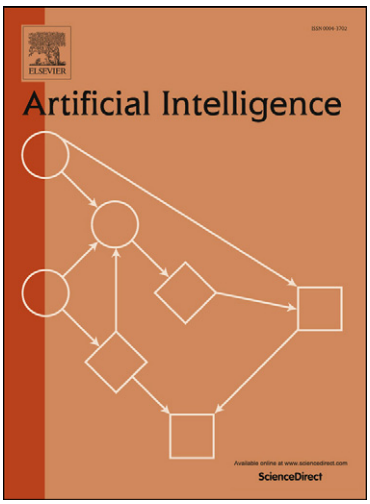
Accepted Manuscript



Coalition structure generation: A survey

Talal Rahwan, Tomasz P. Michalak, Michael Wooldridge, Nicholas R. Jennings

PII: S0004-3702(15)00119-8
DOI: http://dx.doi.org/10.1016/j.artint.2015.08.004
Reference: ARTINT 2876

To appear in: *Artificial Intelligence*

Received date: 3 October 2014
Revised date: 30 July 2015
Accepted date: 13 August 2015

Please cite this article in press as: T. Rahwan et al., Coalition structure generation: A survey, *Artificial Intelligence* (2015), http://dx.doi.org/10.1016/j.artint.2015.08.004

This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

ACCEPTED MANUSCRIPT

Coalition Structure Generation: A Survey[☆]

Talal Rahwan[†a], Tomasz P. Michalak[†b,c], Michael Wooldridge[b], Nicholas R. Jennings[d,e]

[a] *Masdar Institute of Science and Technology, UAE*
[b] *Department of Computer Science, University of Oxford, UK*
[c] *Institute of Informatics, University of Warsaw, Poland*
[d] *School of Electronics & Computer Science, University of Southampton, UK*
[e] *Department of Computing & Information Technology, King Abdulaziz University, Saudi Arabia*

Abstract

The coalition structure generation problem is a natural abstraction of one of the most important challenges in multi-agent systems: How can a number of agents divide themselves into groups in order to improve their performance? More precisely, the coalition structure generation problem focuses on partitioning the set of agents into mutually disjoint coalitions so that the total reward from the resulting coalitions is maximized. This problem is computationally challenging, even under quite restrictive assumptions. This has prompted researchers to develop a range of algorithms and heuristic approaches for solving the problem efficiently. This article presents a survey of these approaches. In particular, it surveys the main dynamic-programming approaches and anytime algorithms developed for coalition structure generation, and considers techniques specifically developed for a range of compact representation schemes for coalitional games. It also considers settings where there are constraints on the coalitions that are allowed to form, as well as settings where the formation of one coalition could influence the performance of other co-existing coalitions.

Keywords: Coalitional Games, Set Partitioning, Coalition Structure Generation

1. Introduction

The multi-agent systems research field is concerned with understanding and building systems containing multiple autonomous software entities (called *agents*) that may have different preferences, goals, beliefs, and capabilities [85]. One of the key objectives of the multi-agent systems domain is to build agents that can take joint, coordinated actions, for example to improve their performance, or to achieve goals that are beyond the capabilities of individual agents. Such interaction may be useful both in cases where the agents are *cooperative* (i.e., their goal is to maximize some overarching system-wide objective) as well as cases where they are *selfish* (i.e., each agent acts in its own best interests, regardless of the consequences on other agents).

The way the agents are organized in a system influences, or even governs, their interaction. For example, if the agents are organized hierarchically, then an agent would only be able to coordinate with its parent and/or children in the hierarchy. Clearly, there are many alternative organizational paradigms (not just hierarchies), each with their own strengths and weaknesses [32]. One such paradigm, which has received much attention in the literature, involves the formation of *coalitions*, i.e., groups of agents that typically exhibit the following characteristics: Firstly, they are goal-directed and short-lived; they are formed with a

[☆]Tomasz P. Michalak and Michael Wooldridge were supported by the European Research Council under Advanced Grant 291528 ("RACE"). Nicholas R. Jennings was supported by the ORCHID Project, funded by an EPSRC (Engineering and Physical Research Council) under the grant EP/I011587/1.

[†]Talal Rahwan and Tomasz Michalak contributed equally to this article.

ACCEPTED MANUSCRIPT

purpose in mind and dissolve when that purpose no longer exists, or when they cease to suit that purpose. Secondly, coordination occurs among members of the same coalition, but not among members of different coalitions. Thirdly, the organizational structure within each coalition is usually flat (rather than hierarchical, for example).

A wide range of potential applications of coalition formation have been considered in the literature. For example, by forming coalitions: autonomous sensors can improve their surveillance of certain areas [31]; virtual power plants can reduce the uncertainty of their expected energy output [13]; cognitive radio networks can increase their throughput [38]; and buyers can obtain lower prices through bulk purchasing [39]. It should be noted, however, that the coalition structure generation problem, being an abstraction, only serves as a first step towards understanding and building real-world solutions (as is the case with many research topics in cooperative game theory, where the focus on abstraction means that the solutions disregard much domain-specific information that may be critical when solving real-world problems).

Generally speaking, the coalition-formation process involves three main activities [78]:

- *Forming a coalition structure.* This activity involves each agent joining a coalition.[1] This is done either endogenously (by the agents deciding autonomously among themselves using some bargaining procedure), or exogenously (e.g., by a system designer). The resulting set of coalitions is called a *coalition structure*. Typically, we are interested in the coalition structure that maximizes social welfare, or minimizes the agents' incentive to deviate from their coalitions.

- *Solving the optimization problem of each coalition.* This activity addresses the following question: How should the members of a coalition coordinate their activities such that the performance of the coalition is maximized?

- *Dividing the reward of each coalition among its members.* If the benefits of cooperative action are accrued to a coalition as a whole, then the members of that coalition will need to agree on how to divide these benefits amongst themselves. Typically the goal is to do this in such a way as to satisfy certain desirable criteria, such as *fairness* (where each agent's reward reflects its contribution to the game), or *stability* (where no group of agents can selfishly benefit by forming their own coalition). In this context, a *solution concept* specifies (i) which coalitions to form, and (ii) how the payoff of every formed coalition is divided among its members.

In this article, we focus on the problem of identifying coalition structures that maximize social welfare. While the relevance of this problem is clear when the agents are cooperative, it is perhaps less so when the agents are selfish. The main relevance in the latter setting arises when we need to compute solution concepts that inherently require the agents to be partitioned optimally. This is the case, for instance, with the *core*—one of the key solution concepts in coalitional game theory [24, 29]. According to this scheme, for any division of rewards to be stable, a necessary condition is that the agents form a social welfare-maximizing coalition structure (see, e.g., [16], Proposition 2.21). In other words, computing a stable outcome implies solving the coalition structure generation problem. Other fundamental game theoretic contexts where the solution to this problem may be useful are the *Price of Anarchy* and the *Price of Stability*. Specifically, in the coalitional game context, the Price of Anarchy is defined as the ratio between the *worst* stable[2] coalition structure and the welfare-maximizing coalition structure [35]. Similarly, the Price of Stability is defined as the ratio between the *best* stable coalition structure and the welfare-maximizing coalition structure [3].

