitional Games (MACGs) [45], which generalize both of these.[23] Any fully-expressive representation can represent any characteristic function, but different representations may require a different amount of space to represent the same game. Work on representing functions in a concise form has provided a insightful definition regarding the succinctness of representations [18, 47, 25, 20]: a language L is at least as expressive as a language L' if there is a polynomial translator that converts any input of L' into an equivalent input in L (where 'equivalence' means that the value functions coincide).[24] The running time of algorithms that analyze cooperative games depends on the input size, so a polynomial algorithm that takes the input in one language may not be useful if the input is given in another language—the input would need to be converted to the required language, which may take an exponential amount of time and may result in an exponentially long representation. CSGs are closely related to the set-cover problem, and can compactly represent domains where agents must have a certain resource or cover certain skills in order to achieve a task. In contrast, marginal contribution networks and the sparse synergy representation are quite different from a set-cover domain, and so there appears to be no straightforward conversion from these forms into a compact CSG representation. Similarly, we do not see a way to convert a compact CSG representation into a compact form of

[23]Ieong et al. [45] provide a full discussion of the relation between marginal contribution networks, the sparse synergy representation, and MACGs.

[24]Note that some of our complexity results imply a succinctness gap: CSGs are strictly more succinct than the explicit representation unless $P = NP$ (some problems, such as DUMMY, are hard for WTSGs, but can trivially be solved in polynomial time when the input is given in the explicit representation, as this input under the explicit representation is already of a size exponential in the number of agents). We thank an anonymous reviewer for pointing this out.

these other representations.

MACGs [45] are a general framework for building representation languages for cooperative games, and is in fact a "meta-model." The properties of a MACG representation depends on the choice of an *aggregator* which takes attributes associated with agents and outputs a numeric value. Specifically, the expressive power of an MACG representation and its computational properties depend on the the aggregators used. Thus, the computational properties of the MACG model must be examined under a *a specific choice of aggregators.* It is possible to express CSGs using certain such aggregators, such as an aggregator that tests whether at least one of the agents in a set is endowed with a certain skill. In this sense, CSGs can be thought of as a restricted case of the MACG framework, so our analysis provides algorithmic results for this specific aggregator.

We now consider the representations that we see as closest to the CSG representation language.

5.2. Linear Production Games

Linear production games [52] are games where agents pool together resources in order to manufacture finished goods that can be sold at a given market price. In such games, there are n different types of input resources, and each agent is endowed with a certain quantity of each resource. Similarly, there are k different finished goods that can be produced, each of which can be sold at a certain market price. Each such finished good requires a certain amount of the n different input resources. When several agents form a coalition, they have at their disposal the sum of the individual endowments of the input resources. The utility a coalition can produce is the maximal revenue the coalition can achieve for the finished goods it can produce (fractional quantities of goods are allowed).

Linear production games have certain similarities with CSGs, and especially with WTSGs. Both have different "resources" with which agents are endowed—the input resources in linear production games and the agent skills in CSGs. Similarly, there are outputs that coalitions of agents strive to achieve—finished goods in linear production games, and completed tasks in CSGs. Finally, the market prices of finished goods in linear production games are somewhat parallel to the weights of tasks in WTSG.

The key difference between linear production games and CSGs is that in CSGs the input resources have no *quantities.* As these resources in CSGs are skills, they can be used as many times as required. In contrast, quantities lie at the heart of linear production games, where the amounts of the finished goods a coalition can manufacture depends on the amounts of input resources a coalition has, and the amount required to produce each finished good.

Interestingly, a key result regarding linear production games is that they always have a non-empty core [52]. In fact, a certain core imputation can be found in these games using linear programming duality. Since CSGs can express games with empty cores, there are some games that can be expressed as a CSG, but not as a linear production game. On the other hand, for some CSG forms, even testing whether the core is empty is computationally hard, whereas a core imputation always exists in a linear production game, and can be found in polynomial time.

5.3. Anonymous Proof Solutions

Yokoo et al. [68] consider open, anonymous coalitional 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 for any $a_i \neq a_j$ we have $S_i \cap S_j = \phi$. The characteristic function of the game is defined on the set of skills that a coalition has: $v : 2^S \to \mathbb{R}$. Yokoo et al. [68] do not consider the computational complexity of calculating solution concepts, focusing instead on strategic agent behavior.

