/ The University of Electro-Communications

Coalition structure generation in cooperative games with compact representations

Coalition structure generation in cooperative games with compact representations

Suguru Ueda[1] · **Atsushi Iwasaki[2]** · **Vincent Conitzer[3]** · **Naoki Ohta[4]** · **Yuko Sakurai[5]** · **Makoto Yokoo[6]**

Abstract This paper presents a new way of formalizing the coalition structure generation problem (CSG) so that we can apply constraint optimization techniques to it. Forming effective coalitions is a major research challenge in AI and multi-agent systems. CSG involves partitioning a set of agents into coalitions to maximize social surplus. Traditionally, the input

✉ Suguru Ueda
    sgrueda@cc.saga-u.ac.jp

    Atsushi Iwasaki
    iwasaki@is.uec.ac.jp

    Vincent Conitzer
    conitzer@cs.duke.edu

    Naoki Ohta
    n-ohta@fc.ritsumei.ac.jp

    Yuko Sakurai
    yuko.sakurai@aist.go.jp

    Makoto Yokoo
    yokoo@inf.kyushu-u.ac.jp

[1] Department of Information Science, Saga University, Saga, Japan

[2] Graduate School of Informatics and Engineering, University of Electro-Communications, Tokyo, Japan

[3] Department of Computer Science, Duke University, Durham, USA

[4] College of Information Science and Engineering, Ritsumeikan University, Shiga, Japan

[5] Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology, Tokyo, Japan

[6] Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka, Japan

of the CSG problem is a black-box function called a characteristic function, which takes a coalition as input and returns the value of the coalition. As a result, applying constraint optimization techniques to this problem has been infeasible. However, characteristic functions that appear in practice often can be represented concisely by a set of rules, rather than treating the function as a black box. Then we can solve the CSG problem more efficiently by directly applying constraint optimization techniques to this compact representation. We present new formalizations of the CSG problem by utilizing recently developed compact representation schemes for characteristic functions. We first characterize the complexity of CSG under these representation schemes. In this context, the complexity is driven more by the number of rules than by the number of agents. As an initial step toward developing efficient constraint optimization algorithms for solving the CSG problem, we also develop mixed integer programming formulations and show that an off-the-shelf optimization package can perform reasonably well.

Keywords Multiagent systems · Cooperative games · Coalition structure generation · Compact representation

1 Introduction

Coalition formation is an important capability in automated negotiations among self-interested agents. Coalition structure generation (CSG) involves partitioning a set of agents into coalitions to maximize social surplus. This problem has become a popular research topic in AI and multi-agent systems (MAS) [4]. Possible CSG applications include distributed vehicle routing [32], multi-sensor networks [7], and so on. The CSG problem is equivalent to a complete set partition problem [44], and various algorithms have been developed for solving it. Sandholm et al. [31] propose an anytime algorithm with worst-case guarantees. However, to obtain an optimal coalition structure, this algorithm must check all of the coalition structures. Thus, the worst-case time complexity is $O(n^n)$, where n is the number of agents. On the other hand, dynamic programming (DP) based algorithms [21,25,44] are guaranteed to find an optimal solution in $O(3^n)$. The CSG problem can also be considered in partition function games (PFGs), where the value of a coalition depends on how the other agents are partitioned. Rahwan et al. [27] first considered the CSG problem in PFGs. We expand the discussion of related literature of CSG problems in the next subsection.

Such existing works on CSG assume that the characteristic function is represented implicitly, and we have only oracle access to the function, where the value of a coalition (or a coalition structure as a whole) can be obtained using some procedure. This is because representing an arbitrary characteristic function explicitly requires $\Theta(2^n)$ numbers, which is prohibitive for a large n. When a characteristic function is represented by a black-box function, there is no room for applying constraint optimization techniques.

However, characteristic functions that appear in practice often display significant structure, and such characteristic functions can probably be represented much more concisely. Indeed, recently, several new methods for representing characteristic functions have been developed [5,6,14,20]. These representation schemes, which capture the characteristics of the interactions among agents in a natural and concise manner, can significantly reduce the representation size. It is natural to assume that an organizer who wants to solve a CSG problem has knowledge on possible interactions among agents and can concisely represent her knowledge by a set of rules. For example, let us consider a situation where a professor is

dividing students in her laboratory into several research groups. Each student has a specific feature, e.g., good/bad at programming, theory, writing, etc. The professor knows the synergies among these features, e.g., there is positive synergy between a student who is good at programming and another student who is good at theory. From the knowledge about students' features and the synergies, the professor can construct a set of rules. Surprisingly, to our knowledge, prior to our work these representation schemes had not yet been used for CSG, which is our goal in this paper. Using these compact representation schemes, a characteristic function is represented by a set of rules, rather than treating the function as a black box. The idea is to solve the CSG problem more efficiently by directly applying constraint optimization techniques to this compact representation.

We examine three representative compact representation schemes: (i) marginal contribution nets (MC-nets) [14] and embedded marginal contribution nets [20], (ii) synergy coalition groups (SCGs) [6], and (iii) SCGs in multi-issue domains (SCGs in MID) [5]. The optimal choice of a representation scheme depends on the application.

Quite interestingly, we find that there exists some common structure among these cases; in essence, the problem is to find a subset of rules that maximizes the sum of rule values under certain constraints. For each case, we show that the CSG problem is NP-hard, and the size of a problem instance is naturally measured by the number of rules rather than the number of agents. Furthermore, as an initial step toward developing efficient constraint optimization algorithms for solving the CSG problem, we give a mixed integer programming (MIP) formulation that captures the above structure. We show that an off-the-shelf optimization package (CPLEX) can solve the resulting MIP problem instances reasonably well.

1.1 Related works

This subsection briefly explores related work. Traditional models of coalitional game theory often assume that the characteristic function is super-additive, forming the grand coalition is guaranteed to be optimal, and the main research topic in economics is how to divide the gain of the grand coalition among agents. The traditional theory of coalitional games provides a number of solution concepts, such as the core [10], the Shapley value [34], and the nucleolus [33]. The main research topic in computer science is to analyze the computational complexity of problems related to these solution concepts. Since the seminal work by Megiddo [19], many works have been conducted, e.g., [8,9,11,12,17].

More recently, AI and MAS researchers have been considering the case where the characteristic function is not super-additive, i.e., where forming the grand coalition is not optimal. Often, in such a case, agents should form a coalition structure to maximize the reward they can obtain. This is called the coalition structure generation (CSG) problem, which has been an active research topic in AI and MAS.[1] Many algorithms for solving the CSG problem have been developed. For example, as we mentioned in the introduction, Sandholm et al. [31] develop an anytime algorithm with worst-case guarantees, and Rahwan et al. [29] develop another anytime algorithm called IP, while Rahwan and Jennings [25] develop a dynamic programming (DP) based algorithm, which runs in $O(3^n)$. Furthermore, Michalak et al. [21] develop an algorithm called ODP-IP by combining the anytime and the DP approaches. To the best of our knowledge, the state-of-the-art algorithm is ODP-IP, which just takes seconds to solve an instance with 25 agents.

The above works assume that a characteristic function is given as a black-box function. However, representing that function requires an exponential number of agent combinations.

---

[1] See [28] for a comprehensive survey of many results in this literature.

Thus, several concise representation schemes for a characteristic function have been proposed: marginal contribution nets (MC-nets) [14], synergy coalition groups (SCGs) [6], and SCGs in multi-issue domains (SCGs in MID) [5]. It is natural to believe that utilizing the structure of a concise representation scheme helps us develop more efficient CSG algorithms. Subsequent to our first conference paper on this topic [24], several papers discuss the CSG problem under some concise representation. For example, Ueda et al. [39] considers the CSG problem where the value of a coalition is calculated by solving a distributed constraint optimization problem [22]. Aziz and de Keijzer [1] and Ueda et al. [40] studied the CSG problem under an agent-type representation where agents are partitioned into several types and agents with the same type are identical.

Mixed integer programming (MIP) is a useful technique to solve an optimization problem like the CSG problem. As another work utilizing MIP technique beside our work, Tran-Thanh et al. [37] have proposed the coalitional skill vector model. In their model, there exists a set of skills and each agent has a skill vector which represents the agents' level. They formalized the CSG problem as a MIP formulation. Since their representation is different from our three representations, their MIP formulation has a different structure: 2^n decision variables with an exponential number of constraints. To solve this MIP formulation, they formalized an LP relaxation problem and its dual problem. Then they developed a constraint generation based algorithm that solved the instances with 500 agents in less than an hour.

While the value of a coalition depends on its agents in characteristic function games, it can be affected by how the others are partitioned if we consider real-world applications. Such a game, which is represented by a partition function, is called a partition function game (PFG) [36]. In economics, the above solution concepts are extended to handle such games with externalities. The representative solution concepts include the Myerson value [23], which is an extension of the Shapley value for PFG. On the other hand, associated computational problems have been considered by AI and MAS researchers. Rahwan et al. [27] first considered the CSG problem in the restricted classes of PFGs where only positive or negative externalities exist. Michalak et al. [20] extend MC-nets to handle externalities, and their proposed representation is called the embedded MC-nets representation. Skibski et al. [35] propose another representation called Partition Decision Trees and developed efficient algorithms that compute the extensions of the Shapley value in polynomial time.

Another research line in AI and MAS considers games on graphs. In these researches, the existence of an underlying graph is assumed, and the graph represents, for example, a communication network among agents. Voice et al. [42] introduced the independence of disconnected members (IDM) property in which two agents do not affect each other if they are disconnected on the graph. They formalized a graph coalition structure problem (GCSG) where a characteristic function satisfies the IDM property and examined the computational complexity of GCSG. They also developed algorithms for various types of graphs. Furthermore, Voice et al. [43] proposed Coalition Formation with Sparse Synergies (CFSS), where a coalition is feasible if and only if there exists a connected subgraph of the given underlying graph. However, a CSG algorithm for CFSS will be inefficient since the search space grows exponentially with regards to the number of agents. To overcome this issue, Bistaffa et al. [2] proposed an anytime algorithm, which provides an anytime solution with quality guarantees.

Rahwan et al. [26] proposed a very general framework for constrained coalition formation (CCF) games. Even though SCG and CCF utilize an organizer's knowledge of the relations among the agents, they have different properties for representing games. Assume an organizer only knows (1) coalitions where the synergies exist, and (2) the values of these coalitions. If the synergies are sparse, the organizer can directly and concisely represent her knowledge using SCG. It is possible to represent her knowledge on (1) using CCF, where the synergies

are represented as positive constraints. However, using CCF, the organizer must provide a characteristic function as well. In principle, a characteristic function takes any coalition as its argument and returns its value, regardless of the coalition has synergy or not. It is not obvious whether the organizer can concisely represent such a characteristic function. If we explicitly represent a characteristic function as a table, we require $O(2^n)$ space. Liao et al. [18] proposed a CSG algorithm that utilizes an MaxSAT solver. However, their SAT encoding method is for characteristic function games and cannot handle partition function games in which externalities exist among coalitions. On the other hand, our method can handle a partition function game represented as embedded MC-nets. Iwasaki et al. [15] develop an empirically efficient algorithm for computing imputation in such situations. They further propose a new solution concept called weak ε-core$^+$.

2 Model

2.1 Characteristic function games

Let A be the set of all agents, where $|A| = n$. We assume a characteristic function game, i.e., the value of coalition S is given by characteristic function v. Characteristic function $v : 2^A \rightarrow \mathbb{R}$ assigns a value to each set of agents (coalition) $S \subseteq A$. Without loss of generality, we assume $\forall S \subseteq A, v(S) \geq 0$ holds. As previously shown [31], even if some coalitions' values are negative, as long as each coalition's value is bounded (i.e., not infinitely negative), we can normalize the coalition values so that all the values are non-negative. This rescaled game is strategically equivalent to the original game.

Coalition structure generation (CSG) involves partitioning a set of agents into coalitions to maximize the social surplus. Coalition structure CS is a partition of A into disjoint, exhaustive coalitions. To be more precise, $CS = \{S_1, S_2, \ldots\}$ satisfies the following conditions:

$$\forall i, j \ (i \neq j), \ S_i \cap S_j = \emptyset, \ \bigcup_{S_i \in CS} S_i = A.$$

In other words, in CS, each agent belongs to exactly one coalition, and some agents may be alone in their coalitions.

For example, in a game with three agents, a, b, and c, there are seven possible coalitions: $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{b, c\}$, $\{a, c\}$, $\{a, b, c\}$, and five possible coalition structures: $\{\{a\}, \{b\}, \{c\}\}$, $\{\{a, b\}, \{c\}\}$, $\{\{a\}, \{b, c\}\}$, $\{\{b\}, \{a, c\}\}$, $\{\{a, b, c\}\}$.

We denote by $\Pi(A)$ the space of all coalition structures over A. The value of coalition structure CS, denoted as $V(CS)$, is given by:

$$V(CS) = \sum_{S_i \in CS} v(S_i).$$

Optimal coalition structure CS^* satisfies the following condition:

$$\forall CS \in \Pi(A), V(CS^*) \geq V(CS).$$

We say a characteristic function is super-additive, if for any disjoint sets $S_i, S_j, v(S_i \cup S_j) \geq v(S_i) + v(S_j)$ holds. If the characteristic function is super-additive, solving CSG becomes trivial, i.e., the grand coalition (the coalition of all agents) is optimal.