[1] Games with overlapping coalitions, where an agent can join multiple coalitions simultaneously, have also been considered. See, e.g., the work by Shehory and Kraus [83] who first studied these settings among cooperative agents, and the work by Chalkiadakis et al. [15] who considered selfish agents.

[2] The word "stable" is used here in its broader sense; it is not restricted to the *core*, but could refer to the Nash equilibrium, for example.

ACCEPTED MANUSCRIPT

Arguably, the first attempts to study the algorithmic aspects of coalition formation in the multi-agent community were those made by Shehory and Kraus [81], by Ketchpel [37], and by Zlotkin and Rosenschein [96]. Prior to these works, the primary focus in the literature was on the theoretical analysis of the properties of various solution concepts, rather than focusing on the development of coalition formation *algorithms*. Several heuristic algorithms were later on proposed by Shehory and Kraus for various settings [82, 83, 84]. Continuing this line of research, the seminal work by Sandholm et al. [78] studied the computational aspects of identifying coalition structures with worst-case guarantees on solution quality. Since then, numerous algorithms have been proposed to solve this problem, using a range of different techniques. In what follows, we present a comprehensive survey of this literature, and discuss a variety of directions from which the coalition structure generation problem has been approached. Careful attention has been given to ensure that the intuitions behind the different algorithms and their underlying theorems are presented in an accessible and clear manner. As such, the reader is not assumed to have expertise in combinatorial optimization or game theory.

The remainder of the paper is structured as follows.

- **Section 2** (*page 4*) introduces our key definitions and notational conventions.
- **Section 3** (*page 6*) presents alternative representations of the search space (i.e., the space of possible coalition structures).
- **Section 4** (*page 7*) presents a dynamic programming algorithm that computes an optimal coalition structure. This is an exact algorithm, but not an anytime algorithm.
- **Section 5** (*page 11*) presents exact, anytime algorithms whose solution quality improves monotonically over time. In particular:
    - **Section 5.1** (*page 11*) presents algorithms that divide the space into subspaces and specify the order in which these subspaces are searched, so that the worst-case guarantee on solution quality improves with each subspace. Those algorithms do not specify how each subspace is searched.
    - **Section 5.2** (*page 15*) presents an algorithm that divides the space into subspaces based on the sizes of the coalitions in each coalition structure. Each promising subspace is then searched in a depth-first manner while applying a branch-and-bound technique.
    - **Section 5.3** (*page 16*) presents dynamic programming-based anytime algorithms.
    - **Section 5.4** (*page 21*) presents the integer programming formulation of the coalition structure generation problem.
- **Section 6** (*page 21*) presents metaheuristic algorithms, which do not provide any guarantees on solution quality, but can handle very large problems.
- **Section 7** (*page 23*) discusses the coalition structure generation problem under various compact representations of the characteristic function. More specifically:
    - **Section 7.1** (*page 23*) focuses on settings where the value of a coalition is the solution to a *distributed constraint optimization problem (DCOP)*.
    - **Section 7.2** (*page 24*) considers *Coalitional Skill Games*, where every agent has a set of skills, and the value of a coalition depends on the skills of its members.
    - **Section 7.3** (*page 27*) focuses on the *agent-type representation*, where some of the agents are identical.
    - **Section 7.4** (*page 29*) discusses the *Synergy Coalition Groups* representation, where only the coalitions with synergy are explicitly modeled.
    - **Section 7.5** (*page 30*) considers the case where the characteristic function is expressed using the *Marginal Contribution net (MC-net)* representation.
- **Section 8** (*page 34*) considers settings where a coalition is allowed to form if it satisfies certain constraints. In more detail:
    - **Section 8.1** (*page 35*) discusses the *Constrained Coalition Formation* framework, where constraints are expressed in terms of acceptable coalition sizes, and in terms of subsets of agents whose presence in a coalition is desirable, and other subsets whose presence is prohibited.

ACCEPTED MANUSCRIPT

- **Section 8.2** (*page 37*) discusses *Graph-restricted Games*, where agents are connected via a graph, and a coalition is feasible if it induces a connected subgraph.

- **Section 9** (*page 40*) focuses on *Partition Function Games (PFGs)*, where the value of a coalition may differ depending on how non-members are partitioned. In particular:
  - **Section 9.1** (*page 40*) introduces *externalities*—a central notion in PFGs.
  - **Section 9.2** (*page 41*) shows how, in certain subclasses of PFGs, it is possible to establish a bound on solution quality by only searching part of the solution space.
- **Section 11** (*page 45*) concludes the article and outlines potential future directions.
- **Appendix A** (*page 50*) summarizes the main notation used throughout the article.

2. Preliminaries and Problem Definition

The basic assumption underpinning the work surveyed in this article is that the agents in the multi-agent system can coordinate their activities, and that there is potentially some benefit to be gained from such collective action. A cooperative game is a natural way to model such a setting.[3] Here, the setting is called a *game*, which is populated by n *players* (or *agents*), the set of which is denoted by $A = \{a_1, a_2, \ldots, a_n\}$, and the term "*coalition*" is used to refer to a non-empty subset of A. Furthermore, a collection of pairwise-disjoint coalitions is called a "*coalition structure*". Formally:

Definition 1. *For any coalition, C, a **coalition structure** over C is a collection of coalitions, $CS = \{C_1, \ldots, C_k\}$, such that $\bigcup CS = C$, and $C_i \cap C_j = \emptyset$ for any $i, j \in \{1, \ldots, k\} : i \neq j$. The set of coalition structures over C will be denoted by Π^C.*

For example, given a set of five agents, $A = \{a_1, a_2, a_3, a_4, a_5\}$, and a coalition, $C = \{a_3, a_4, a_5\}$, a possible coalition structure over A is: $\{\{a_1\}, \{a_2, a_3\}, \{a_4, a_5\}\}$, and a possible coalition structure over C is: $\{\{a_3\}, \{a_4, a_5\}\}$. As is common in the literature, when talking about a coalition structure over some coalition, $C \subseteq A$, we will often omit "over C" if it is clear from the context, and simply write "coalition structure". We will denote an arbitrary coalition in a coalition structure using the notation C, possibly with subscripts (e.g., C_1 or C_i), or with primes (e.g., C' or C''). Next, we introduce the notion of an "embedded coalition":

Definition 2. *An **embedded coalition** is a pair, (C, CS), where C is a coalition, and CS is a coalition structure over A that contains C. That is, $CS \in \Pi^A : C \in CS$. The set of all embedded coalitions will be denoted by EC.*

In some settings modelled by coalitional games, the reward (or *value*) attainable by a coalition can be expressed in monetary terms; the reward can be placed in one big pot, and the agents can then decide on how to distribute it among themselves. Such a setting is called a *transferable utility game*. In contrast, there are settings in which the reward of a coalition is expressed as a vector that specifies the individual reward of every member; an agent cannot alter this allocation, say, by transferring (some of) its reward to other agents. Such a setting is called a *non-transferable utility game* [60]. Between those two classes, transferable utility games have received most attention from the multi-agent systems community to date. Given this, we will restrict our attention to transferable utility games and from now on simply write "game" instead of "coalitional game with transferable utility".