Still, the expressiveness of their anonymous-environment model is essentially equivalent to that of *general* CSGs. In one direction, any game compactly represented by the anonymous-environment model can also be represented as a compact 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. In the other direction, the anonymous-environment model can directly map a subset of skills that a coalition has to the utility of that coalition, and so can express instances described in the CSG model, which defines the utility through tasks. On the other hand, our task-based CSG representation is at least as compact as the anonymous-environment representation, and can sometimes save an exponential amount of space over that representation. The reason for this is that if a certain skill subset $S' \subset S$ enables the achievement of 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).

5.4. Coalitional Resource Games

Wooldridge and Dunne [67] 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 is in regard to properties of goal subsets that are *successful*—both feasible and satisfying for a coalition. Wooldridge and Dunne [67] 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); and whether a successful goal set G' for a coalition is optimal in its use of the resource r (co-NP complete). Many results for CRGs are negative, so that although small representations, answering many questions regarding them is computationally hard.

Formally, Coalitional Resource Games (CRGs) 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 $en : I \times R \rightarrow \mathbb{N}$ mapping an agent and a resource to the amount of that resource that the agent has, and a requirement function $req : G \times R \rightarrow \mathbb{N}$ 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 $en(C, r_j) = \sum_{a_i \in C} en(a_j, r_j)$. 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 $sat(C) = \{G' \subset G | \forall a_i \in C,\ G_i \cap G' \neq \phi\}$. 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 $feas(C) = \{G' \subset G | \forall r_j \in R,\ req(G', r_j) \leq en(C, r)\}$.

Our model of CSGs is somewhat similar to that of CRGs.[25] CSGs define

[25] Another somewhat related class of games are *Cooperative Boolean Games* (CBGs) [32].

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). This can be modeled 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 expressive 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 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, such 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

These games can be viewed as a hybrid of CRGs and a non-cooperative game proposed by Harrensein et al. [43]. Despite structural similarities, CBGs are not transferable-utility cooperative games, and so it is not possible to compare their expressiveness with CSGs.

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 satisfied just by the project having succeeded, but rather gain utility according to the share of the revenue received.

6. Conclusions

We examined a simple but expressive model of cooperation among agents, Coalitional Skill Games (CSGs). 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 we could calculate the value of a coalition in polynomial time in all restricted forms, although some problems of interest remain computationally hard. Some key results provide 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 ϵ-core and least-core in general domains and regarding computing power indices. Power indices, however, can be approximated using general algorithms. On the other hand, this work only provided positive results regarding ϵ-core and least-core related problems in CSGs in quite restricted domains. It would be interesting to see if there are other restricted domains where such problems can be tackled, and to find a power index approximation method tailored for CSGs that would outperform such generic power index approximation schemes. Second, there are game-theoretic solution concepts refining the least-core, such as the nucleolus [58], and a further step would be to examine the complexity of computing these solution concepts. We conjecture that computing the nucleolus is hard in general CSGs but could be handled in restricted domains. Third, it is still open to decide whether core non-emptiness is co-NP-complete for TCSGs and WTSGs. Finally, it would be interesting to see how CSG-based models can be used in applied settings.

7. Acknowledgment

This work was partially supported by Israel Science Foundation grant #898/05, by Israel Ministry of Science and Technology grant #3-6797, and by the Google Inter-University Center for Electronic Markets and Auctions.

References

[1] Alber, J., Bodlaender, H., Fernau, H., Kloks, T., and Niedermeier, R. Fixed parameter algorithms for dominating set and related problems on planar graphs. *Algorithmica 33*, 4 (2002), 461–493.

[2] Aziz, H., Brandt, F., and Harrenstein, P. Monotone cooperative games and their threshold versions. In *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)* (Toronto, May 2010), pp. 1017–1024.

[3] Aziz, H., Lachish, O., Paterson, M., and Savani, R. Spanning connectivity games. *Arxiv preprint arXiv:0906.3643* (2009).

[4] Aziz, H., and Paterson, M. False name manipulations in weighted voting games: splitting, merging and annexation. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)* (2009), International Foundation for Autonomous Agents and Multiagent Systems, pp. 409–416.

[5] Bachrach, Y. The least-core of threshold network flow games. *Mathematical Foundations of Computer Science 2011* (2011), 36–47.