Super-additivity means that any pair of coalitions is better off by merging into one. One might think that super-additivity holds in most cases since the agents in the composite coalition can work separately and perform at least as well as the case when they were in different

coalitions. However, organizing a large coalition can be costly; e.g., there might be coordination overhead like communication costs or anti-trust penalties. Also, if time is limited, the agents might not have time to carry out the communications and computations required for effective coordination within the composite coalition, so component coalitions may be more advantageous. In any case, even if the characteristic function is superadditive for the simple reason that agents in a composite coalition can always choose to work separately in subteams of the coalition, this still leaves the problem of finding the optimal subteam structure, which is the same problem as the CSG problem we face here. That is, in this case probably the most natural representation of the characteristic function v is a function v' that gives the values of coalitions without considering that they can work in subteams, and we would have to solve the CSG problem with respect to v'. Thus, we assume a characteristic function can be non-super-additive.

Example 1 Assume four agents, a, b, c, and d. The characteristic function is given as follows:

$$
\begin{aligned}
&v(\{a\}) = 3, &&v(\{b\}) = 3, &&v(\{c\}) = 2, \\
&v(\{d\}) = 2, &&v(\{a,b\}) = 6, &&v(\{a,c\}) = 5, \\
&v(\{a,d\}) = 5, &&v(\{b,c\}) = 5, &&v(\{b,d\}) = 5, \\
&v(\{c,d\}) = 2, &&v(\{a,b,c\}) = 8, &&v(\{a,b,d\}) = 8, \\
&v(\{a,c,d\}) = 5, &&v(\{b,c,d\}) = 5, &&v(\{a,b,c,d\}) = 5.
\end{aligned}
$$

In this case, there exist multiple optimal CSs. For example, $\{\{a,b,c\},\{d\}\}$ and $\{\{a,b,d\},\{c\}\}$ are optimal CSs, and the value of these CSs is 10.

2.2 Compact representations

Let us briefly describe three existing compact representation schemes: marginal contribution nets, synergy coalition groups, and multi-issue domains. We first introduce a concise representation of a characteristic function called *marginal contribution networks (MC-nets)*, developed by Ieong and Shoham [14].

Definition 1 (*MC-nets*) An *MC-net* consists of a set of *rules R*. Each rule $r \in R$ is of the following form: $(L_r) \rightarrow v_r$, where L_r is the condition of this rule, which is a conjunction of literals over A, i.e., $a_1 \wedge \cdots \wedge a_k \wedge \neg a_{k+1} \wedge \cdots \wedge \neg a_m$. We call $P_r = \{a_1, \ldots, a_k\}$ *positive literals* and $N_r = \{a_{k+1}, \ldots, a_m\}$ *negative literals*. We say that rule r is *applicable* to coalition S if L_r is true when the values of all Boolean variables that correspond to the agents in S are set to true, and the values of all Boolean variables that correspond to agents in $A \setminus S$ are set to false, i.e., $\bigwedge_{a \in S} a \wedge \bigwedge_{b \in A \setminus S} \neg b \models L_r$ holds. For coalition S, $v(S)$ is given as $\sum_{r \in R_S} v_r$, where R_S is the set of rules applicable to S. Thus, for coalition structure CS, $V(CS)$ is given as $\sum_{S \in CS} \sum_{r \in R_S} v_r$.

In MC-nets, the condition of a rule must be the conjunctions of some literals. Such a rule is *basic*. Also, we call a rule that has a more complicated condition a *non-basic* rule. A non-basic rule must be transformed into multiple basic rules, whose conditions are disjointed from each other. For example, a non-basic rule, which has form $(a \vee b \vee c) \rightarrow v$, is transformed into three basic rules: $(a) \rightarrow v$, $(\neg a \wedge b) \rightarrow v$, and $(\neg a \wedge \neg b \wedge c) \rightarrow v$. Furthermore, without loss of generality, we assume each rule has at least one positive literal. For example, if a rule has form $\neg a_1 \rightarrow 1$ and there exist agents $a_1, a_2, \ldots, a_n$, we can create the following equivalent rules: $\neg a_1 \wedge a_2 \rightarrow 1, \neg a_1 \wedge \neg a_2 \wedge a_3 \rightarrow 1, \ldots, \neg a_1 \wedge \neg a_2 \wedge \cdots \wedge \neg a_{n-1} \wedge a_n \rightarrow 1$.

Example 2 Assume five agents, a, b, c, d, and e, and four rules: $r_1 : (b \wedge e) \rightarrow 3$, $r_2 : (a \wedge b \wedge c \wedge \neg d) \rightarrow 2$, $r_3 : (a \wedge d) \rightarrow 1$, and $r_4 : (c \wedge \neg e) \rightarrow 1$. In this case, r_1 and r_2 are applicable to coalition $\{a, b, c, e\}$, but r_3 and r_4 are not. Thus, $v(\{a, b, c, e\})$ equals $3 + 2 = 5$.

Next we describe a concise representation of a characteristic function called a *synergy coalition group (SCG)*, introduced by Conitzer and Sandholm [6]. The main idea is to explicitly represent the value of a coalition only when there exists some *positive* synergy.

Definition 2 (*SCG*) An *SCG* consists of a set of pairs of the following form: $(S, v(S))$. For any coalition S, the value of the characteristic function is

$$v(S) = \max \left\{ \sum_{S_i \in p_S} v(S_i) \right\},$$

where p_S is a partition of S, i.e., all the S_i are disjoint and $\bigcup_{S_i \in p_S} S_i = S$, and for all the S_i, $(S_i, v(S_i)) \in SCG$. To avoid senseless cases that have no feasible partitions, we require that $(\{a\}, 0) \in SCG$ whenever $\{a\}$ does not receive a value elsewhere in SCG.

Thus, if the value of coalition S is not given explicitly in SCG, it is calculated from the possible partitions of S. Using this original definition, we can represent only super-additive characteristic functions, i.e., for any disjoint sets $S_i, S_j, v(S_i \cup S_j) \geq v(S_i) + v(S_j)$ holds. But, as mentioned in Sect. 2.1, if the characteristic function is super-additive, solving CSG becomes trivial: the grand coalition is optimal. To allow for characteristic functions that are not super-additive, we add the following requirement on partition p_S:

– For all possible subsets p'_S of partition p_S where $|p'_S| \geq 2$,

$$\left(\bigcup_{S_i \in p'_S} S_i, v \left(\bigcup_{S_i \in p'_S} S_i \right) \right) \notin SCG$$

   holds.

This additional condition requires that if the value of a coalition is explicitly given in SCG, then we cannot further divide it into smaller subcoalitions to calculate the values. In this way, we can represent *negative* synergies.

The (modified) SCG can represent any characteristic function, including characteristic functions that are non-super-additive or even non-monotone. This is because in the worst case, we can explicitly give the value of every coalition. Due to the additional condition, only these explicit values can be used to calculate the characteristic function.

Example 3 Let there be five agents, a, b, c, d, and e, and let $SCG = \{(\{a\}, 0), (\{b\}, 0), (\{c\}, 1), (\{d\}, 2), (\{e\}, 3), (\{a, b\}, 3), (\{a, b, c\}, 3)\}$.

In this case, there exists positive synergy between agents a and b, since $v(\{a, b\}) = 3 > v(a) + v(b) = 0 + 0 = 0$. On the other hand, there exists negative synergy among agents a, b, and c. This is because $v(\{a, b, c\}) = 3 < v(\{a, b\}) + v(\{c\}) = 3 + 1 = 4$, which does not satisfy super-additivity. Thus, we cannot divide $\{a, b, c\}$ into any subcoalitions to calculate $v\{a, b, c\}$. Furthermore, when we calculate the value of a coalition including agents a, b, and c, we use the value of $\{a, b, c\}$.

For $S = \{a, b, c, d, e\}$, the value of S is calculated by $v(\{a, b, c\}) + v(\{d\}) + v(\{e\}) = 3 + 2 + 3 = 8$.

Finally, we introduce the concept of a *multi-issue domain* [5]. In a multi-issue domain, there are k independent issues. The overall value of a coalition is the sum of the values of the coalition for individual issues. More specifically, we assume k characteristic functions $v_1, v_2, \ldots, v_k$ such that for any $S \subseteq A$, $v(S) = \sum_{i=1}^{k} v_i(S)$. If each v_i can be represented concisely, then this leads to a concise representation for v. In this paper, we assume that v_i is represented by SCG_i.

Definition 3 (*SCGs in multi-issue domains*) We represent the characteristic function by a vector of $SCGs$ $(SCG_1, \ldots, SCG_k)$. For any $S \subseteq A$, $v(S) = \sum_{i=1}^{k} v_i(S)$, where v_i is calculated using SCG_i. Also, for coalition structure CS, we denote $V_i(CS) = \sum_{S \in CS} v_i(S)$. Thus, $V(CS) = \sum_{i=1}^{k} V_i(CS)$.

Example 4 Assume four agents a, b, c, and d, and two $SCGs$: $SCG_1 = \{(\{a\}, 0), (\{b\}, 0), (\{c\}, 1), (\{d\}, 0), (\{a, b\}, 2), (\{a, b, c\}, 2)\}$, $SCG_2 = \{(\{a\}, 0), (\{b\}, 0), (\{c\}, 0), (\{d\}, 1), (\{a, b, c\}, 2)\}$. In this case, $v(\{a, b, c\})$ is $v_1(\{a, b, c\}) + v_2(\{a, b, c\}) = 2 + 2 = 4$.

2.3 Partition function games

When externalities exist among coalitions, the value of a coalition depends on the coalition structure to which it belongs. An *embedded coalition* is a pair (S, CS), where $S \in CS \in \Pi(A)$. Denote the set of all embedded coalitions as M, i.e., $M := \{(S, CS) : CS \in \Pi(A), S \in CS\}$. A partition function is mapping $w : M \to \mathbb{R}$.

Example 5 Assume four agents, a, b, c, and d. A partition function is given as follows:

$$
\begin{array}{ll}
w(\{a\}, \{\{a\}, \{b\}, \{c\}, \{d\}\}) = 1, & w(\{b\}, \{\{a\}, \{b\}, \{c\}, \{d\}\}) = 1, \\
w(\{c\}, \{\{a\}, \{b\}, \{c\}, \{d\}\}) = 1, & w(\{d\}, \{\{a\}, \{b\}, \{c\}, \{d\}\}) = 3, \\
w(\{a, b\}, \{\{a, b\}, \{c\}, \{d\}\}) = 3, & w(\{c\}, \{\{a, b\}, \{c\}, \{d\}\}) = 1, \\
w(\{d\}, \{\{a, b\}, \{c\}, \{d\}\}) = 1, & w(\{a\}, \{\{a\}, \{b, d\}, \{c\}\}) = 1, \\
w(\{b, d\}, \{\{a\}, \{b, d\}, \{c\}\}) = 1, & w(\{c\}, \{\{a\}, \{b, d\}, \{c\}\}) = 1, \\
w(\{a\}, \{\{a\}, \{b\}, \{c, d\}\}) = 1, & w(\{b\}, \{\{a\}, \{b\}, \{c, d\}\}) = 1, \\
w(\{c, d\}, \{\{a\}, \{b\}, \{c, d\}\}) = 4, & w(\{a, c\}, \{\{a, c\}, \{b\}, \{d\}\}) = 2, \\
w(\{b\}, \{\{a, c\}, \{b\}, \{d\}\}) = 1, & w(\{d\}, \{\{a, c\}, \{b\}, \{d\}\}) = 3, \\
w(\{a\}, \{\{a\}, \{b, c\}, \{d\}\}) = 1, & w(\{b, c\}, \{\{a\}, \{b, c\}, \{d\}\}) = 2, \\
w(\{d\}, \{\{a\}, \{b, c\}, \{d\}\}) = 3, & w(\{a, d\}, \{\{a, d\}, \{b\}, \{c\}\}) = 1, \\
w(\{b\}, \{\{a, d\}, \{b\}, \{c\}\}) = 1, & w(\{c\}, \{\{a, d\}, \{b\}, \{c\}\}) = 1, \\
w(\{a\}, \{\{a\}, \{b, c, d\}\}) = 1, & w(\{b, c, d\}, \{\{a\}, \{b, c, d\}\}) = 2, \\
w(\{b\}, \{\{b\}, \{a, c, d\}\}) = 1, & w(\{a, c, d\}, \{\{b\}, \{a, c, d\}\}) = 2, \\
w(\{c\}, \{\{c\}, \{a, b, d\}\}) = 1, & w(\{a, b, d\}, \{\{a\}, \{a, b, d\}\}) = 3, \\
w(\{d\}, \{\{d\}, \{a, b, c\}\}) = 1, & w(\{a, b, c\}, \{\{d\}, \{a, b, c\}\}) = 4, \\
w(\{a, b\}, \{\{a, b\}, \{c, d\}\}) = 3, & w(\{c, d\}, \{\{a, b\}, \{c, d\}\}) = 2, \\
w(\{a, c\}, \{\{a, c\}, \{b, d\}\}) = 2, & w(\{b, d\}, \{\{a, c\}, \{b, d\}\}) = 1, \\
w(\{a, d\}, \{\{a, d\}, \{b, c\}\}) = 1, & w(\{b, c\}, \{\{a, d\}, \{b, c\}\}) = 2, \\
w(\{a, b, c, d\}, \{\{a, b, c, d\}\}) = 4. &
\end{array}
$$

In this case, there are 4 optimal CSs. For example, $\{\{a\}, \{b\}, \{c, d\}\}$ is one optimal CS, and the value of this CS is 6.

The game defined in Example 5 has externalities. In particular, the value of $\{d\}$ in $\{\{a\}, \{b\}, \{c\}, \{d\}\}$ is 3, whereas in $\{\{c\}, \{d\}, \{ab\}\}$ it is 1. This means that the formation of coalition $\{a, b\}$ induced a negative externality of 2 on $\{d\}$.

Michalak et al. [20] proposed a concise representation of a partition function called *embedded MC-nets*, which is an extension of MC-nets.