The following are two distinct classes of games that differ in terms of the factors that influence a coalition's value:

[3] We refer the reader to [58] for an introduction to the theory of cooperative games and [16] for a detailed introduction to the use of such models in computer science. See also [85] for a thorough introduction to game theory in multi-agent systems.

ACCEPTED MANUSCRIPT

- **Characteristic function games (CFGs)** are those where the value of a coalition depends solely on the identities of its members. Such a game is given by a pair, (A, v), where A is the set of agents and v is a function—called the *characteristic function*—that maps each coalition, C, to its value, $v(C)$. Formally: $v : 2^A \to \mathbb{R}$.

- **Partition function games (PFGs)** are those where the value of a coalition entirely depends on both the identities of its members as well as the way non-members are partitioned. Such a game is given by a pair, (A, w), where A is the set of agents and w is a function—called the *partition function*—that maps each embedded coalition, (C, CS), to its value, $w((C, CS))$. Formally, $w : EC \to \mathbb{R}$. We will write $w(C, CS)$ instead of $w((C, CS))$ for brevity.

Observe that CFGs form a proper subclass of PFGs. Furthermore, a coalition $C \subseteq A$ has exactly one value in CFGs, while in PFGs it has as many values as there are ways to partition the agents in $A \setminus C$. As such, CFGs tend to be much easier to work with (which is, perhaps, why they have received more attention in the literature on algorithmic game theory). This fact is reflected in this article; all of the remaining Sections deal with CFGs, apart from Section 9, which focuses on PFGs.

A common assumption in the literature is that the worth (or *value*) of a coalition structure is simply the sum of the values of the coalitions in it. More formally, in CFGs, for any coalition, $C \subseteq A$, the value of a coalition structure, $CS \in \Pi^C$, is denoted by $V(CS)$ and is given by:

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

On the other hand, in PFGs, the value of CS is denoted by $W(CS)$ and is given by:

$$W(CS) = \sum_{C_i \in CS} w(C_i, CS).$$

As is standard in the literature (e.g., [78, 19, 67]) we assume that $v(C) \geq 0$ for any $C \subseteq A$, and that $w(C, CS) \geq 0$ for any $(C, CS) \in EC$. Now, we are ready to define the computational problem that we focus on throughout this article.

Definition 3. *The **coalition structure generation problem** is the problem of finding a coalition structure over A whose value is maximal. Such a coalition structure is said to be "optimal", and is denoted by CS^*. More specifically, in CFGs, coalition structure generation is the problem of finding $CS^* \in \arg\max_{CS \in \Pi^A} V(CS)$, while in PFGs it is the problem of finding $CS^* \in \arg\max_{CS \in \Pi^A} W(CS)$.*

This is a combinatorial optimization problem, which in principle could be solved to optimality by brute-force search. However, this is not practicable, as the number of possible coalition structures over n agents—known as *the Bell number* and denoted by B_n [9]—satisfies:

$$\alpha n^{n/2} \leq B_n \leq n^n.$$

for some positive constant α (see, e.g., Sandholm et al. [78] for proofs of these bounds and de Bruijn [20] for an asymptotically tight bound). Unfortunately, in PFGs, it is not possible to avoid brute-force search given an arbitrary partition function. This is because knowing the value of a coalition structure, $CS \in \Pi^A$, tells us nothing about the value of any other coalition structure, $CS' \in \Pi^A : CS' \neq CS$, even if CS and CS' have some elements (i.e., coalitions) in common. Therefore, studying the coalition structure generation problem in PFGs is only interesting when additional assumptions are placed on the partition function w (see Section 9 for more details).

ACCEPTED MANUSCRIPT

On the other hand, in CFGs, brute-force search can sometimes be avoided, as we will show in subsequent sections. Nevertheless, finding an optimal coalition structure in CFGs is NP-complete given oracle access to the characteristic function (see Proposition 2 in [78]). The proof is based on a reduction from the set-packing decision problem—where we are given a constant q and a family of sets $\mathcal{S}$ and we need to determine whether a set of q mutually-disjoint sets exists—to the following decision problem: given a characteristic function with which every coalition in $\mathcal{S}$ has a value of 1, and every coalition outside of $\mathcal{S}$ has a value of 0, determine whether a coalition structure exists whose value is at least q.

3. Space Representations

Understood in its simplest terms, the coalition structure generation problem implies searching Π^A—the space of all possible coalition structures over A—in order to find one whose value is maximal. Several approaches to this problem are based on the observation that the set Π^A has some structure, which can be exploited to speed up the search process. To understand how these approaches work, it is useful to consider representations for Π^A that exploit this structure. We therefore begin by presenting the two main representations of Π^A that have been proposed. Observe that every coalition structure in Π^A represents a possible solution to the coalition structure generation problem. As such, the terms "*coalition structure*", "*partition*", and "*solution*" will be used interchangeably, and Π^A will often be referred to as the "*search space*".

The first representation we consider was proposed by Sandholm et al. [78], and is called the *coalition structure graph*. Specifically, in this undirected graph:

- Every node represents a coalition structure. These nodes are categorized into levels, $\Pi^A_1, \ldots, \Pi^A_n$, where level Π^A_i contains the nodes that represent all coalition structures containing exactly i coalitions.
- An edge connects two coalition structures if and only if: (1) they belong to two consecutive levels Π^A_i and Π^A_{i-1}, and (2) the coalition structure in Π^A_i can be obtained from the one in Π^A_{i-1} by splitting one coalition into two.

A four-agent example can be seen in Figure 1.[4]

While the above representation categorizes the coalition structures according to the *number of coalitions* they contain, an alternative representation was proposed to categorize them based on the *sizes of the coalitions* they contain [71]. More specifically, this representation divides the space of coalition structures into disjoint subspaces that are each represented by an *integer partition* of n. Recall that an integer partition of n is a multiset of positive integers, or *parts*, whose sum (with multiplicities) is equal to n [2]. We will denote the set of all such integer partitions by $\mathcal{I}^n$. For instance, $\mathcal{I}^4 = \{\{4\}, \{1,3\}, \{2,2\}, \{1,1,2\}, \{1,1,1,1\}\}$. For every $I \in \mathcal{I}.$, we denote by $\Pi^A_I \subseteq \Pi^A$ the subspace consisting of all the coalition structures within which the sizes of the coalitions match the parts of I. For instance, $\Pi^{\{a_1,a_2,a_3,a_4\}}_{\{1,1,2\}}$ is the subspace consisting of all the coalition structures within which two coalitions are of size 1 and one coalition is of size 2. This representation can be encoded by an *integer partition graph* [62]. This is an undirected graph, where every subspace is represented by a node, and two nodes representing $I, I' \in \mathcal{I}^n$, are connected via an edge if and only if there exists two parts $i, j \in I$ such that $I' = (I \setminus \{i,j\}) \uplus \{i+j\}$ (here $\uplus$ denotes the multiset union operation). For example, Figure 2 shows the integer partition graph of four agents, as well as the subspaces that correspond to every node in the graph.