[6] Bachrach, Y., and Elkind, E. Divide and conquer: False-name manipulations in weighted voting games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 975–982.

[7] Bachrach, Y., Elkind, E., Meir, R., Pasechnik, D., Zuckerman, M., Rothe, J., and Rosenschein, J. S. The cost of stability in coalitional games. In *The Second International Symposium on Algorithmic Game Theory (SAGT 2009)* (Paphos, Cyprus, October 2009), pp. 122–134.

[8] Bachrach, Y., Markakis, E., Procaccia, A. D., Rosenschein, J. S., and Saberi, A. Approximating power indices. In *The Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)* (2008), pp. 943–950.

[9] Bachrach, Y., Meir, R., Jung, K., and Kohli, P. Coalitional structure generation in skill games. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI-2010)* (2010), pp. 703–708.

[10] Bachrach, Y., Meir, R., Zuckerman, M., Rothe, J., and Rosenschein, J. S. The cost of stability in weighted voting games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009)* (2009), pp. 1289–1290.

[11] Bachrach, Y., and Rosenschein, J. Coalitional skill games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 1023–1030.

[12] Bachrach, Y., and Rosenschein, J. S. Power in threshold network flow games. *Autonomous Agents and Multi-Agent Systems 18*, 1 (2009), 106–132.

[13] Bachrach, Y., Rosenschein, J. S., and Porat, E. Power and stability in connectivity games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 999–1006.

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

[15] Baye, M., Kovenock, D., and De Vries, C. Rigging the lobbying process: an application of the all-pay auction. *The American Economic Review* (1993), 289–294.

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

[17] Bodlaender, H. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the Twenty-Fifth*

annual ACM symposium on Theory of computing (1993), ACM, pp. 226–234.

[18] Cadoli, M., Donini, F. M., Liberatore, P., and Schaerf, M. Comparing space efficiency of propositional knowledge representation formalisms.

[19] Chalkiadakis, G., Elkind, E., and Wooldridge, M. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning 5*, 6 (2011), 1–168.

[20] Chevaleyre, Y., Endriss, U., Lang, J., et al. *Expressive power of weighted propositional formulas for cardinal preference modelling*. Institute for Logic, Language and Computation (ILLC), University of Amsterdam, 2006.

[21] Chowdhury, J. The motivational impact of sales quotas on effort. *Journal of Marketing Research* (1993).

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

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

[24] Conitzer, V., and Sandholm, T. 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)* (2004), pp. 219–225.

[25] Coste-Marquis, S., Lang, J., Liberatore, P., and Marquis, P. Expressive power and succinctness of propositional languages for preference representation. *Proceedings of KR-2004* (2004).

[26] Czajkowski, K., Foster, I., Kesselman, C., Sander, V., and Tuecke, S. Snap: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In *Job scheduling strategies for parallel processing* (2002), Springer, pp. 153–183.

[27] Dang, V. D., Dash, R. K., Rogers, A., and Jennings, N. R. Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'06)* (2006), p. 635.

[28] Dash, R., Jennings, N., and Parkes, D. Computational-mechanism design: A call to arms. *Intelligent Systems, IEEE 18*, 6 (2003), 40–47.

[29] Demaine, E., Fomin, F., Hajiaghayi, M., and Thilikos, D. Subexponential parameterized algorithms on bounded-genus graphs and h-minor-free graphs. *Journal of the ACM (JACM) 52*, 6 (2005), 866–893.

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

[31] Denzinger, J., and Kronenburg, M. Planning for distributed theorem proving. In *Proc. KI-96 (LNAI Volume 1137)* (1996), pp. 43–56.

[32] Dunne, P., van der Hoek, W., Kraus, S., and Wooldridge, M. Cooperative boolean games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems* (2008), International Foundation for Autonomous Agents and Multiagent Systems, pp. 1015–1022.

[33] Elkind, E., Goldberg, L., Goldberg, P., and Wooldridge, M. Computational complexity of weighted threshold games. In *Proceedings of the 22nd National Conference on Artificial intelligence (AAAI'07)* (2007), pp. 718–723.

[34] Elkind, E., Goldberg, L., Goldberg, P., and Wooldridge, M. On the dimensionality of voting games. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-2008)* (2008), pp. 13–17.