Definition 4 (*Embedded MC-nets*) An *embedded MC-nets* consists of set of embedded rules ER. Each embedded rule $er \in ER$ has the following form: $(L_1)|(L_2)$, $\ldots, (L_l) \rightarrow v_{er}$, where each $L_1, L_2, \ldots, L_l$ is a conjunction of literals over A. L_1, which we call the *internal* condition, is the condition that must be satisfied in the coalition that receives the value. $L_2, \ldots, L_l$, which we call the *external conditions*, must be satisfied in other coalitions. We say that embedded rule er is applicable to coalition S in CS if L_1 is applicable to S and each $L_2, \ldots, L_l$ is applicable to some coalition $S' \in CS \setminus \{S\}$. For coalition S, $w(S, CS)$ is given as $\sum_{er \in ER_{(S,CS)}} v_{er}$, where $ER_{(S,CS)}$ is the set of embedded rules applicable to S in CS.

Note that for an embedded rule, there exists an implicit constraint such that external conditions must be satisfied in coalitions $CS \setminus \{S\}$. By adding each positive literal in internal condition L_1 to the negative literals of all external conditions $L_2, \ldots, L_l$ as well as by adding each positive literal in external conditions $L_2, \ldots, L_l$ to the negative literals of internal condition L_1, we can explicitly represent this implicit constraint. We say an embedded rule is in an *explicit form* if the above condition is satisfied. For example, if an original rule is $(a)|(b), (c) \rightarrow v$, its explicit form is $(a \wedge \neg b \wedge \neg c)|(b \wedge \neg a), (c \wedge \neg a) \rightarrow v$. For simplicity, in the rest of this paper, we assume each embedded rule is in an explicit form.

Example 6 Assume the following rules. Here, er_1 is an embedded rule.

$$r_1 : (a) \rightarrow 1, \qquad r_2 : (b) \rightarrow 1,$$
$$r_3 : (c) \rightarrow 1, \qquad r_4 : (d \wedge \neg a \wedge \neg b) \rightarrow 3,$$
$$r_5 : (a \wedge b) \rightarrow 1, \quad er_1 : (d \wedge \neg a \wedge \neg b)|(a \wedge b \wedge \neg d) \rightarrow -2.$$

If $CS = \{\{a, b\}, \{c\}, \{d\}\}$, all the rules are applicable. Thus, $V(CS) = 1+1+1+3+1-2 = 5$.

3 MIP formulations of coalition structure generation

In this section, we consider coalition structure generation problems, assuming that a characteristic function is given using one of the concise representations introduced in the previous section. For each concise representation, we develop MIP formulations to solve the CSG problem. We also analyze the computational complexity and show that finding an optimal coalition structure is NP-hard.

3.1 Marginal contribution nets

We consider CSG problems when a characteristic function is given using MC-nets representations.

3.1.1 Difficulty of handling negative rules

In MC-nets representations, a set of rules corresponds to a coalition structure if each rule in the set is applicable to some coalition in that coalition structure. Thus, if all rules are positive, we can solve CSG problems by solving a reward maximization problem among the rules. However, as shown in the following example, when there exist negative rules handling the negative value rules for CSG problems is a challenging issue.

Example 7 Assume three agents, a, b, and c, and three rules: $r_1 : (a \wedge b) \rightarrow 3$, $r_2 : (b \wedge c \wedge \neg a) \rightarrow 2$, $r_3 : (a \wedge \neg c) \rightarrow -3$. In this game, $R' = \{r_1, r_2\}$ maximizes $\sum_{r \in R'} v_r$ and gives $2 + 3 = 5$. R' is applicable to $\{\{a\}, \{b, c\}\}$, and there is no such coalition structure other than $\{\{a\}, \{b, c\}\}$. However, the correct value of this coalition structure is not 5 but $2 + 3 - 3 = 2$ since r_3 is also applicable to coalition $\{a\}$. In this case, The correct optimal coalition structure is $\{\{a, c\}, \{b\}\}$, whose value is 3 and a corresponding rule set is $\{r_2\}$.

In general, a negative reward in a reward maximization problem is a pest. When all rules have positive values, choosing a rule never hurts. Thus, we can solve CSG problems by a solver that tries to choose as many rules as possible under some constraints, which only specify the conditions where rules cannot be selected at the same time. If we simply include a negative value rule, the solver just ignores this rule if it is allowed to do so, since choosing it *hurts*. We must describe the condition under which the solver is forced to choose this negative value rule as a result of choosing several other positive value rules. Since such a condition involves the interaction among multiple rules, it can be quite complicated and difficult to handle efficiently.

To handle negative value rules, we introduce a full transformation approach and a dummy rules approach. We show that the former approach is not scalable and that we can encode the problem as a MIP formulation with the latter approach.

3.1.2 Full transformation approach

One might think that handling negative value rules is unnecessary, since every characteristic function can be represented by only positive value rules as long as no coalition has a negative value. Thus, we introduce an algorithm that we call a full transformation algorithm. We assume that R is divided into two groups: sets of positive value rules R_+ and negative value rules R_-.

Definition 5 (*Full transformation algorithm*) The full transformation algorithm is defined as follows:

1. Set $R'_- = R_-$, $R'_+ = R_+$.
2. If $R'_- = \emptyset$, return R'_+.
3. Remove one rule $r_x : (L_x) \rightarrow -v_x$ from R'_-.
4. Remove one rule $r_i : (L_i) \rightarrow v_i$ from R'_+, such that $L_x \wedge L_i \not\models \bot$. If no such rule exists, return failure.
5. If $\neg L_x \wedge L_i \not\models \bot$, create a set of basic rules that is the transformation of non-basic rule $(\neg L_x \wedge L_i) \rightarrow v_i$. Add them to R'_+.
6. Create new basic rule $(L_x \wedge L_i) \rightarrow v_i - v_x$. If $v_i - v_x > 0$, add this rule to R'_+. If $v_i - v_x < 0$, add it to R'_-.
7. If $L_x \wedge \neg L_i \not\models \bot$, create a set of basic rules that is the transformation of non-basic rule $(L_x \wedge \neg L_i) \rightarrow -v_x$. Add them to R'_-. Go to 2.

Let us explain the basic ideas of this algorithm. Since we assume that $\forall S, v(S) \geq 0$ holds, if negative value rule $r_x : (L_x) \rightarrow -v_x$ is applicable to coalition S, there exists at least one positive value rule $r_i : (L_i) \rightarrow v_i$, which is also applicable to S. In other words, r_i can partially eliminate the effect of r_x. We transform r_x and r_i into the following three rules:

r'_1: $(\neg L_x \wedge L_i) \rightarrow v_i$, which is added in Step 5,
r'_2: $(L_x \wedge L_i) \rightarrow v_i - v_x$, which is added in Step 6, and
r'_3: $(L_x \wedge \neg L_i) \rightarrow -v_x$, which is added in Step 7.

It is obvious that the two original rules, r_x and r_i, and these three rules are equivalent. Since r'_1 and r'_3 are non-basic, they must be transformed into multiple basic rules.

Using negative value rules can reduce the efforts for describing a characteristic function. In fact, the representation size might significantly increase when we describe a characteristic function by only positive value rules. We here provide an upper bound of the number of transformed rules, although the details of results related to the naïve approach are explained in the Appendix. Consider l agents that are involved in negative value rules in a MC-net. Since $l - 1$ dummy rules are created to handle the agents, the number does not exponentially increase. The naïve transformation generates an exponential number of rules with respect to the number of original rules.

Before proceeding to the analysis, we restrict our attention to a set of rules that we consider a *minimum rule set*.

Definition 6 (*Minimum rule set*) Set of rules M is a minimum rule set if

- the value of each coalition represented by M is non-negative,
- M has at least one negative value rule,
- if any positive value rule is excluded from M, the remaining set of rules is not a minimum rule set, and
- M is not divided into multiple disjoint minimum rule sets.

From the definition, an arbitrary set of rules M that is not minimum is divided into some disjoint minimum rule set and some positive value rules (that are not minimum). We divide the disjoint minimum rule sets into a collection of rule sets, each of which is a minimum rule set, transform the negative value rules therein into positive value rules, and obtain a rule set with only positive value rules, in conjunction with non-minimum positive value rules, which is equivalent to M.

Let us first consider rule set M with one positive value rule r^+ and one negative value rule r^-. Assume that there exist n agents and define the rules as follows:

$$r^+ : (L^+) \rightarrow v^+$$
$$r^- : (L^-) \rightarrow -v^-.$$

We show that the upper bound of the number of transformed rules is $O(n)$. We classify the possible coalitions with n agents into those to which each combination of the rules is applicable. In this case, we need to consider two kinds of coalitions: one to which r^+ and r^- are applicable, and another to which only r^+ is applicable. Let D_1 denote the former set of coalitions and let D_2 denote the latter set. The conditions and values are described as

$$D_1 : (L^+ \wedge L^-) \quad \rightarrow v^+ - v^-,$$
$$D_2 : (L^+ \wedge \neg L^-) \rightarrow v^+.$$

It is clear that we require only a single positive value rule, which is applicable to each element of D_1. In contrast, since the condition for D_2, i.e., $(L^+ \wedge \neg L^-)$, includes the negation of L^-, it is the disjunction of the conditions. We need to transform it into the conjunction of the disjoint ones. For example, assume that L^- is $(a \wedge b \wedge \neg c)$. We then divide negation $\neg L^- = (\neg a \vee \neg b \vee c)$ into three disjoint conditions, $(\neg a)$, $(a \wedge \neg b)$, and $(a \wedge b \wedge c)$. How many conditions we require depends on the number of literals contained in $\neg L^-$. Since $\neg L^-$ involves all the agents in the worst case, it is divided into n conditions. Thus, we require $O(n)$ rules to create a rule set with only positive value rules, which is equivalent to M.

Next, consider general rule set M with k positive value rules and s negative value rules. Again assume that there exist n agents and define the rules as follows:

$$r_i^+ : (L_i^+) \to v_i^+ \ (1 \le i \le k)$$
$$r_j^- : (L_j^-) \to -v_j^- \ (1 \le j \le s).$$

Next we show that the upper bound of the number of transformed rules is $O(2^{(k+s)} \cdot n^{(k+s-1)})$. We classify the possible coalitions with n agents into those to which each combination of the rules is applicable. In this case, the number of their collections is 2^{k+s}. However, since the values of any coalition are non-negative based on the assumption, we need to exclude 2^s cases where no positive value rules are applied. Thus, we consider $2^{k+s} - 2^s$ kinds of coalitions:

$$D_1 : (L_1^+ \wedge \cdots \wedge L_k^+ \wedge L_1^- \wedge L_2^- \wedge \cdots \wedge L_{s-2}^- \wedge L_{s-1}^- \wedge L_s^-)$$
$$\to v_1^+ + \cdots + v_k^+ - v_1^- - v_2^- - \cdots - v_{s-2}^- - v_{s-1}^- - v_s^-$$
$$D_2 : (L_1^+ \wedge \cdots \wedge L_k^+ \wedge L_1^- \wedge \cdots \wedge L_{s-2}^- \wedge L_{s-1}^- \wedge \neg L_s^-)$$
$$\to v_1^+ + \cdots + v_k^+ - v_1^- - \cdots - v_{s-2}^- - v_{s-1}^-$$
$$D_3 : (L_1^+ \wedge \cdots \wedge L_k^+ \wedge L_1^- \wedge \cdots \wedge L_{s-2}^- \wedge \neg L_{s-1}^- \wedge L_s^-)$$
$$\to v_1^+ + \cdots + v_k^+ - v_1^- - \cdots - v_{s-2}^- - v_s^-$$
$$D_4 : (L_1^+ \wedge \cdots \wedge L_k^+ \wedge L_1^- \wedge \cdots \wedge L_{s-2}^- \wedge \neg L_{s-1}^- \wedge \neg L_s^-)$$
$$\to v_1^+ + \cdots + v_k^+ - v_1^- - \cdots - v_{s-2}^-$$
$$\cdots$$
$$D_{(2^{k+s}-2^s)} : (\neg L_1^+ \wedge \cdots \neg L_{k-1}^+ \wedge L_k^+ \wedge \neg L_1^- \wedge \cdots \wedge \neg L_s^-)$$
$$\to v_k^+.$$

Consider collection of coalitions D_i $(1 \le i \le 2^{k+s} - 2^s)$. As well as the case with one positive and one negative value rule, the number of required conditions depends on how many negations are involved in the conditions of D_i. Thus, we require $O(n^{k+s-1})$ rules to create a positive value rule set, which is equivalent to the conditions of D_i. For all D_i, we require $O(2^{k+s} \cdot n^{k+s-1})$ to create a positive value rule set, which is equivalent to M. Accordingly, this naive transformation generates an exponential number of rules with respect to k and s in the worst case. Our approach with dummy rules which we introduce later is computationally more tractable.

3.1.3 Dummy rules approach

Instead of transforming negative value rules into positive value rules, we introduce the idea of using dummy rules as another way of handling negative value rules that is more concise and efficient. Before introducing dummy rules, we define a feasible set to represent a coalition structure by a set of rules.

Definition 7 (*Feasible rule set*) We say set of rules $R' \subseteq R$ is *feasible* if there exists a CS, where each rule $r \in R'$ is applicable to some $S \in CS$ and $\forall r_- \in R_- \setminus R', r_-$ is not applicable to any $S \in CS$.

Clearly, for each coalition structure CS, there exists at least one feasible rule set $R' \in R$ such that R' is applicable to CS and $V(CS) = \sum_{r \in R'} v_r$ holds. Thus, the problem of finding CS^* is equivalent to finding feasible rule set R' to maximize $\sum_{r \in R'} v_r$.