Having described some key representations for the search space, we will now present different approaches to the coalition structure generation problem, some of which depend directly upon those representations.

[4]The reason for highlighting some of the edges in Figure 1 will be made clear in the following section.

ACCEPTED MANUSCRIPT

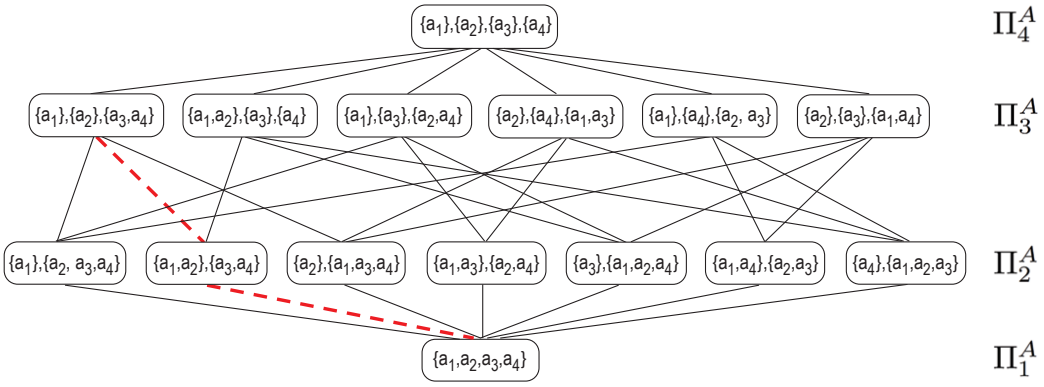


Figure 1: The coalition structure graph of 4 agents.

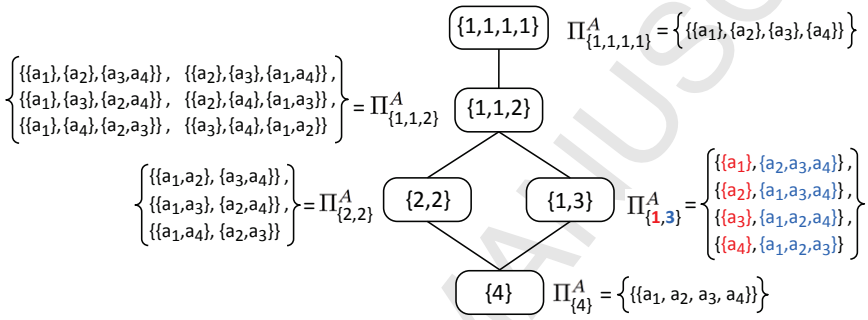


Figure 2: The integer partition graph representation of 4 agents.

4. Dynamic Programming Algorithms

The first dynamic programming algorithm for the coalition structure generation problem, called DP, was proposed by Yeh [93]. The DP algorithm depends directly on the following theorem.

Theorem 1. *Given a coalition $C \subseteq A$, let $f(C)$ be the value of an optimal coalition structure over C. That is, $f(C) = \max_{CS \in \Pi^C} V(CS)$. Then*

$$f(C) = \begin{cases} v(C) & \text{if } |C| = 1 \\ \max\left\{ v(C) \ , \ \max_{\{C',C''\} \in \Pi^C} \left(f(C') + f(C'') \right) \right\} & \text{otherwise.} \end{cases} \tag{1}$$

Proof. The result is immediate when $|C| = 1$. Thus, for the remainder of the proof we will assume that $|C| > 1$. We make use of the following lemma.

Lemma 1. *For any coalition $C \subseteq A$, if $CS = \{C_1, \ldots, C_{k>1}\}$ is an optimal coalition structure over (i.e., an optimal partition of) C, then for any $j : 1 \leq j < k$, it holds that $CS' = \{C_1, \ldots, C_j\}$ is an optimal coalition structure over $C' = \cup CS'$, and $CS'' = \{C_{j+1}, \ldots, C_k\}$ is an optimal coalition structure over $C'' = \cup CS''$.*

Proof of Lemma 1. Observe that $C = C' \cup C''$ and that $CS = CS' \cup CS''$ and $V(CS) = V(CS') + V(CS'')$. Suppose for the sake of contradiction that CS' was not an optimal coalition structure over C'. Then there exists another one, say $CS^\dagger \in \Pi^{C'}$, such that $V(CS^\dagger) > V(CS')$. However, this implies that $CS^\dagger \cup CS''$ is a coalition structure whose value—$V(CS^\dagger) + V(CS'')$—is greater than $V(CS) = V(CS') + V(CS'')$, which contradicts the assumption that CS is optimal.

ACCEPTED MANUSCRIPT

The same contradiction is reached if we assume that CS'' is not an optimal coalition structure over C'', which completes the proof of the lemma. □

Now, let $opt(C)$ be some optimal partition of C, i.e., $opt(C) \in \arg\max_{P \in \Pi^C} V(P)$. Lemma 1 shows that if $|opt(C)| > 1$, then there exists a coalition structure $\{C', C''\} \in \Pi^C$ such that $opt(C) = opt(C') \cup opt(C'')$. On the other hand, if $|opt(C)| = 1$, then surely we would have $opt(C) = \{C\}$ and $V(opt(C)) = v(C)$. Equation (1) covers both possibilities by taking the maximum over $v(C)$ and $\max_{\{C',C''\} \in \Pi^C} \big(f(C') + f(C'')\big)$. This concludes the proof of Theorem 1. □

Based on Theorem 1, the way DP works is by iterating over all the coalitions of size 1, and then over all those of size 2, and then size 3, and so on until size n. For every such coalition, C, it computes $f(C)$ using Equation (1). As can be seen from the equation, whenever $|C| \neq 1$, the algorithm must compare $v(C)$ with $\max_{\{C',C''\} \in \Pi^C} f(C') + f(C'')$ to determine which one is greater. The outcome of this comparison is stored in a variable called $t(C)$. In more detail:

- if $v(C)$ was greater, the algorithm sets $t(C) = \{C\}$, indicating that it is *not* beneficial to split C.
- if $v(C)$ was smaller, the algorithm sets $t(C) = \arg\max_{\{C',C''\} \in \Pi^C} f(C') + f(C'')$ to record the best way of splitting C into two coalitions.

As for cases where $|C| = 1$, the algorithm always sets $t(C) = \{C\}$ (because it is not possible to split C). By the end of this process, $f(A)$ will be computed, which is by definition equal to $V(CS^*)$. What remains is to compute CS^* itself. This is done recursively as illustrated in the following example:

Example 1. *Given $A = \{a_1, a_2, a_3, a_4\}$, suppose that $t(A) = \{\{a_1, a_2\}\{a_3, a_4\}\}$, i.e., it is most beneficial to split A into $\{a_1, a_2\}$ and $\{a_3, a_4\}$. Moreover, suppose that $t(\{a_1, a_2\}) = \{\{a_1\}, \{a_2\}\}$, while $t(\{a_3, a_4\}) = \{a_3, a_4\}$, i.e., it is most beneficial to split $\{a_1, a_2\}$ into $\{a_1\}$ and $\{a_2\}$, and not to split $\{a_3, a_4\}$. Then, $CS^* = \{\{a_1\}, \{a_2\}, \{a_3, a_4\}\}$.*

Figure 3 illustrates the workings of DP given a sample characteristic function and a set of four agents (Example 1 is illustrated as "step 5" in the figure). DP takes hours on a modern desktop computer to partition 30 agents.