[35] Elkind, E., and Pasechnik, D. Computing the nucleolus of weighted voting games. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms* (2009), Society for Industrial and Applied Mathematics, pp. 327–335.

[36] Faliszewski, P., and Hemaspaandra, L. The complexity of power-index comparison. *Algorithmic Aspects in Information and Management* (2008), 177–187.

[37] Feige, U. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM) 45*, 4 (1998), 634–652.

[38] Fisher, M., and Wooldridge, M. Distributed problem-solving as concurrent theorem proving. In *Multi-Agent Rationality (MAAMAW-97)*. Springer-Verlag, 1997.

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

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

[41] Grötschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimization.* Springer-Verlag, New York, 1988.

[42] Guo, J., and Niedermeier, R. Exact algorithms and applications for Tree-like Weighted Set Cover. *Journal of Discrete Algorithms 4*, 4 (2006), 608–622.

[43] Harrenstein, P., van der Hoek, W., Meyer, J., and Witteveen, C. Boolean games. In *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge* (2001), Morgan Kaufmann Publishers Inc., pp. 287–298.

[44] Ieong, S., and Shoham, Y. Marginal contribution nets: A compact representation scheme for coalitional games. In *Proceedings of the 6th ACM Conference on Electronic Commerce* (2005), pp. 193–202.

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

[46] Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., and Shimada, S. RoboCup Rescue: search

and rescue in large-scale disasters as a domain for autonomous agents research. In *1999 IEEE International Conference on Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings* (1999), vol. 6, pp. 739–743.

[47] Liberatore, P., and Schaerf, M. Arbitration (or how to merge knowledge bases). *Knowledge and Data Engineering, IEEE Transactions on 10*, 1 (1998), 76–90.

[48] Malizia, E., Palopoli, L., and Scarcello, F. Infeasibility certificates and the complexity of the core in coalitional games. In *Proceedings of the International Joint Conference on Artificial Intelligence* (2007), pp. 1402–1407.

[49] Mantrala, M., Sinha, P., and Zoltners, A. Structuring a multiproduct sales quota-bonus plan for a heterogeneous sales force: A practical model-based approach. *Marketing Science* (1994), 121–144.

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

[51] Owen, G. Multilinear extensions and the Banzhaf Value. *Naval Research Logistics Quarterly 22*, 4 (1975), 741–750.

[52] Owen, G. On the core of linear production games. *Mathematical programming 9*, 1 (1975), 358–370.

[53] Rasmusen, E. Lobbying when the decisionmaker can acquire independent information. *Public Choice 77* (1993), 899–913.

[54] Resnick, E., Bachrach, Y., Meir, R., and Rosenschein, J. S. The cost of stability in network flow games. *Mathematical Foundations of Computer Science 2009* (2009), 636–650.

[55] Robertson, N., and Seymour, P. Graph minors. ii. algorithmic aspects of tree-width. *Journal of algorithms 7*, 3 (1986), 309–322.

[56] Rosenschein, J. S., and Genesereth, M. R. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence* (Los Angeles, California, August 1985), pp. 91–99.

[57] Scerri, P., Pynadath, D., Johnson, L., Rosenbloom, P., Si, M., Schurr, N., and Tambe, M. A prototype infrastructure for distributed robot-agent-person teams. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (2003), pp. 433–440.

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

[59] Schrijver, A. *Theory of Linear and Integer Programming.* Wiley and Sons, 1986.

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

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

[62] Shapley, L. S., and Shubik, M. Quasi-cores in a monetary economy with nonconvex preferences. *Econometrica: Journal of the Econometric Society 34*, 4 (1966), 805–827.

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

[64] Sims, M., Goldman, C. V., and Lesser, V. Self-organization through bottom-up coalition formation. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (2003), pp. 867–874.

[65] Tarjan, R. E., and Yannakakis, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing 13* (1984), 566.

[66] Vadhan, S. The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing 31*, 2 (2002), 398–427.

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

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

[69] Zlotkin, G., and Rosenschein, J. S. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the National Conference on Artificial Intelligence (AAAI'94)* (Seattle, Washington, August 1994), pp. 432–437.

[70] Zuckerman, M., Faliszewski, P., Bachrach, Y., and Elkind, E. Manipulating the quota in weighted voting games. In *The Twenty-Third National Conference on Artificial Intelligence (AAAI-2008)* (2008), pp. 215–220.