In Example 2, $\{r_2, r_4\}$ is feasible because it is applicable to $\{\{a, b, c\}, \{d, e\}\}$ and the value of each rule is not negative. $R' = \{r_1, r_2\}$ is also feasible because R' is applicable to $\{\{a, b, c, e\}\{d\}\}$. Since R' maximizes $\sum_{r \in R'} v_r$, $\{\{a, b, c, e\}\{d\}\}$ is the optimal coalition structure, whose value is 5. On the other hand, $\{r_1, r_2, r_4\}$ and $\{r_2, r_3\}$ are infeasible because there is no coalition structure where all the sets of rules are applicable. Let us consider another example that contains negative value rules.

We add dummy rules to directly encode the problem as a MIP formulation as follows.

Definition 8 (*Dummy rules (for basic rules)*) Assume there exists negative value rule r_x : $(L_x) \to -v_x$ $(v_x > 0)$, where $L_x = \bigwedge_{a_i \in P_x} a_i \wedge \bigwedge_{a_j \in N_x} \neg a_j$, $P_x = \{a_1, a_2, \ldots a_k\}$, $N_x = \{a_{k+1}, a_{k+2}, \ldots, a_m\}$. Dummy rules generated by this negative value rule are of the following two types:

(i) $(a_1 \wedge \neg a_i) \to 0$, where $a_i \in P_x \setminus \{a_1\}$,
(ii) $(a_1 \wedge a_j) \to 0$, where $a_j \in N_x$.

We denote $D(L_x)$ as a set of dummy rules created from L_x.

Theorem 1 *A negative value rule is applicable to a coalition in coalition structure CS if and only if none of its dummy rules are applicable to any coalition in CS.*

Proof The condition of a dummy rule can be either $a_1 \wedge \neg a_i$ or $a_1 \wedge a_j$. In either case, it is clear that when this dummy rule is applicable to a coalition in CS, the negative value rule is not applicable to any coalition in CS. Also, if all dummy rules are inapplicable to any coalition in CS, it means that $a_1, a_2, \ldots, a_k$ are in identical coalition S, while $a_{k+1}, a_{k+2}, \ldots, a_m$ are not in S. Thus, the negative value rule is applicable to S.   □

With dummy rules, we can describe the condition where the solver is forced to choose this negative value rule. In brief, we add a constraint where at least one of a negative value rule and the dummy rules created from that rule must be chosen. Note that, from Theorem 1, if no dummy rule created by a negative value rule is chosen, there must exist a coalition such that the negative value rule is applicable, and the solver must choose the rule.

Then we classify the relations between rules to specify the conditions where they cannot be selected at the same time.

Definition 9 (*Relation between rules*) The possible relations between two rules, r and r', can be classified into the following four nonoverlapping and exhaustive cases:
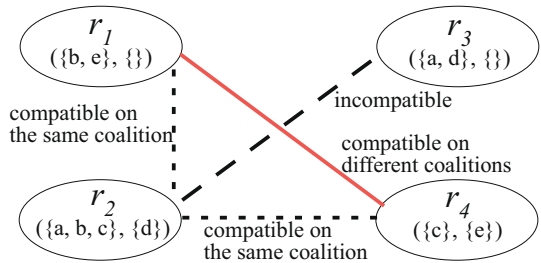
Compatible on the same coalition: $P_r \cap P_{r'} \neq \emptyset$ and $P_r \cap N_{r'} = P_{r'} \cap N_r = \emptyset$. For example, in Example 2, r_1 and r_2 are compatible on the same coalition; if r_1 and r_2 are applicable at the same time, there must be a coalition S with $S \supseteq \{a, b, c, e\}$ and $d \notin S$.
Incompatible: $P_r \cap P_{r'} \neq \emptyset$, and ($P_r \cap N_{r'} \neq \emptyset$ or $P_{r'} \cap N_r \neq \emptyset$). For example, r_2 and r_3 are incompatible; these two rules are not applicable at the same time.
Compatible on different coalitions: $P_r \cap P_{r'} = \emptyset$, and ($P_r \cap N_{r'} \neq \emptyset$ or $P_{r'} \cap N_r \neq \emptyset$). For example, r_1 and r_4 are compatible on different coalitions; if r_1 and r_4 are applicable at the same time, there must be two different coalitions, S_1 and S_2, where $S_1 \supseteq \{b, e\}$ and $S_2 \supseteq \{c\}$.
Independent: $P_r \cap P_{r'} = \emptyset$, and $P_r \cap N_{r'} = P_{r'} \cap N_r = \emptyset$. For example, r_1 and r_3 are independent. These two rules can be applied to the same coalition or to different coalitions.

Let us consider a graphical representation of an MC-net in which each vertex is a rule, and between any two vertices, there exists an edge whose type is one of the four cases

Fig. 1 Graphical representation of Example 2

described above. "compatible on the same coalition", "incompatible", "compatible on different coalitions", or "independent". Figure 1 shows the graphical representation of Example 2 ("independent" edges are not shown).

Definition 10 (*Consistent*) Set of rules R' is consistent if it satisfies the following conditions.

(a) R' includes no pair of rules/vertices connected by an "incompatible" edge, and
(b) if two rules/vertices in R' are connected by a "compatible on different coalitions" edge, then they are not reachable via "compatible on the same coalition" edges within R'.

Consistency guarantees that set of rules R' is applicable to some coalition structure. Let us consider the set of rules in Example 2. Set of rules $\{r_2, r_3\}$ does not satisfy (a) because r_2 and r_3 are connected by an "incompatible" edge. Then let us consider set of rules $\{r_1, r_2, r_4\}$. In this case, r_1 and r_4 are connected by a "compatible on different coalitions" edge but they are reachable via r_2 where both r_1 and r_4 are connected to r_2 by "compatible on the same coalition" edges. Thus, $\{r_1, r_2, r_4\}$ does not satisfy (b). An example of a consistent set of rules is $\{r_1, r_3, r_4\}$, which is applicable to coalition structure $\{\{a, d\}, \{b, e\}, \{c\}\}$.

Definition 11 (*Covering rule set*) Set of rules R' covers all the negative value rules if, $\forall r_- \in R_-$, R' includes either r_- or at least one dummy rule created from r_-.

By using a notion of consistency and a covering rule set, we can characterize feasible rule sets and the following theorems hold.

Theorem 2 *Set of rules R' is applicable to some coalition structure if and only if R' is consistent.*

Proof First, we prove the "if" part. From (a), there exists no incompatible edge within R'. From (b), R' can be divided into groups $G_1, G_2, \ldots, G_k$ where the rules within G_i are reachable from each other by "compatible on the same coalition" edges, there exists no "compatible on different coalitions" edge between the rules in G_i, and there exists no "compatible on the same coalition" edge between rules that belong to different groups.

Let us choose $CS = \{S_1, S_2, \ldots, S_k\}$ so that S_i is the union of all positive literals of $r \in G_i$. Then, for $i \neq j$, $S_i \cap S_j = \emptyset$ holds. This is because $S_i \cap S_j \neq \emptyset$ implies that there exists at least one pair $r \in G_i$, $r' \in G_j$ for which r and r' are connected by a "compatible on the same coalition" edge (since there cannot be an "incompatible" edge between them). But this contradicts the way in which $G_1, \ldots, G_k$ are chosen. Thus, $\{S_1, \ldots, S_k\}$ is a valid coalition structure.[2]

---

[2] If some agent is not included in any S_i, we can assume it forms its own coalition.

Next, we show that for any $r \in G_i$, r is applicable to coalition S_i. Clearly, S_i contains all the positive literals of r. It remains to be shown that S_i does not contain any negative literals of r. For the sake of contradiction, assume S_i contains agent a, where a is a negative literal of r. Then there exists another rule $r' \in G_i$ for which a is a positive literal. There must be a "compatible on different coalitions" or an "incompatible" edge between r and r'. Either case leads to a contradiction. Hence, R' is applicable to CS.

Next, we prove the "only if" part. If R' does not satisfy the above conditions, then there exists no coalition structure where R' is applicable. Clearly, if (a) is not satisfied, i.e., some $r, r' \in R'$ are connected by an "incompatible" edge, then there exists no coalition structure where r and r' are applicable at the same time.

Now, assume (b) is not satisfied, i.e., there exist $r_i, r_j \in R'$ such that r_i and r_j are connected by a "compatible on different coalitions" edge and are reachable by "compatible on the same coalition" edges within R'. Assume r_i is applicable to coalition S_i and r_j is applicable to coalition S_j. Since r_i and r_j are connected by a "compatible on different coalitions" edge, S_i and S_j must be different. However, S_i must contain all of the positive literals of the rules reachable from r_i via "compatible on the same coalition" edges; otherwise, some rule in R' is not applicable. Similarly, S_j must contain all the positive literals of rules reachable from r_j via "compatible on the same coalition" edges. Since r_i and r_j are reachable from each other via "compatible on the same coalition" edges, S_i and S_j must be the same; but this contradicts the fact that they must be different.   □

Theorem 3 *Set of rules R' is feasible if it is consistent and covers all the negative value rules. Furthermore, for any feasible rule set R' (which does not cover all the negative values), there exists another rule set $R''(\supseteq R')$ where $\sum_{r \in R''} v_r = \sum_{r \in R'} v_r$ and R'' is consistent and covers all the negative value rules.*

Proof First, we prove that if R' is consistent and covers all the negative value rules, it is feasible. Since it is consistent, from Theorem 2, there exists coalition structure CS, such that each rule $r \in R'$ is applicable to some $S \in CS$. Thus, to prove that R' is feasible, it suffices to show that $\forall r_- \in R_- \setminus R'$, r_- is not applicable to any $S \in CS$. Since R' covers all the negative value rules, for each negative value rule $r_- \in R_- \setminus R'$, R' contains at least one dummy rule created from r_- and that rule is applicable to some $S \in CS$. Thus, from Theorem 1, $\forall r_- \in R_- \setminus R'$, r_- is not applicable to any $S \in CS$.

Next, we prove that for any feasible rule set R', there exists rule set R'' s.t. $R'' \supseteq R'$, $\sum_{r \in R''} v_r = \sum_{r \in R'} v_r$, and R'' is consistent and covers all the negative value rules. Since R' is a feasible rule set, there exists CS, where each rule $r \in R'$ is applicable to some $S \in CS$ and $\forall r_- \in R_- \setminus R'$, r_- is not applicable to any $S \in CS$. Note that R' is consistent. Now, for each negative value rule $r_- \in R_- \setminus R'$, we show that if R' does not contain any dummy rule of r_-, we can add at least one dummy rule r_d to R' such that r_d is applicable to some coalition in CS, and thus $R' \cup \{r_d\}$ is consistent. We prove this by contradiction; by assuming that for each dummy rule r_d of r_-, r_d is not applicable to any coalition in CS. There exists $S' \in CS$ such that $a_1 \in S'$. From the way dummy rules are created, S' contains all the positive literals of r_-. If this is not the case, i.e., S' does not contain positive literal a_i, then the dummy rule $(a_1 \land \neg a_i) \to 0$ is applicable to S'. Also, S' contains no negative literal of r_-. If this is not the case, i.e., S' contains one negative literal a_j, then the dummy rule $(a_1 \land a_j) \to 0$ is applicable to S'. However, since S' contains all of the positive literals of r_- and no negative literals of r_-, r_- is applicable to S'. This contradicts the assumption that r_- is not applicable to any $S \in CS$. Thus, there exists at least one dummy rule r_d such that r_d is applicable to some coalition in CS, and thus $R' \cup \{r_d\}$ is consistent.

By continuing to add dummy rules to R', we obtain rule set R'' that is consistent and covers all the negative value rules. It is clear that for R'', $\sum_{r \in R''} v_r = \sum_{r \in R'} v_r$ holds since the value of a dummy rule is 0.

Hence, for any feasible rule set R', there exists rule set R'' s.t. $R'' \supseteq R'$, $\sum_{r \in R''} v_r = \sum_{r \in R'} v_r$, and R'' is consistent and covers all the negative value rules. □

From Theorem 3, when considering feasible rule sets, we can restrict our attention to rule sets that are consistent and cover all negative value rules without loss of generality.

Theorem 4 *When the characteristic function is represented as an MC-net, finding an optimal coalition structure is NP-hard. Moreover, unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial-time $O(|R|^{1-\epsilon})$ approximation algorithm for any $\epsilon > 0$, where $|R|$ is the number of rules.*

Proof The maximum independent set problem is to choose $V' \subseteq V$ for a graph $G = (V, E)$ such that there exists no edge between vertices in V', and $|V'|$ is maximized under this constraint. It is NP-hard, and unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial-time $O(|V|^{1-\epsilon})$ approximation algorithm for any $\epsilon > 0$ [13,45]. We reduce an arbitrary maximal independent set instance to a CSG problem instance as follows. For each $v \in V$, let there be agent a_v; also, for each $e \in E$, let there be agent a_e. For each $v \in V$, we create a rule $r_v : (\bigwedge_{a_i \in P_{r_v}} a_i \wedge \bigwedge_{a_j \in N_{r_v}} \neg a_j)$ where $P_{r_v} = \{a_v\} \cup \{a_e : v \in e\}$, $N_{r_v} = \{a_w : (v, w) \in E\}$. Thus, rules are "incompatible" if they correspond to the neighboring vertices and "independent" otherwise. It follows that feasible rule sets correspond exactly to the independent sets of vertices. □

The reduction in Theorem 4 relies heavily on the "incompatibilities" between rules. If there are no "incompatibilities," then the problem is equivalent to the multi-cut problem [41], which is a generalization of the min-cut problem. Note that even without negative value rules, finding an optimal coalition structure using MC-nets is NP-hard.