Theorem 2. *Given a set of n agents, the dynamic programming algorithm, DP, computes an optimal coalition structure in $O(3^n)$ time.*

Proof. For every coalition of size s, DP checks all possible ways of splitting it in two, so the number of operations is: $\sum_{s=1}^{n} \binom{n}{s} 2^{s-1}$. It is straightforward to show that this number equals $\frac{1}{2}3^n - \frac{1}{2}$, simply by using the following standard equality, and assigning $a = 2$ and $b = 1$:

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}.$$

□

Having described DP, let us now analyze its workings. The operation of the DP algorithm can usefully be visualized on the coalition structure graph [63]. In particular, what DP actually does is the following: (1) **evaluate every movement** upward in the graph, (2) **store the best movements** in the table t, and finally (3) **move upwards in the graph**, starting from the bottom node, as long as it is beneficial to do so. To better understand the intuition behind this, observe that every movement upward in the graph corresponds to *splitting* one coalition into two (see Figure 1). Also observe that DP works by determining, for every coalition $C \subseteq A$, whether it is beneficial to split C, and if so what is the best such split (the answer to this question is stored in $t(C)$). This means that if DP were to move upwards in the graph, starting

ACCEPTED MANUSCRIPT

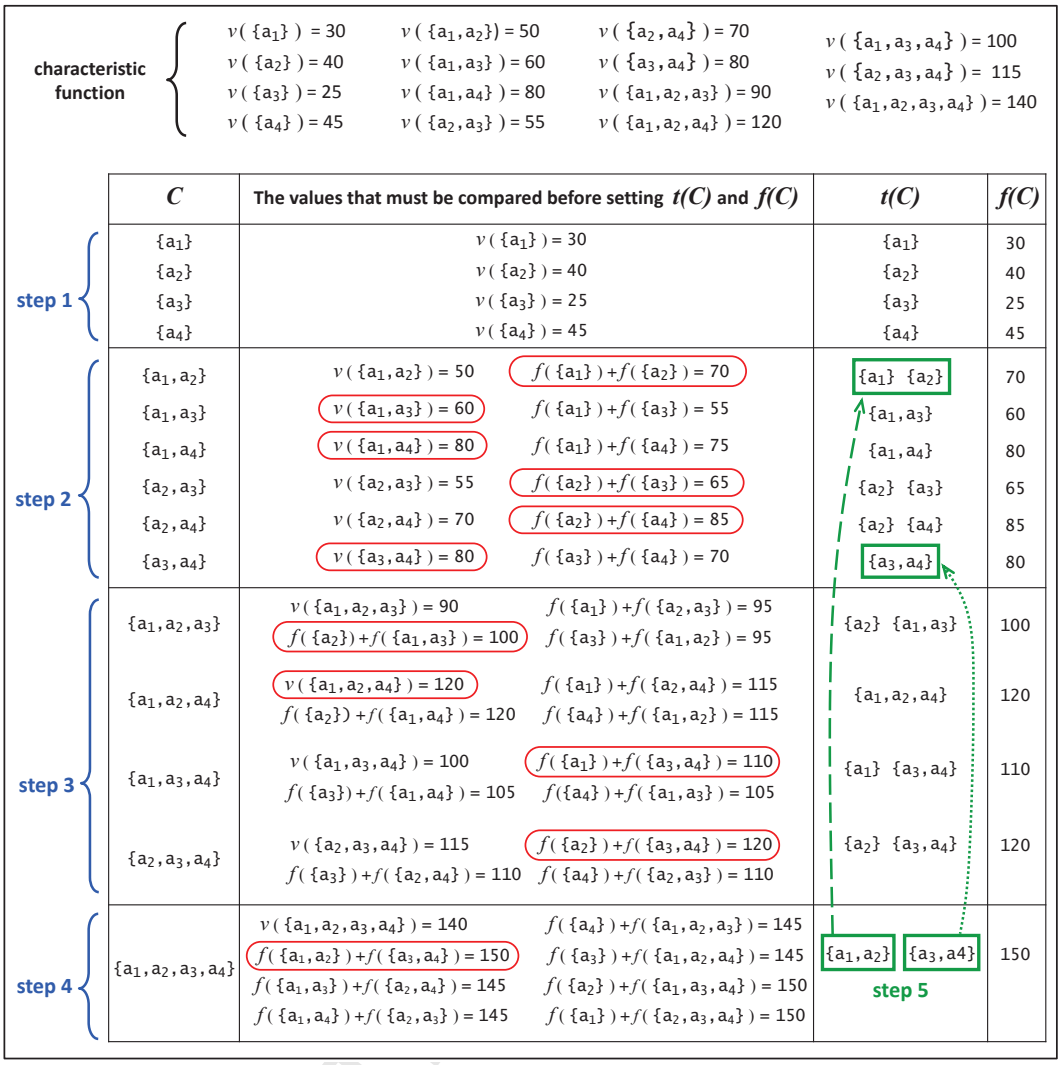


Figure 3: A four-agent example of how DP computes $t(C)$ and $f(C)$ for every $C \subseteq A$.

from the bottom node, then every time it reaches a node (i.e., a coalition structure) that contains C, it can determine (based on $t(C)$) whether it is beneficial to make a movement that involves splitting C, and if so what is the best such movement. Finally, once DP evaluates all possible movements (i.e., once it computes $t(C)$ for all $C \subseteq A$), it starts making a series of movements upward until an optimal node is reached, after which no movement is beneficial. For instance, the way DP reached $\{\{a_1\}, \{a_2\}, \{a_3, a_4\}\}$ in Example 1 can be visualized as movements through the dashed path in Figure 1, where the first movement involved splitting $\{a_1, a_2, a_3, a_4\}$ into $\{a_1, a_2\}$ and $\{a_3, a_4\}$, and the second movement involved splitting $\{a_1, a_2\}$ into $\{a_1\}$ and $\{a_2\}$.

From this visualization it is clear that, for every $CS : |CS| > 2$, there are multiple paths that start from the bottom node of the graph, and end with the node containing CS (see Figure 1). This raises the following question: *Can DP reach an optimal node through any of the paths leading to that node?* The answer is "*yes*" [63]. To understand the intuition behind this, recall that if it is beneficial to split a coalition C, then DP sets $t(C) = \arg\max_{\{C', C''\} \in \Pi^C} f(C') + f(C'')$. This implies that if there is more than one solution, DP has no preference over which one to store in $t(C)$. The argmax that ends up stored in $t(C)$ will determine the movement that DP makes whenever it encounters a node representing some $CS : C \in CS$.

ACCEPTED MANUSCRIPT

However, by definition, all the moves that correspond to those argmaxes will eventually lead to an equally favorable partition of C.