A CSG using MC-nets can be modeled as finding a rule set that satisfies the condition in Theorem 3 and maximizes the sum of the values.

Definition 12 (*MIP formulation of CSG for MC-nets*) The problem of finding feasible rule set R' that maximizes $\sum_{r \in R'} v_r$ can be modeled as follows:

$$max \ \sum_{r \in R} v_r \cdot x(r)$$

$\quad$ s.t. $\forall e = (r, r')$, where e is an "incompatible" edge,
$$x(r) + x(r') \leq 1, \text{ --- (i)}$$
$\quad \forall e = (r_i, r_j)$, where e is
$\quad$ a "compatible on different coalitions" edge and $i < j$,
$$dis(e, r_i) = 0, dis(e, r_j) \geq 1, \text{ --- (ii)}$$
$\qquad \forall e' = (r_1, r_2)$, where e' is
$\qquad$ a "compatible on the same coalition" edge,
$$dis(e, r_1) \leq dis(e, r_2) + (1 - x(r_1)) + (1 - x(r_2)), \text{ --- (iii)}$$
$$dis(e, r_2) \leq dis(e, r_1) + (1 - x(r_1)) + (1 - x(r_2)), \text{ --- (iv)}$$
$\quad \forall r_- \in R_-$, where $d_1, \ldots, d_k$ are the dummy rules of r_-,
$$x(r_-) + x(d_1) + \cdots + x(d_k) \geq 1, \text{ --- (v)}$$
$\quad \forall r \in R, x(r) \in \{0, 1\}$.

$x(r) = 1$ means that rule r is selected. Constraint (i) ensures that two rules connected by an "incompatible" edge will not be selected at the same time. Also, for each "compatible on different coalitions" edge $e = (r_i, r_j)$, we define a distance/potential for e, so that $dis(e, r_i) =$

0 and $dis(e, r_j) \geq 1$ (ii). Constraints (iii) and (iv) ensure that if both r_1 and r_2 are selected, where r_1 and r_2 are connected by a "compatible on the same coalition" edge, then the distance/potential of these two rules for the aforementioned e must be equal. Then the facts that $dis(e, r_i) = 0$ and $dis(e, r_j) \geq 1$ ensure that r_i and r_j are not reachable from each other via "compatible on the same coalition" edges. Using such a distance/potential is a standard method for representing connectivity constraints in MIP formalization without enumerating possible paths. Constraint (v) ensures that negative value rule r_- or at least one dummy rule created from r_- is selected.

Example 8 Let us show how constraints (ii)–(iv) work with rules of MC-nets in Example 2. Suppose that rules r_1, r_2 and r_4 are selected, i.e., $x(r_1) = x(r_2) = x(r_4) = 1$. Since rules r_1 and r_4 are connected via "compatible on different coalitions" edge $e_{14} = (r_1, r_4)$, $dis(e_{14}, r_1) = 0$ and $dis(e_{14}, r_4) \geq 1$ due to constraint (ii). For "compatible on the same coalition" edge $e_{12} = (r_1, r_2)$, from constraints (iii) and (iv), $dis(e_{14}, r_1) \leq dis(e_{14}, r_2) + (1 - x(r_1)) + (1 - x(r_2)) = dis(e_{14}, r_2) + (1 - 1) + (1 - 1) = dis(e_{14}, r_2)$ and $dis(e_{14}, r_2) \leq dis(e_{14}, r_1) + (1 - x(r_1)) + (1 - x(r_2)) = dis(e_{14}, r_1)$ must be hold. Therefore, we have $dis(e_{14}, r_1) = dis(e_{14}, r_2)$. Similarly, for "compatible on the same coalition" edge $e_{24} = (r_2, r_4)$, we have $dis(e_{14}, r_2) = dis(e_{14}, r_4)$. Thus, we cannot satisfy constraints (ii)–(iv) and select rules r_1, r_2 and r_4 at the same time. Actually, set of rules $\{r_1, r_2, r_4\}$ is not feasible because r_1 and r_4 are connected by a "compatible on different coalitions" edge but they are reachable via "compatible on the same coalition" edges.

In this formulation, the number of binary variables equals the number of all the rules including the dummy rules. The number of constraints is $d_{in} + d_{cd}(2d_{cs} + 1) + |R_-|$, where d_{in}, d_{cd}, d_{cs}, and $|R_-|$ are the number of edges with types "incompatible," "compatible on different coalitions," "compatible on the same coalition," and negative value rules, respectively.

3.1.4 Embedded MC-nets

We extend our method to find an optimal coalition structure when a partition function is represented as an embedded MC-net.

Extending the MIP formulation in Definition 12 to handle embedded MC-nets is rather straightforward. First, we explain how to handle embedded rules whose values are non-negative. For an embedded rule that has form er : $(L_1)|(L_2), \ldots,$ $(L_l) \rightarrow v_{er}$, we create the following basic rules: $r_1 : (L_1) \rightarrow 0, r_2 : (L_2) \rightarrow 0, \ldots,$ $r_l : (L_l) \rightarrow 0$. Assume $x(er), x(r_1), \ldots, x(r_l)$ are 0/1 decision variables in the MIP formulation, i.e., when the value is 1, the rule is selected. An objective function is given by $\sum_{er} v_{er} \cdot x(er)$. Also, we add a constraint where $x(er)$ can be 1 only when all $x(r_1), \ldots, x(r_l)$ are 1. Note that such a constraint is not linear. However, there exists a well-known encoding trick to represent such a non-linear constraint in MIP formulation [3].

Next, we introduce dummy rules to handle negative value embedded rules. For a negative value embedded rule, we create dummy rules from each basic rule obtained by the rule.

Definition 13 (*Dummy rules (for embedded rules)*) Assume there exists negative value embedded rule $r_x : (L_1)|(L_2), \ldots, (L_l) \rightarrow -v_x (v_x > 0)$. Then the dummy rules for r_x are $\bigcup_{L_i} D(L_i)$. Note that $D(L)$ is a set of dummy rules created from L.

Theorem 5 *A negative value embedded rule is applicable to a coalition with coalition structure CS if and only if all of its dummy rules are not applicable to any coalition in CS.*

We omit the proof since it is basically identical to Theorem 1.

Finally, we obtain an extended MIP formulation from Definition 12.

Definition 14 (*MIP formulation of CSG using embedded MC-nets*) The problem of finding an optimal coalition structure can be modeled as follows:

$$max \ \sum v_{er} \cdot x(er)$$

$$\text{s.t.} \forall e = (r, r'), \text{ where } e \text{ is an "incompatible" edge,}$$
$$x(r) + x(r') \leq 1,$$
$$\forall e = (r_i, r_j), \text{ where } e \text{ is}$$
$$\text{a "compatible on different coalitions" edge and } i < j,$$
$$dis(e, r_i) = 0, dis(e, r_j) \geq 1,$$
$$\forall e' = (r_1, r_2), \text{ where } e' \text{ is}$$
$$\text{a "compatible on the same coalition" edge,}$$
$$dis(e, r_1) \leq dis(e, r_2) + (1 - x(r_1)) + (1 - x(r_2)),$$
$$dis(e, r_2) \leq dis(e, r_1) + (1 - x(r_1)) + (1 - x(r_2)),$$
$$\forall er, \text{ where } r_1, r_2, \ldots, r_l \text{ are created from } er,$$
$$x(r_1) + \cdots + x(r_l) \leq l \cdot x(er) + (l - 1 - x(er)), \text{ — (vi)}$$
$$x(er) \leq x(r_1), \ldots, x(er) \leq x(r_l), \text{ — (vii)}$$
$$\forall er_-, \text{ where } er_- \text{ has a negative value and}$$
$$d_1, \ldots, d_k \text{ are the dummy rules of } er_-,$$
$$x(er_-) + x(d_1) + \cdots + x(d_k) \geq 1, \text{ — (viii)}$$
$$\forall r \in R, x(r) \in \{0, 1\}.$$

In the MIP formulation, we add constraints (vi) and (vii) to the MIP formulation in Definition 12 and replace the constraint of the dummy rules as (viii) from (v). Constraints (vi) and (vii) ensure that, for each embedded rule er, er is selected if and only if all of the rules in it are selected. Constraint (viii) ensures that negative value embedded rule er_- or at least one dummy rule created from the rules in er_- is selected.

In this formulation, the number of binary variables equals the sum of the number of all rules including dummy rules and the number of embedded rules. The number of constraints is $d_{in} + d_{cd}(2d_{cs} + 1) + |ER| + |ER_-|$, where $|ER|, |ER_-|$ are the number of embedded rules and negative value rules, respectively.

3.2 Synergy coalition group

In this section, we develop an MIP formulation for finding an optimal coalition structure when a characteristic function is represented as an SCG. We show that when searching for CS^*, we need to consider only the coalitions that are explicitly described in SCG.

Theorem 6 *There exists coalition structure CS for which $V(CS) = V(CS^*)$ and $\forall S \in CS, (S, v(S)) \in SCG$.*

Proof For the sake of contradiction, assume there exists some CS^* so that $V(CS^*)$ is strictly larger than any CS that only consists of elements of SCG. Let us examine coalition $S \in CS^*$ that is not an element of SCG. From the definition of SCG, there exists a partition of S (denoted as p_S) such that $v(S) = \sum_{S_i \in p_S} v(S_i)$, and each S_i is an element of SCG. Then, by replacing each such S by p_S, we obtain a new coalition structure CS that only consists of elements of SCG, and $V(CS) = V(CS^*)$ holds, so we have the desired contradiction. □

Due to Theorem 6, finding CS^* is equivalent to a weighted set packing problem: equivalently to the winner determination problem in combinatorial auctions [30], where each agent is an item and each coalition described in SCG is a bid.

Theorem 7 *When the characteristic function is represented as an SCG, finding an optimal coalition structure is NP-hard. Moreover, unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial-time $O(|SCG|^{1-\epsilon})$ approximation algorithm for any $\epsilon > 0$.*

Proof This follows directly from the corresponding inapproximability for the winner determination problem [30] and the maximum independent set problem [45]. □

Definition 15 (*MIP formulation of CSG for SCG*) The problem of finding CS^* can be modeled as follows:

$$max \sum_{(S, v(S)) \in SCG} v(S) \cdot x(S)$$

$$s.t. \ \forall a \in A, \sum_{S \ni a} x(S) = 1,$$

$$x(S) \in \{0, 1\}.$$

$x(S)$ is 1 if S is included in CS^*, 0 otherwise.

In this formulation (which corresponds to a standard winner determination formulation), the number of binary variables equals $|SCG|$, and the number of constraints equals the number of agents.

3.3 Multi-issue domain

When there are multiple issues, optimal coalition structure CS^* may need to contain a coalition S that is not explicitly described in any SCG_i. For example, assume that in issue i, a and b have a strong positive synergy. Also, in issue j, b and c have strong positive synergy. Then coalition $\{a, b, c\}$ might need to be included in CS^*, even though $\{a, b, c\}$ appears in neither SCG_i nor SCG_j.

Definition 16 (*Value-producing subset*) Given coalition structure CS, we say that SCG_i' (where $SCG_i' \subseteq SCG_i$) is a *value-producing subset* of SCG_i for CS, if SCG_i' consists exactly of the elements of SCG_i that are used to calculate $V_i(CS)$. Thus, $V_i(CS) = \sum_{(S, v_i(S)) \in SCG_i'} v_i(S)$.

In Example 4, $SCG_1' = \{(\{a, b, c\}, 2), (\{d\}, 0)\}$ and $SCG_2' = \{(\{a, b, c\}, 2), (\{d\}, 1)\}$ are value-producing subsets for $CS = \{\{a, b, c\}, \{d\}\}$. From this definition, value-producing subset SCG_i' must contain all the agents, and the elements of SCG_i' must be disjoint. We call a subset that satisfies these conditions a *valid subset*.

Definition 17 (*Valid subset*) $SCG_i' \subseteq SCG_i$ is a valid subset if $\bigcup_{(S, v_i(S)) \in SCG_i'} S = A$, and $\forall(S, v_i(S)), (S', v_i(S')) \in SCG_i'$ where $S \neq S'$, $S \cap S' = \emptyset$ holds.

Theorem 8 *Valid subset $SCG_i' \subseteq SCG_i$ is a value-producing subset of SCG_i for CS if and only if for each $S \in CS$, either one of the following conditions holds:*

1. $(S, v_i(S)) \in SCG_i'$,
2. $\exists p_S$, where p_S is a partition of S, such that $|p_S| \geq 2$, $\forall S' \in p_S$, $(S', v_i(S')) \in SCG_i'$, and $\forall p_S' \subseteq p_S$, where $|p_S'| \geq 2$, $(\bigcup_{S'' \in p_S'} S'', v_i(\bigcup_{S'' \in p_S'} S'')) \notin SCG_i$.

We omit the proof since it is straightforward from the (modified) definition of SCG. Quite interestingly, we can define the possible relations between elements in $SCGs$ in the same way as we did for MC-nets.

Definition 18 (*Relations between coalitions*) The possible relations between two coalitions, $(S, v_i(S)) \in SCG_i$ and $(S', v_j(S')) \in SCG_j$, can be classified into the following four cases, which are nonoverlapping and exhaustive:

Compatible on the same coalition: $i \neq j$ and $S \cap S' \neq \emptyset$. For example, in Example 4, $(\{a, b\}, 2) \in SCG_1$ and $(\{a, b, c\}, 2) \in SCG_2$ are compatible on the same coalition. If these two elements are part of the value-producing subsets at the same time, there must be coalition S with $S \supseteq \{a, b, c\}$.