The above observation raises yet another question: *what happens if DP is modified so that it avoids evaluating some of the movements?* Rahwan et al. [64] proved that, as long as there still exists a path of evaluated movements, leading from the bottom node to an optimal solution, DP will still be able to reach that solution. To understand the intuition behind this, suppose that for some coalition C, the algorithm did not evaluate a movement that involves splitting C into two particular coalitions, C_1 and C_2. In other words, suppose that the term: $\max_{\{C',C''\}\in\Pi^C} f(C') + f(C'')$ in Equation (1) was replaced with: $\max_{\{C',C''\}\in\Pi^C\setminus\{\{C_1,C_2\}\}} f(C') + f(C'')$. This implies that, whenever a coalition structure $CS : C \in C$ is reached, the movement from CS to the coalition structure $CS' = (CS\backslash C) \cup \{C_1, C_2\}$ would no longer be an option. Similarly, every path containing this movement would not be an option. This can be visualized by *removing the edge* that connects CS to CS'. Importantly, however, this removal does not affect DP's evaluation of any path not containing the removed edge. This is precisely why, if there still exists a path of evaluated movements that lead to CS', then DP would still be able to reach CS'.

Based on the above observation, Rahwan et al. [64] developed a version of DP which avoids the evaluation of many movements, without losing the guarantee of having a path that leads to every node in the graph. To describe Rahwan et al.'s version more formally, we need additional notation. To this end, for any two coalitions $C', C'' \in \mathcal{C}^A$, let us write $C' < C''$ if and only if C' precedes C'' lexicographically, e.g., we write $\{a_1, a_3, a_9\} < \{a_1, a_4, a_5\}$ and $\{a_4\} < \{a_4, a_5\}$. Now, given two disjoint coalitions C_1 and C_2, let m^{C_1,C_2} denote the movement that corresponds to splitting $C = C_1 \cup C_2$ into C_1 and C_2. Moreover, let $\mathcal{M}$ denote the set of all possible movements in the graph. Finally, let M^* be a subset of movements defined as follows:

$$M^* = \left\{ m^{C',C''} \in \mathcal{M} \ : \ \Big(C' \cup C'' = A\Big) \text{ or } \Big(C' < C'' < A \setminus (C' \cup C'')\Big) \right\}. \tag{2}$$

While DP evaluates every movement in $\mathcal{M}$, Rahwan et al.'s version only evaluates the movements in M^* based on the following theorem.

Theorem 3. *Starting from the bottom node in the coalition structure graph, it is possible to reach every node in the graph through a series of movements in M^*.*

Proof. It suffices to prove that for every $k \geq 2$, every $CS = \{C_1, \ldots, C_k\}$ is reachable from some coalition structure CS' with $|CS'| = k - 1$ via some movement in M^*. Assume without loss of generality that $C_1 < \cdots < C_k$. We will show that CS is reachable from the coalition structure $(CS\backslash\{C_1, C_2\}) \cup \{C_1 \cup C_2\}$ via M^*. To this end, it suffices to show that $m^{C_1,C_2} \in M^*$.

First, suppose that $k = 2$. In this case, we have $CS = \{C_1, C_2\}$, and so $C_1 \cup C_2 = A$. This means that $m^{C_1,C_2} \in M^*$.

Now, suppose that $k > 2$. In this case, since $C_1 < \cdots < C_k$, we obtain $C_1 < C_2 < (C_3 \cup \ldots \cup C_k)$, and hence $C_1 < C_2 < A \setminus (C \cup C')$. Thus, we also have $m^{C_1,C_2} \in M^*$. □

Rahwan et al. [64] proved that $|M^*| = \left|\Pi_2^A\right| + \left|\Pi_3^A\right|$. They also proved that it is not possible to evaluate fewer movements than $|M^*|$, while still maintaining the guarantee of finding an optimal solution. This makes M^* *optimal* in the sense that it minimizes the number of evaluated movements. Based on this, the version of DP that evaluates M^* is called ODP, which stands for Optimal DP.

ODP avoids approximately two thirds of the operations compared to DP, making it the fastest algorithm guaranteed to find an optimal solution in $O(3^n)$ time to date. This is significantly less than $\omega(n^{n/2})$—the time required to exhaustively enumerate all coalition structures. However, the disadvantage is that ODP provides no interim solution before completion, meaning that it is not possible to trade computation time for solution quality.

ACCEPTED MANUSCRIPT

Finally, we mention the dynamic-programming algorithm by [14], which is based on the inclusion-exclusion principle. The algorithm uses an interesting technique in which information is encoded into extremely large numbers (e.g., consisting of hundreds, or even thousands, of digits each). This is the state-of-the-art algorithm in terms of worst-case complexity: it runs in $O(2^n)$. In practice, however, the algorithm is slow since the theoretical analysis does not take into consideration the time required to for the many inevitable multiplications of such huge numbers. In fact, in practice, the algorithm has been shown to have a running time of approximately 6^n. So, for example, it takes several months to solve a problem of 15 agents, while ODP requires only 0.01 seconds for that many agents.

5. Anytime Exact Algorithms

An *anytime algorithm* is one whose solution quality improves monotonically as computation time increases [95]. In our case, this is particularly desirable as the agents might not always have sufficient time to run the algorithm to completion due to the exponential size of the search space. Moreover, being anytime makes the algorithm robust against failure; if the execution is stopped before the algorithm would normally have terminated, then it can still return a solution that is better than the initial—or any other intermediate—one.

In this section, we will focus on anytime algorithms that return optimal solutions or, at least, provide worst-case guarantees on the quality of their solutions. In more detail, Section 5.1 presents algorithms that divide the space into subspaces, and specify the order in which these subspaces must be searched, so as to ensure that the worst-case guarantee on solution quality improves after each subspace. Section 5.2 presents an algorithm that searches the subspaces of the integer partition graph one at a time, using depth-first branch-and-bound search. Section 5.3 presents anytime algorithms that use dynamic programming techniques. Finally, Section 5.4 presents an integer programming formulation of the coalition structure generation problem.

5.1. Identifying Subspaces with Worst-Case Guarantees

If the space is too large to be fully searched, then *can we search through only a subset of this space, and be guaranteed to find a solution that is within a certain bound β from the optimal solution?*

To address this question, a number of algorithms have been proposed, which all work by (1) dividing the search space into disjoint and exhaustive subspaces, and (2) identifying a *sequence* in which these subsets are searched, such that the worst-case bound on solution quality is guaranteed to improve after each subspace. In particular, after each subspace is searched, a bound is established by comparing the coalition structures that have already been searched against those that are yet to be searched. This comparison is carried out "offline", i.e., without analyzing the characteristic function at hand. Having bounds that are independent of the coalition values themselves means that the bounds are guaranteed for *any characteristic function*. This also makes such algorithms applicable in any situation where the *coalition structure values are observable, but the coalition values are not*. Observe that such algorithms do not specify how each subspace is searched. However, one can extend such an algorithm (possibly in many directions) by specifying the techniques used to search each subspace. Such techniques can capitalize on the extra information accrued during the search, e.g., to avoid brute-force search and hopefully establish better bounds than those established offline. Next, we provide an overview of those algorithms for characteristic function games.[5]

The first such algorithm was proposed in the seminal paper by Sandholm et al. [78], and is mainly based on the following theorem.

[5]Other algorithms that follow the same design paradigm have been proposed for partition function games (see Section 9.2 for more details).

ACCEPTED MANUSCRIPT

Theorem 4. *To establish a worst-case bound β, it suffices to search the lowest two levels of the coalition structure graph, i.e., Π_1^A and Π_2^A. With this search, the bound is $\beta = n$, and the number of searched coalition structures is 2^{n-1}. Moreover, no algorithm can establish any bound by searching a different set of at most 2^{n-1} coalition structures.*

Proof. For a partial search to establish a bound on solution quality, every coalition $C \subseteq A$ must appear in at least one of the searched coalition structures. This is due to the possibility of having a single coalition whose value is arbitrarily greater than that of other coalitions. Now, since the grand coalition appears in Π_1^A, and every other coalition $C \subset A$ appears in $\{C, A \backslash C\} \in \Pi_2^A$, then the value of the best coalition structure in $\Pi_1^A \cup \Pi_2^A$ is no less than: $\max_{C \subseteq A} v(C)$. On the other hand, since CS^* can include at most n coalitions, its value is no greater than: $n \times \max_{C \subseteq A} v(C)$. This means $\frac{V(CS^*)}{\max_{CS \in \Pi_1^A \cup \Pi_2^A} V(CS)} \leq n$.

As for the number of searched coalition structures, it suffices to note that: $\Pi_1^A \cup \Pi_2^A = \cup_{C \subseteq N : a_1 \in C} \{C, A \backslash C\}$, and so that number is: $|C \subseteq N : a_1 \in C| = 2^{n-1}$.

Finally, we need to prove that for every set of coalition structures $X \neq \Pi_1^A \cup \Pi_2^A$ whose search establishes a bound, the following holds: $|X| > 2^{n-1}$. Since searching X establishes a bound, every coalition must belong to at least one coalition structure in X. This implies that $\{A\} \in X$, i.e., $\Pi_1^A \subseteq X$. Now, let $Y = \{\{P_1, Q_1\}, \cdots, \{P_k, Q_k\}\}$ denote the coalition structures in Π_2^A that are not in X. Observe that for $i, j : i \neq j$, at least one of the following is true: $P_i \cap P_j \neq \emptyset$, $P_i \cap Q_j \neq \emptyset$, or $Q_i \cap Q_j \neq \emptyset$. Thus, we must have at least $k+1$ coalition structures in X that are not in Π_2^A (this is the only way to ensure that every coalition in $\{P_1, \cdots, P_k, Q_1, \cdots, Q_k\}$ appears in some coalition structure in X, given that $X \cap Y = \emptyset$). $\square$

Based on Theorem 4, the algorithm starts by searching the bottom two levels. After that, if additional time is available, the algorithm searches the remaining levels one by one, starting from the top level and moving downwards, i.e., the algorithm's entire sequence is: $\langle (\Pi_1^A \cup \Pi_2^A), \Pi_n^A, \Pi_{n-1}^A, \ldots, \Pi_3^A \rangle$. Sandholm et al. proved that the bound improves after each step. In particular, once the algorithm completes searching level Π_i^A, the bound becomes $\beta = \lfloor n/h \rfloor$, where $h = \lfloor (n-i)/2 \rfloor + 2$. The only exception is when $n \equiv h - 1 (mod\ h)$ and $n \equiv i (mod\ 2)$, in which case the bound becomes $\beta = \lceil n/h \rceil$. Importantly, this means that after searching $(\Pi_1^A \cup \Pi_2^A)$ and establishing the bound $\beta = n$, one can drop (i.e., improve) the bound to $\beta = \lceil n/2 \rceil$ by searching Π_n^A, which only contains a single coalition structure.

Dang and Jennings [19] proposed a different algorithm, which starts by searching $\Pi_1^A \cup \Pi_2^A$ and then searches Π_n^A (as Sandholm et al.'s algorithm does). After that, however, instead of searching Π_i^A with i running from $n-1$ down to 3 (as Sandholm et al. do), the algorithm searches all coalition structures that have at least one coalition whose size is not smaller than $\lceil n(q-1)/q \rceil$, with q running from $\lfloor (n+1)/4 \rfloor$ down to 2. Dang and Jennings proved that, for any given value of q, the algorithm establishes a bound $\beta = 2q - 1$.

So far, we have established bounds on solution quality in certain subspaces (those are the subspaces specified by Sandholm et al. and by Dang and Jennings). But *how can we establish a bound on solution quality in an arbitrary subspace?* Furthermore, for each of the aforementioned subspaces, the bound is established on the best solution in that subspace with respect to the best solution in the entire space, i.e., CS^*. This raises yet another question: *Can we bound the best solution in a subspace, with respect to the best solution in some other (possibly overlapping) subspace?* To answer these questions, we need additional notation. Recall that Π^A denotes the set of possible partitions of A. In a similar way, let Π^{CS} denote the set of possible partitions of CS. For instance, given $CS = \{\{a_1\}, \{a_2, a_3\}, \{a_4\}\}$ we have:

$$\Pi^{CS} = \left\{ \begin{array}{cc} \{\ \{\{a_1\}\},\ \{\{a_2, a_3\}\},\ \{\{a_4\}\}\ \}, & \{\ \{\{a_1\}, \{a_2, a_3\}\},\ \{\{a_4\}\}\ \}, \\ \{\ \{\{a_1\}\},\ \{\{a_2, a_3\}, \{a_4\}\}\ \}, & \{\ \{\{a_1\}, \{a_4\}\},\ \{\{a_2, a_3\}\}\ \}, \\ \multicolumn{2}{c}{\{\ \{\{a_1\}, \{a_2, a_3\}, \{a_4\}\}\ \}} \end{array} \right\}$$

ACCEPTED MANUSCRIPT

Finally, for every subspace $\Pi' \subseteq \Pi^A$, let $\delta(\Pi')$ be the set consisting of every non-empty subset of every coalition structure in Π'. For instance, if $\Pi' = \{CS_1, CS_2\}$ where $CS_1 = \{\{\{a_1, a_2\}, \{a_3\}\}$ and $CS_2 = \{\{a_1\}, \{a_2, a_3\}\}\}$, then $\delta(\Pi')$ consists of all non-empty subsets of CS_1 or CS_2, i.e., the following six subsets: (1) $\big\{\{a_1, a_2\}\big\}$, (2) $\big\{\{a_3\}\big\}$, (3) $\big\{\{a_1, a_2\}, \{a_3\}\big\}$, (4) $\big\{\{a_1\}\big\}$, (5) $\big\{\{a_2, a_3\}\big\}$, and (6) $\big\{\{a_1\}, \{a_2, a_3\}\big\}$. Now, we are ready to answer the above questions with the following theorem [69]:

Theorem 5. *For any two subspaces, $\Pi', \Pi'' \subseteq \Pi^A$, a bound can be established on $\frac{\max_{CS \in \Pi''} V(CS)}{\max_{CS \in \Pi'} V(CS)}$ if and only if the following holds for every $C \subseteq A$:*

$$\exists CS'' \in \Pi'' : C \in CS'' \qquad \Rightarrow \qquad \exists CS' \in \Pi' : C \in CS'. \tag{3}$$

Furthermore, if the above condition holds, then we can establish the following bound:

$$\frac{\max_{CS \in \Pi''} V(CS)}{\max_{CS \in \Pi'} V(CS)} \leq \max_{CS \in \Pi''} \left(\min_{P \in \Pi^{CS} : P \subseteq \delta(\Pi')} |P| \right). \tag{4}$$

Proof. If condition (3) is not satisfied, then there exists a coalition that appears in Π'' but not in Π'. Since this coalition can be arbitrarily better than all other coalitions, it is not possible to establish a bound on $\frac{\max_{CS \in \Pi''} V(CS)}{\max_{CS \in \Pi'} V(CS)}$.