Incompatible: $i = j$ and $S \cap S' \neq \emptyset$. For example, $(\{a, b\}, 2) \in SCG_1$ and $(\{a, b, c\}, 2) \in SCG_1$ are incompatible; they cannot be used simultaneously.

Compatible on different coalitions: $i = j$, and there exists $(S \cup S', v_i(S \cup S')) \in SCG_i$. Assume that SCG_i contains $(\{a, b\}, 2)$, $(\{c\}, 1)$, and $(\{a, b, c\}, 2)$. For coalition structure CS that contains $\{a, b, c\}$, when SCG_i' has $(\{a, b\}, 2)$ and $(\{c\}, 1)$, it is not a value-producing subsets of SCG_i because the second condition of Theorem 8 is violated. There exists a partition $\{\{a, b\}, \{c\}\}$ such that the joint coalition $\{a, b, c\}$ is included in SCG_i. Therefore, the supersets of $\{a, b\}$ and $\{c\}$ must belong to different coalitions. Otherwise, $(\{a, b, c\}, 2)$ is used to calculate v_i. To be more precise, this relation must be extended to a hyper-edge. If there exists $(S'', v_i(S'')) \in SCG_i$, such that $\forall \hat{S} \in p_{S''}$, $(\hat{S}, v_i(\hat{S})) \in SCG_i$ holds, where $p_{S''}$ is a partition of S'', then we create a hyper-edge that connects the elements in $p_{S''}$. Note that we need to add (hyper-) edges only if the characteristic function is sub-additive for S and S' (if $S \cup S'$ has a value more than $v_i(S) + v_i(S')$, we do not have to create such a hyper-edge.)

Independent: otherwise. For example, $(\{a, b\}, 2) \in SCG_1$ and $(\{d\}, 0) \in SCG_1$ are independent. They can be used in both cases.

The following conditions characterize whether coalitions are value-producing.

Theorem 9 $(SCG_1', \ldots, SCG_k')$, *where each SCG_i' is a valid subset of SCG_i, is a vector of value-producing subsets for some CS if and only if the following conditions hold:*

(a) $(SCG_1', \ldots, SCG_k')$ *include no pair of coalitions connected by an "incompatible" edge, and*

(b) *if a set of coalitions in $(SCG_1', \ldots, SCG_k')$ is connected by a "compatible on different coalitions" hyper-edge, then there exists at least one element that is not reachable from other elements via "compatible on the same coalition" edges.*

We omit the proof since it is basically the same as that of Theorem 3.

Definition 19 (*MIP formulation in multi-issue domains*) The problem of finding value-producing subsets that maximize the summation of values can be modeled as follows:

$max \ \sum_{p=(S, v_*(S)) \in \bigcup_{i=1}^k SCG_i} v_*(S) \cdot x(p)$

s.t. $\forall e = (p, p')$, where e is an "incompatible" edge,

$\quad x(p) + x(p') \leq 1$,

$\quad \forall e = (p_1, p_2, \ldots, p_l)$, where e is

$\quad\quad$ a "compatible on different coalitions" hyper-edge,

$\quad\quad dis(e, p_1) = 0, dis(e, p_2) + \cdots + dis(e, p_l) \geq 1$, — (i)

$\quad\quad \forall e' = (p_i, p_j)$, where e' is

$\quad\quad\quad$ a "compatible on the same coalition" edge,

$\quad\quad\quad dis(e, p_i) \leq dis(e, p_j) + (1 - x(p_i)) + (1 - x(p_j))$,

$\quad\quad\quad dis(e, p_j) \leq dis(e, p_i) + (1 - x(p_i)) + (1 - x(p_j))$,

$\quad \forall p \in \bigcup_{i=1}^k SCG_i, x(p) \in \{0, 1\}$.

$x(p) = 1$ means element p in $\bigcup_{i=1}^{k} SCG_i$ is selected. This formulation is basically the same as Definition 12, except for constraint (i). This constraint means that for hyper-edge e that connects nodes $p_1, p_2, \ldots, p_l$, at least one element must be unreachable. The numbers of variables and constraints are basically the same as in the case of the MC-nets.

Theorem 10 *When the characteristic function is represented as SCGs in a multi-issue domain, finding an optimal coalition structure is NP-hard. Moreover, unless $\mathcal{P} = \mathcal{NP}$, there exists no polynomial-time $O(m^{1-\epsilon})$ approximation algorithm for any $\epsilon > 0$, where m is the number of elements in SCGs.*

Proof We can use the same proof as Theorem 4.   □

4 Evaluation

4.1 Settings

We experimentally evaluate our proposed methods. All of the tests were run on a Core i7-4790 processor with 32GB RAM on a Windows 8.1 Pro Edition. We used CPLEX 12.6.1 for solving the integer programming problem instances.

Michalak et al. [21] report that their ODP-IP algorithm can solve problem instances with 25 agents in less than 100 seconds. We cannot directly compare our results with these results since the CSG formalizations are different. Here, we are not comparing the efficiency of particular algorithms, but checking the scalability of different formalizations. Their algorithm inevitably evaluates all of the possible $O(2^n)$ coalitions. Thus, it is very unlikely that their approaches can scale up to $n = 100$. On the other hand, the advantage of these approaches is that they do not rely on particular representations.

Let us classify problem instances by how many agents are involved in each element of a compact representation, i.e., a rule in MC-nets or a coalition in SCG. We concentrate on three simple and typical cases that are likely to be observed in practice: (i) each element tends to involve a small number of agents; (ii) each element tends to involve a large number of agents; and (iii) there is no bias on the number of agents in each element, that is, each element involves any number of agents with equal probability. Unfortunately, there exist no widely accepted standard benchmark instances for coalition structure generation problems. Thus, in a similar manner to Iwasaki et al. [15], we randomly generate instances using probability distributions, described as follows. To generate problem instances, we choose one of three distributions, decay, normal, and uniform, and determine the number of agents in each element based on the chosen distribution.[3] The instances made by using the decay distribution capture case (i). The normal distribution corresponds to case (ii), and the uniform distribution corresponds to case (iii). They are quite likely to occur in practical situations and are useful to deepen the understanding of the features of our proposed technique, although we admit that this classification is slightly rough.

Let us explain how we construct the base elements for each case. For case (i), using the decay distribution we create elements, e.g., the rules included in MC-nets or the coalitions included in SCG. First, we create a coalition with one randomly chosen agent. Then we

---

[3] The decay distribution generates the number of agent in each element as follows: starting with 1, repeatedly increment the size of the element with probability α until the size is not incremented or the size of the element equals to $|A|$, where $\alpha = 0.55$. For convenience, we partly use the Combinatorial Auction Test Suite [16] to create coalitions with arbitrary distribution.

repeatedly add a new random agent with probability α until an agent is not added or the element includes all the agents, where $\alpha = 0.55$. For case (ii), the size of element $|S|$ is drawn from the normal distribution, and then we randomly add agents to the element so that the number of agents who belong to it equals $|S|$. For case (iii), we use a uniform distribution so that the size of each element $|S|$ is consistent with the uniform distribution over $[1, n]$. Notice that for any of the distributions, the value of each element is drawn from uniform distribution $(0, |S| \times 10)$.

For MC-nets, each of the elements with their values corresponds to each rule r and its value v_r. We apply each element to rule $(\bigwedge_{a \in S} a) \to v_r$ and modify each rule by randomly moving an agent from positive to negative literals with probability $p = 0.2$. For embedded MC-nets, we further repeatedly add a new condition of a rule $(L_1) \to v_e r$ with probability $\beta = 0.15$ until a new one is not added any more. We here create a new condition, i.e., conjunction of literals over A as we construct base elements from each probability distribution. Finally, for both MC-nets and embedded MC-nets, we pick some rule with probability $q = 0.2$ and convert the positive values drawn from $(0, |S| \times 10)$ to negative values $[-|S| \times 10, 0)$. Note that there is no problem instance such that some coalition has a negative value computed from the positive and negative value rules, i.e., in all of the generated problem instances, $v(S) > 0$ holds for all $S \subseteq A$.

For SCG, each of the elements with values corresponds to each coalition and its value. We apply each element to coalition S with value $v(S)$. As we explained, we generate the sizes of coalitions included in SCG based on each probability distribution and specify the values from the uniform distribution $(0, |S| \times 10)$. For SCGs in MID, we create a set of coalitions for each issue that has the identical number of elements. We also fix the number of issues at five.

In our experiments, we fix the number of all agents, which we refer to as $\#agents$, and vary the number of elements in a compact representation. We here refer to the number of rules as $\#rules$ and the number of coalitions as $\#coalitions$. Because we fix $\#agents$, the characteristic (or partition) functions have the same size across $\#rules$ or $\#coalitions$. Thus, the difficulty of each instance is influenced by $\#rules$ or $\#coalitions$.

Through the following experiments, we generate 100 problem instances for each combination of cases and the representations show the performance of the geometric average of the instances. Also, we set the time limit to 10^5 msec; if the runtime of the solver (CPLEX) exceeds this time limit, we terminate the execution and exclude this problem instance when calculating the average runtime.
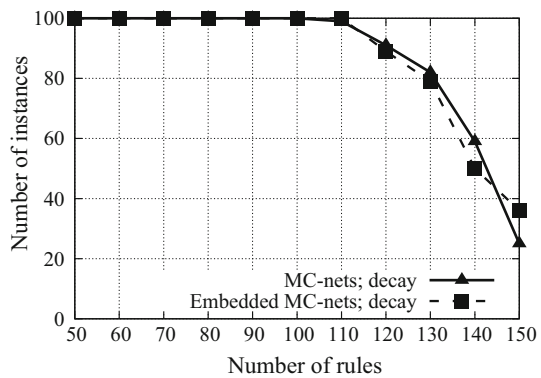
4.2 MC-nets and embedded MC-nets

This subsection explores the performance of the standard and embedded MC-nets. Figures 2 and 4 illustrate the average runtimes of 100 problem instances for each distribution on the y-axis. Figures 3 and 5 show the ratio of instances where the optimal coalition structure is obtained within a time limit of 10^5 msec.

For case (i) where rules are generated from the decay distribution, we set $\#agents = 100$ and vary the number of rules $\#rules$ in the (embedded) MC-nets from 50 to 150 (the x-axis). Figure 2 shows that the runtimes of MC-nets gradually increase and we can handle instances with up to 150 rules. In particular, when $\#rules$ is less than 100, our MIP formulations provide optimal coalition structures within less than 10^4 msec on average and solve every instance within the time limit, as described in Fig. 3. In contrast, when $\#rules$ exceeds 100, some instances cannot be solved within the predetermined time limit. The number of unsolved instances increases in $\#rules$. In fact, when $\#rules = 130$, we can solve 80% (82/100) of
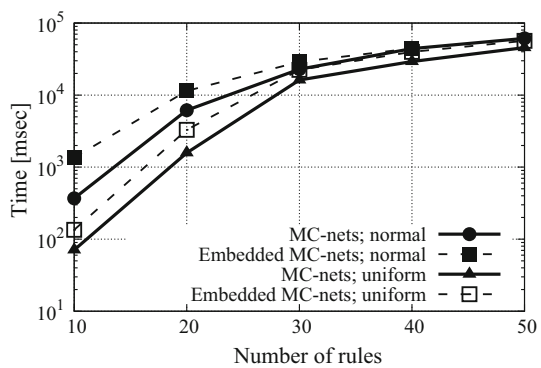
Fig. 2 Average runtimes
(MC-nets and embedded
MC-nets; decay)
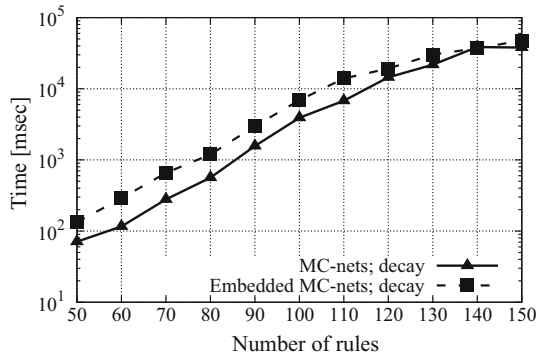
Fig. 3 #Instances solved within
10^5 ms (MC-nets and embedded
MC-nets; decay)

Fig. 4 Average runtimes
(MC-nets and embedded
MC-nets; normal and uniform)

the instances in 10^5 msec, while when #$rules = 150$, we can solve only 25% (25/100) of
them.

Turning to the difference between the standard and embedded MC-nets, it is relatively
small and is magnified when the number of rules increases. When #$rules$ is 100, the differ-
ences are at most 3×10^3 msec across #$rules$, and when it is 150, they reach 10^4 msec. The
magnitude of the differences is affected by the number of constraints in Definition 14, which
is essentially the number of embedded rules. Because we herein assume that a new embedded
rule is added with probability $\beta = 0.15$, only $15 \sim 20$ embedded rules are generated for 100
rules.

Fig. 5 #Instances solved within 10^5 ms (MC-nets and embedded MC-nets; normal and uniform)
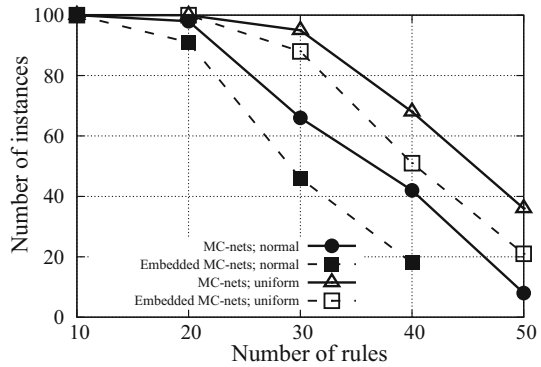

Let us examine cases (ii) and (iii) for the normal and uniform distributions, whose tendencies closely resemble each other. Figure 4 illustrates the runtimes and Fig. 5 shows the ratio of the solvable instances within the time limit. For those cases, we reduce #*agents* from 100 to 10 and vary #*rules* in (embedded) MC-nets from 10 to 50. We discuss why the instance size is rather smaller than the case for the decay distributions later. Note that we here use normal distribution $N(8, 1)$ with a mean of 8 and a variance of 1 from which we draw the number of agents involved in each rule.

Figure 4 shows that our MIP formulations for cases (ii) and (iii) take even more time to obtain the solutions than for case (i), although the instance sizes are rather small. In fact, the upper limit of #*rules*, with which they provide a solution within the time limit, is only 50 for the cases (ii) and (iii), while it reaches 150 for case (i). When #*rules* = 50, case (i) takes 71 msec to obtain the solutions, but case (ii) requires about 61309 msec. Furthermore, Fig. 5 reveals that many instances are not solvable in a reasonable amount of time. Even with #*rules* = 30, for both normal and uniform distributions, 34/100 and 5/100 instances could not be solved within the time limit. Especially for the embedded MC-nets, we can solve no instances for the normal distribution when #*rules* = 50.

Let us briefly discuss why the normal or uniform distribution generates many more difficult instances than the decay distribution. One key reason can be found in the fact that the rules generated from the latter tend to involve fewer agents than those from the former. An arbitrary pair of rules in an instance is less likely to share some agents for the decay distribution than for the normal or uniform distribution. For example, for ten agents, a rule involves approximately three agents for the decay distribution, while one involves approximately eight agents for the normal distribution. Thus, a pair of rules from the decay distribution shares fewer agents than one from the other distributions. Consider our MIP formulations with a set of MC-net rules as a graph. A graph of an instance from the decay distribution is sparser than one from the normal or uniform distribution. In particular, the latter likely constructs a complete graph with many constraints as edges. Therefore, the graphs are much less complicated for case (i) than case (ii) or (iii). To solve cases (ii) or (iii), we need to explore a huge amount of combinations of associated rules.

The other is the sharp increase of the number of the dummy rules. To solve an instance with negative value rules, we must create constraints for each agent involved in each negative value rule. For example, if a rule involves eight agents, we require seven dummy rules for each negative value rule. Since a rule likely involves more agents for the normal or uniform distribution than for the decay distribution, the required number of dummy rules increases, and our MIP formulations face an increasing number of constraints.

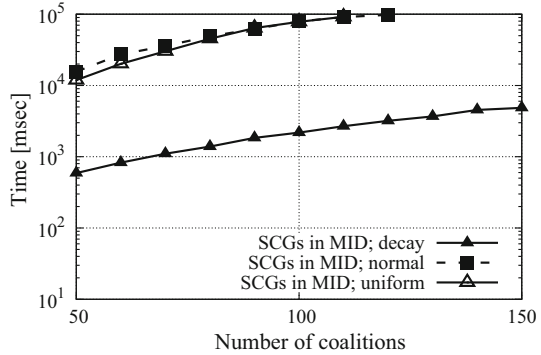**Fig. 6** Average runtimes (SCG; decay, normal, and uniform)
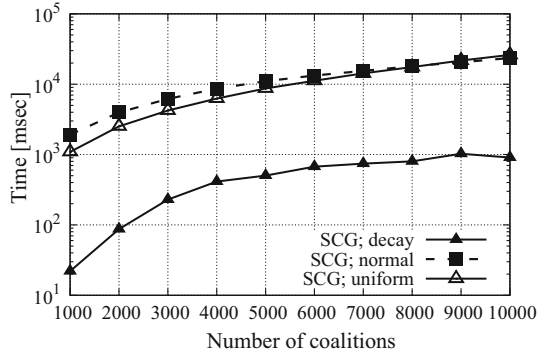
4.3 SCG and SCGs in MID

This subsection evaluates the performance of SCG and SCGs in MID. First, we explain how well we perform on SCG in our settings. Figure 6 illustrates the average runtimes of 100 problem instances for each distribution on the y-axis. We set $\#agent = 1000$ and vary the number of coalitions $\#coalitions$ from 1000 to 10000 (the x-axis). Note that we use normal distribution $N(900, 50^2)$ with a mean of 900 and a variance of 50^2 from which we draw the number of agents involved in each element for case (ii).

Figure 6 shows that our MIP formalization can handle SCG instances with up to 10, 000 coalitions. From this result, we can solve instances with more base elements by applying the MIP formalization based on SCG, compared with the MIP formalization based on MC-nets. For example, for the decay distribution (case (i)), SCG takes 908 msec to obtain the solution on average when it handles 10, 000 coalitions (instances made from 10, 000 base elements), while the standard MC-nets takes 1568 msec when it has only ninety rules (instances made from 90 base elements). For the normal and decay distributions (cases (ii) and (iii)), SCG is still easier to solve than MC-nets, although the runtimes are much longer than for case (i). For example, when $\#coalitions = 10000$, cases (ii) and (iii) are performed in 23574 and 26094 msec, while case (i) takes 908 msec. Also, there is only a slight difference between cases (ii) and (iii), which is at most approximately 3000 msec across the number of coalitions. Note that, in those cases, we can solve all the SCG instances within 10^5 msec (the time limit).

These results show that the MIP formulation of SCG is more scalable than that of MC-nets with regard to the number of base elements. Note that this does not directly means SCG is better than MC-nets. When a game is represented by MC-nets and SCG, MC-nets tends to be more concise. For example, let us consider a game represented by rules of MC-nets R, where each rule $r \in R$ consists of only positive literals. To transform R into SCG, firstly, we need $|R|$ pairs of a coalition and its value, each of which has the form: (P_r, v_r), where P_r is the set of positive literals and v_r is the value of rule r. Next, for each set of rules R' that shares some agents in positive literals, we need a pair $(\bigcup_{r \in R'} P_r, \sum_{r \in R'} v_r)$. For example, assume four agents $a, b, c,$ and d and three rules $r_1 : (a \wedge b) \rightarrow v_1, r_2 : (a \wedge c) \rightarrow v_2, r_3 : (a \wedge d) \rightarrow v_3$. By applying the above argument, SCG of this game is given as: $SCG = \{(\{a, b\}, v_1), (\{a, c\}, v_2), (\{a, d\}, v_3), (\{a, b, c\}, v_1 + v_2), (\{a, b, d\}, v_1 + v_3), (\{a, c, d\}, v_2 + v_3), (\{a, b, c, d\}, v_1 + v_2 + v_3)\}$. In this case, SCG contains 7 pairs, which is larger than $|R|$. If some rule in MC-nets contains negative literals, transforming MC-nets to SCG becomes more complicated.

Fig. 7 Average runtimes (SCGs in MID; decay, normal, and uniform)

Fig. 8 #Instances solved within 10^5 ms (SCGs in MID; decay, normal, and uniform)
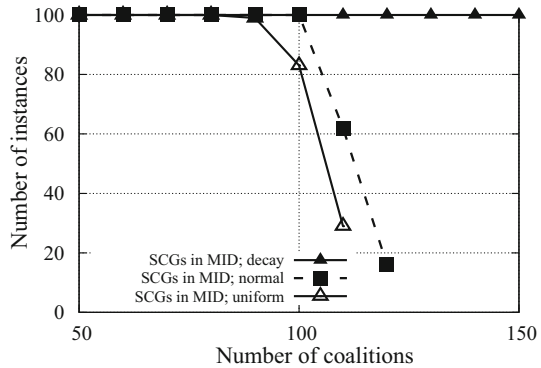
Second, let us turn to SCGs in MID. Figure 7 illustrates the average runtimes and Fig. 8 shows the ratio of instances where we obtain the optimal coalition structures within the time limit of 10^5 msec. Recall that we fix the number of issues to five. We let the sizes of the problem instances be smaller than SCG. Precisely, we set $\#agent = 100$ and vary the number of coalitions $\#coalitions$ from 50 to 150. The mean and variance for the normal distribution remain unchanged.

Figure 7 reveals that SCGs in MID is easier to solve than MC-nets, but harder than SCG. While, for case (i), the runtimes never exceed 10^4 msec across $\#coalitions$, for cases (ii) and (iii), they take at least 10^4 msec for any $\#coalitions$. Also, for case (i), we can solve all the instances within the time limit. However, for the other two cases, we could not solve some instances, particularly when $\#coalitions$ exceeds 90. In fact, for case (ii), when $\#coalitions$ is 120, we can solve only 16/100 instances, while for case (iii), when it is 110, we can solve only 29/100 instances. If we further increase the number of coalitions, we could not solve the instances at all.

5 Conclusion

This paper provides MIP formulations for CSG problems by utilizing four compact representation schemes: for characteristic function games, MC-nets, SCG, and SCGs in MID, and for partition function games, embedded MC-nets. Though we proved that CSG problems under these representations are NP-hard and inapproximable, we could solve instances of significant size by off-the-shelf optimization packages, such as CPLEX and GUROBI. Our simulation reveals that our proposed methods with MC-nets or SCGs in MID solved the problems with 150 rules or coalitions within 10^5 msec, and those with SCG solved the problem with up to 10000 coalitions within 10^5 msec. Future works will develop algorithms (i) that can find an optimal solution more efficiently, (ii) that can return a suboptimal solution in any time, and (iii) that can find an approximate solution quickly by utilizing constraint optimization techniques.

Appendix: Transformation algorithms for MC-nets with negative value rules

In the Apendix, we explain the details of results related to the naïve approach. We can guarantee that the full transformation algorithm terminates, i.e., the following theorem holds.

Theorem 11 *The full transformation algorithm terminates.*

Proof By one iteration of this algorithm, negative value rule r_x is eliminated if $L_x \wedge \neg L_i \models \bot$ and $v_i \geq v_x$. If $L_x \wedge \neg L_i \not\models \bot$, a set of negative value rules is added in Step 7, but the conditions of these rules, i.e., $L_x \wedge \neg L_i$, are more specific than L_x. Also, if $v_i < v_x$, a new negative value rule is added in Step 6, but its condition, i.e., $L_x \wedge L_i$, is more specific than L_x and also disjoint with $L_x \wedge \neg L_i$. Furthermore, the value of this rule, i.e., $v_i - v_x$, is closer to 0 than original value $-v_x$. Thus, by one iteration of this algorithm, the conditions of the negative value rules become more specific and/or the negative value becomes closer to 0. Therefore, this algorithm cannot be infinitely iterated and will eventually terminate. ☐

Example 9 Let us describe the full transformation algorithm, assuming $r_x : (L_x) \rightarrow -1$, where $L_x = a \wedge d \wedge e$, and $r_1 : (L_1) \rightarrow 1$, where $L_1 = a \wedge \neg b \wedge \neg c$, are selected. Since $L_1 \wedge \neg L_x = (a \wedge \neg b \wedge \neg c) \wedge (\neg a \vee \neg d \vee \neg e) \not\models \bot$ holds, we create non-basic rule $(L_1 \wedge \neg L_x) \rightarrow 1$ in Step 5 and we obtain two basic rules from this rule: $(a \wedge \neg b \wedge \neg c \wedge \neg d) \rightarrow 1$ and $(a \wedge \neg b \wedge \neg c \wedge d \wedge \neg e) \rightarrow 1$. We do not create any new rule in Step 6 since $v_{r_1} + v_{r_x} = 1 - 1 = 0$. Finally, since $\neg L_1 \wedge L_x = (\neg a \vee b \vee c) \wedge (a \wedge d \wedge e) \not\models \bot$ holds, we create non-basic rule $(\neg L_1 \wedge L_x) \rightarrow -1$ and we obtain two basic rules from this rule: $(a \wedge b \wedge d \wedge e) \rightarrow -1$ and $(a \wedge \neg b \wedge c \wedge d \wedge e) \rightarrow -1$.

Fig. 9 # Of Generated rules (Example 10)
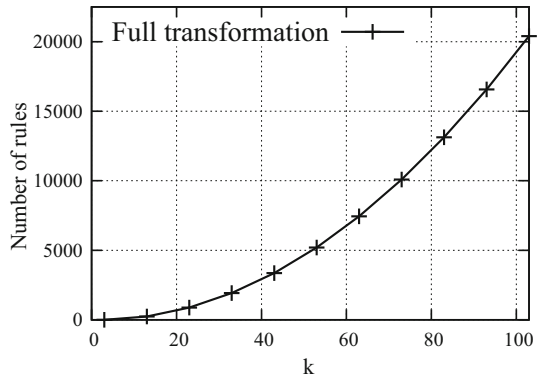
With the full transformation algorithm, we can eliminate all of the negative value rules. However, this approach is not scalable. There exists an instance where the number of newly generated rules becomes $\Omega(n^2)$ using the full transformation algorithm.

Example 10 Consider the following rules:

r_0: $(p_0 \wedge \neg n_1 \wedge \neg n_2 \wedge \cdots \wedge \neg n_k) \to 1$
r_1: $(p_1 \wedge n_1) \to 1$
r_2: $(p_2 \wedge n_2) \to 1$
$\ldots$
r_k: $(p_k \wedge n_k) \to 1$
r_x: $(p_0 \wedge p_1 \wedge p_2 \wedge \ldots \wedge p_k) \to -1.$

This rule set contains $k+1$ positive value rules and one negative value rule, where the total number of agents is $2k+1$. Figure 9 shows the number of newly generated rules from these rule sets by varying k. The number of newly generated rules becomes $\Omega(k^2)$, which is also $\Omega(n^2)$.

Can we reduce the number of required rules using a more clever encoding trick? No, because the following theorem holds:

Theorem 12 *To represent the characteristic function in Example 10 only using positive value rules, we need $\Omega(n^2)$ rules.*

Proof For all $1 \leq i < j \leq k$, we denote $\{p_0, p_1, \ldots, p_k, n_i, n_j\}$ as $S_{i,j}$. For $S_{i,j}$, since only rules r_x, r_i, r_j are applicable, $v(S_{i,j})$ equals 1. Assume that set of positive value rules R'_+ represents v. There must be at least one rule in R'_+ that is applicable to $S_{i,j}$. Represent such a rule as $r_{i,j}$.

Now, we show that $r_{i,j}$ is not applicable to any $S_{i',j'}$, where $1 \leq i' < j' \leq k$ and $i \neq i' \vee j \neq j'$. We derive a contradiction by assuming that $r_{i,j}$ is applicable to $S_{i',j'}$.

When $i = i'$ or $i = j'$, consider coalition $S = \{p_0, p_1, \ldots, p_k, n_i\}$. For S, since only rules r_x, r_i are applicable, $v(S)$ equals 0. However, we show that $r_{i,j}$ is applicable to S, and thus $v(S)$ cannot be 0. $r_{i,j}$ is not applicable to S, if (i) its positive literals include agent n_l, where $l \neq i$, or (ii) its negative literals include at least one of $\{p_0, p_1, \ldots, p_k, n_i\}$. For (i), if $l = j, r_{i,j}$ is not applicable to $S_{i',j'}$. Also, if $l \neq j, r_{i,j}$ is not applicable to $S_{i,j}$. For (ii), $r_{i,j}$ is not applicable to either $S_{i,j}$ and $S_{i',j'}$. This contradicts the assumption that $r_{i,j}$ is applicable to both $S_{i,j}$ and $S_{i',j'}$. We can use a similar argument for the cases where $j = i'$ or $j = j'$.

Then consider the case where i, j, i', j' are different from each other and coalition $S = \{p_0, p_1, \ldots, p_k\}$. For S, since only rules r_x, r_0 are applicable, $v(S)$ equals 0. However, we show that $r_{i,j}$ is applicable to S, and thus $v(S)$ cannot be 0. $r_{i,j}$ is not applicable to S if (i) its positive literals include agent n_l, where $1 \leq l \leq k$, or (ii) its negative literals include at least one of $\{p_0, p_1, \ldots, p_k\}$. For (i), if $l = i$ or $l = j$, $r_{i,j}$ is not applicable to $S_{i',j'}$. If $l \neq i$ and $l \neq j$, $r_{i,j}$ is not applicable to $S_{i,j}$. For (ii), $r_{i,j}$ is not applicable to either $S_{i,j}$ and $S_{i',j'}$. This contradicts the assumption that $r_{i,j}$ is applicable to both $S_{i,j}$ and $S_{i',j'}$.

Thus, for each i, j, where $1 \leq i < j \leq k$, there must be distinct element $r_{i,j}$ in R'_+, and the number of elements in R'_+ must be at least $k(k-1)/2$, which is $\Omega(n^2)$. □

The full transformation algorithm can be easily extended to embedded MC-nets. We replace a condition such as L_i to the condition for embedded rule C_{er}, which is a pair of internal condition L_1 and external conditions $L_2, \ldots, L_l$.

One tricky point is creating the negation of C_{er}. Recall that embedded rule er is applicable to coalition S in CS if L_1 is applicable to S and each $L_2, \ldots, L_l$ is applicable to some coalition $S' \in CS \setminus \{S\}$. Thus, er is not applicable to coalition S in CS if (i) L_1 is not applicable to S, (ii) L_1 is applicable to S, but L_2 is not applicable to *any* coalition in $CS \setminus \{S\}$, (iii) L_1 is applicable to S and L_2 is applicable to some coalition $S' \in CS \setminus \{S\}$, but L_3 is not applicable to *any* coalition in $CS \setminus \{S\}$, and so on. Handling case (i) is easy. Let us examine how to handle case (ii). Assume $L_2 = p_1 \wedge p_2$. We must guarantee that for any coalition $S' \in CS \setminus \{S\}$, $\neg L_2 = \neg p_1 \vee \neg p_2$ holds. If S' does not contain p_1, then $\neg L_2$ holds. If S' contains p_1, then S' must satisfy $\neg p_2$. Since there exists exactly one coalition that contains p_1, it is sufficient to guarantee that there exists some coalition $S' \in CS \setminus \{S\}$, such that $p_1 \wedge \neg p_2$ holds.

To summarize, to represent $\neg C_{er}$, where C_{er} is a pair of internal condition L_0 and external conditions $L_1, \ldots, L_l$, we need the following conditions (here, we assume each $L_i = l_{i_1} \wedge l_{i_2} \wedge l_{i_3} \wedge \ldots$): (i) $(\neg L_0)$, (ii) $(L_0)|(l_{1_1} \wedge \neg l_{1_2})$, $(L_0)|(l_{1_1} \wedge l_{1_2} \wedge \neg l_{1_3})$, …, (iii) $(L_0)|(L_1)(l_{2_1} \wedge \neg l_{2_2})$, $(L_0)|(L_1)(l_{2_1} \wedge l_{2_2} \wedge \neg l_{2_3})$, and so on.

References

1. Aziz, H., & de Keijzer, B. (2011). Complexity of coalition structure generation. In *Proceedings of the 10th international conference on autonomous agents and multiagent systems (AAMAS)* (pp. 191–198).
2. Bistaffa, F., Farinelli, A., Cerquides, J., Rodríguez-Aguilar, J. A., & Ramchurn, S. D. (2014). Anytime coalition structure generation on synergy graphs. In *Proceedings of the 13th international conference on autonomous agents and multi-agent systems (AAMAS)* (pp. 13–20).
3. Castillo, E., Conejo, A. J., Pedregal, P., Garcia, R., & Alguacil, N. (2001). *Building and solving mathematical programming models in engineering and science*. New York: Wiley.
4. Chalkiadakis, G., Elkind, E., & Wooldridge, M. (2011). *Computational aspects of cooperative game theory*. Morgan and Claypool Publishers.
5. Conitzer, V., & Sandholm, T. (2004). Computing shapley values, manipulating value division schemes, and checking core membership in multi-issue domains. In *Proceedings of the 19th national conference on artificial intelligence (AAAI)* (pp. 219–225).
6. Conitzer, V., & Sandholm, T. (2006). Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170(6), 607–619.
7. Dang, V. D., Dash, R. K., Rogers, A., & Jennings, N. R. (2006). Overlapping coalition formation for efficient data fusion in multi-sensor networks. In *Proceedings of the 21st national conference on artificial intelligence (AAAI)* (pp. 635–640).
8. Deng, X., & Papadimitriou, C. H. (1994). On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2), 257–266.
9. Elkind, E., Goldberg, L. A., Goldberg, P. W., & Wooldridge, M. (2009). A tractable and expressive class of marginal contribution nets and its applications. *Mathematical Logic Quarterly*, 55(4), 362–376.
10. Gillies, D. (1953). *Some theorems on n-person games*. Ph.D. thesis, Princeton University.

11. Greco, G., Malizia, E., Palopoli, L., & Scarcello, F. (2011). On the complexity of core, kernel, and bargaining set. *Artificial Intelligence*, *175*(12–13), 1877–1910.
12. Greco, G., Malizia, E., Palopoli, L., & Scarcello, F. (2011). On the complexity of the core over coalition structures. In *Proceedings of the 22nd international joint conference on artificial intelligence (IJCAI)* (pp. 216–221).
13. Håstad, J. (1999). Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, *182*, 105–142.
14. Ieong, S., & Shoham, Y. (2005). Marginal contribution nets: A compact representation scheme for coalitional games. In *Proceedings of the 6th ACM conference on electronic commerce (ACM EC)* (pp. 193–202).
15. Iwasaki, A., Ueda, S., Hashimoto, N., & Yokoo, M. (2015). Finding core for coalition structure utilizing dual solution. *Artificial Intelligence*, *222*, 49–66.
16. Leyton-Brown, K., Pearson, M., & Shoham, Y. (2000). Towards a universal test suite for combinatorial auction algorithms. In *ACM EC* (pp. 66–76).
17. Li, Y., & Conitzer, V. (2014). Complexity of stability-based solution concepts in multi-issue and mc-net cooperative games. In *Proceedings of the 13th international conference on autonomous agents and multi-agent systems (AAMAS)* (pp. 581–588).
18. Liao, X., Koshimura, M., Fujita, H., & Hasegawa, R. (2012). Solving the coalition structure generation problem with maxsat. In *Proceedings of the IEEE 24th international conference on tools with artificial intelligence* (pp. 910–915).
19. Megiddo, N. (1978). Computational complexity of the game theory approach to cost allocation for a tree. *Mathematics of Operations Research*, *3*(3), 189–196.
20. Michalak, T. P., Marciniak, D., Szamotulski, M., Rahwan, T., Wooldridge, M., McBurney, P., et al. (2010). A logic-based representation for coalitional games with externalities. In *Proceedings of the 9th international conference on autonomous agents and multiagent systems (AAMAS)* (pp. 125–132).
21. Michalak, T. P., Rahwan, T., Elkind, E., Wooldridge, M., & Jennings, N. R. (2016). A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence*, *230*, 14–50.
22. Modi, P. J., Shen, W. M., Tambe, M., & Yokoo, M. (2003). An asynchronous complete method for distributed constraint optimization. In *Proceedings of the 2nd international conference on autonomous agents and multiagent systems (AAMAS)* (pp. 161–168).
23. Myerson, R. B. (1977). Values of games in partition function form. *International Journal of Game Theory*, *6*(1), 23–31.
24. Ohta, N., Conitzer, V., Ichimura, R., Sakurai, Y., Iwasaki, A., & Yokoo, M. (2009). Coalition structure generation utilizing compact characteristic function representations. In *Proceedings of the 15th international conference on principles and practice of constraint programming (CP)* (pp. 623–638).
25. Rahwan, T., & Jennings, N. R. (2008). An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th international conference on autonomous agents and multiagent systems (AAMAS)* (pp. 1417–1420).
26. Rahwan, T., Michalak, T. P., Elkind, E., Faliszewski, P., Sroka, J., Wooldridge, M., et al. (2011). Constrained coalition formation. In *AAAI*.
27. Rahwan, T., Michalak, T. P., Jennings, N. R., Wooldridge, M., & McBurney, P. (2009). Coalition structure generation in multi-agent systems with positive and negative externalities. In *Proceedings of the 21st international joint conference on artificial intelligence (IJCAI)* (pp. 257–263).
28. Rahwan, T., Michalak, T. P., Wooldridge, M., & Jennings, N. R. (2015). Coalition structure generation: A survey. *Artificial Intelligence*, *229*, 139–174.
29. Rahwan, T., Ramchurn, S. D., Jennings, N. R., & Giovannucci, A. (2009). An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research (JAIR)*, *34*, 521–567.
30. Sandholm, T. (2002). Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, *135*(1–2), 1–54.
31. Sandholm, T., Larson, K., Andersson, M., Shehory, O., & Tohmé, F. (1999). Coalition structure generation with worst case guarantees. *Artificial Intelligence*, *111*(1–2), 209–238.
32. Sandholm, T., & Lesser, V. R. (1997). Coalitions among computationally bounded agents. *Artificial Intelligence*, *94*(1–2), 99–137.
33. Schmeidler, D. (1969). The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics*, *17*(6), 1163–1170.
34. Shapley, L. S. (1953). A value for n-person games. In *Contributions to the theory of games* (pp. 307–317). Princeton: Princeton University Press.
35. Skibski, O., Michalak, T. P., Sakurai, Y., Wooldridge, M., & Yokoo, M. (2015). A graphical representation for games in partition function form. In *Proceedings of the 29th AAAI conference on artificial intelligence (AAAI)* (pp. 1036–1042).
36. Thrall, R. M., & Lucas, W. F. (1963). N-person games in partition function form. *Naval Research Logistics Quarterly*, *10*(1), 281–298.

37. Tran-Thanh, L., Nguyen, T., Rahwan, T., Rogers, A., & Jennings, N. R. (2013). An efficient vector-based representation for coalitional games. In *Proceedings of the 23rd international joint conference on artificial intelligence (IJCAI)* (pp. 383–389).
38. Ueda, S., Hasegawa, T., Hashimoto, N., Ohta, N., Iwasaki, A., & Yokoo, M. (2012). Handling negative value rules in mc-net-based coalition structure generation In *Proceedings of the 11th international conference on autonomous agents and multiagent systems (AAMAS)* (pp. 795–804).
39. Ueda, S., Iwasaki, A., Yokoo, M., Silaghi, M. C., Hirayama, K.,& Matsui, T. (2010). Coalition structure generation based on distributed constraint optimization. In *Proceedings of the 24th AAAI conference on artificial intelligence (AAAI)* (pp. 197–203).
40. Ueda, S., Kitaki, M., Iwasaki, A., & Yokoo, M. (2011). Concise characteristic function representations in coalitional games based on agent types. In *Proceedings of the 22nd international joint conference on artificial intelligence (IJCAI)* (pp. 393–399).
41. Vazirani, V. V. (2001). *Approximation algorithms*. Berlin: Springer.
42. Voice, T., Polukarov, M., & Jennings, N. R. (2012). Coalition structure generation over graphs. *Journal of Artificial Intelligence Research (JAIR)*, *45*, 165–196.
43. Voice, T., Ramchurn, S. D.,& Jennings, N. R. (2012). On coalition formation with sparse synergies. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems (AAMAS)* (pp. 223–230).
44. Yeh, D. Y. (1986). A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics*, *26*(4), 467–474.
45. Zuckerman, D. (2007). Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, *3*, 103–128.