On the other hand, if condition (3) is satisfied, then Theorem 5 states that (4) holds. To understand the intuition behind this, let us revisit the proof of Theorem 4. In particular, that proof was primarily based on the following observations:

- the best solution in Π^A, i.e., $\arg\max_{CS \in \Pi^A} V(CS)$, **contains at most n coalitions**,
- and every one of those coalitions appears in some $CS \in \Pi^A_1 \cup \Pi^A_2$,
- then $\arg\max_{CS \in \Pi^A} V(CS)$ **is at most n times better** than $\arg\max_{CS \in \Pi^A_1 \cup \Pi^A_2} V(CS)$.

Let us generalize this observation, from *coalitions*, to *groups of disjoint coalitions*. Roughly speaking, if we can identify a set of such groups, where:

- the best solution in Π'', i.e., $\arg\max_{CS \in \Pi''} V(CS)$, **contains at most x groups**,
- and every one of those groups appears in some $CS \in \Pi'$,
- then $\arg\max_{CS \in \Pi''} V(CS)$ **is at most x times better** than $\arg\max_{CS \in \Pi'} V(CS)$.

Equation (4) captures the above observation. The remainder of the proof shows how this is case.

Let us first focus on a **single coalition structure in Π''**, namely CS''. We know from condition (4) that every $C \in CS''$ appears somewhere in Π'. This implies that CS'' can be partitioned into *groups of disjoint coalition*, each appearing somewhere in Π'.[6] In other words, there exists at least one partition of CS'' consisting of elements of $\delta(\Pi')$. Let us focus on the partition of CS'' consisting of the smallest number of such elements. This number is: $x = \min_{P \in \Pi^{CS''} : P \subseteq \delta(\Pi')} |P|$. By partitioning CS'' in this way, we have shown that CS'' contains at most x groups of disjoint coalitions, each being an element of $\delta(\Pi')$ (i.e., every one of those groups appears in some $CS \in \Pi'$). Therefore, CS'' is at most x times better than $\arg\max_{CS \in \Pi'} V(CS)$.

[6] In the worst case, every such group contains exactly one coalition. However, we could have groups containing multiple coalitions each.

ACCEPTED MANUSCRIPT

Finally, if we take **all coalition structures in** Π'' into consideration, we find that none of them can be more than x^* times better than $\arg\max_{CS\in\Pi'} V(CS)$, where $x^* = \max_{CS\in\Pi''}\left(\min_{P\in\Pi^{CS}:P\subseteq\delta(\Pi')}|P|\right)$. □

Now if Π' and Π'' can each be represented by some integer partition(s) of n, i.e., if:

$$\exists\, \mathcal{I}', \mathcal{I}'' \subseteq \mathcal{I} : \left(\Pi' = \bigcup_{I\in\mathcal{I}'} \Pi_I^A\right) \wedge \left(\Pi'' = \bigcup_{I\in\mathcal{I}''} \Pi_I^A\right),$$

then we will show how the result in Theorem 5 can be simplified. To this end, just as Π^{CS} denotes the set of partitions of CS, let Π^I denote the set of partitions of the integer partition I, e.g., $\Pi^{\{1,1,2\}}$ consists of the following four partitions: $\{\{1,1,2\}\}$, $\{\{1,1\},\{2\}\}$, $\{\{1,2\},\{1\}\}$, and $\{\{1\},\{1\},\{2\}\}$. Moreover, just as $\delta(\Pi')$ denotes the set of every non-empty subset of every $CS \in \Pi'$, let $\delta(\mathcal{I}')$ denote the set of every non-empty subset of every $I \in \mathcal{I}'$, e.g., $\delta(\{\{1,1,2\},\{1,3\}\})$ consists of the following seven subsets: $\{1\}$, $\{2\}$, $\{3\}$, $\{1,1\}$, $\{1,2\}$, $\{1,3\}$ and $\{1,1,2\}$. Now, we are ready to state the following corollary, which can be thought of as an integer-partition version of Theorem 5:

Corollary 1. *Let $\mathcal{I}', \mathcal{I}'' \subseteq \mathcal{I}^n$ be two arbitrary sets of integer partitions, and let $\Pi' = \bigcup_{I\in\mathcal{I}'} \Pi_I^A$ and $\Pi'' = \bigcup_{I\in\mathcal{I}''} \Pi_I^A$. A bound can be established on $\frac{\max_{CS\in\Pi''} V(CS)}{\max_{CS\in\Pi'} V(CS)}$ if and only if the following holds for every $i \in \{1,\ldots,n\}$:*

$$\exists I'' \in \mathcal{I}'' : i \in I'' \quad \Rightarrow \quad \exists I' \in \mathcal{I}' : i \in I'. \tag{5}$$

Furthermore, if the above condition holds, then we can establish the following bound:[7]

$$\frac{\max_{CS\in\Pi''} V(CS)}{\max_{CS\in\Pi'} V(CS)} \leq \max_{I\in\mathcal{I}''}\left(\min_{P\in\Pi^I:P\subseteq\delta(\mathcal{I}')}|P|\right). \tag{6}$$

Based on the above corollary, Rahwan et al. [69] proposed an algorithm that searches through a sequence of subspaces, each being Π_I^A for some $I \in \mathcal{I}^n$. This has been shown to outperform both Sandholm et al.'s algorithm and Dang and Jennings's algorithm (e.g., to reach a bound of 2 given 9 agents, Rahwan et al. search around 2,000 solutions, while Sandholm et al. search around 10,000 and Dang and Jennings search around 21,000). The reason behind this gain is that Rahwan et al.'s algorithm is (in a sense) a generalization of the other two algorithms.

Observe that this latter algorithm specifies an order in which integer-partition based subspaces should be searched, and the specifies how the bound improves during this search. However, it does not specify *how* each subspace should be searched. As such, it would be useful to have an algorithm that can efficiently search such subspaces. In the following subsection, we present an algorithm that does exactly that.

5.2. Integer Partition-based Search

An anytime algorithm, called IP, was developed by Rahwan et al. [73] based on the integer partition-based representation from Section 3. In particular, it is based on the observation that, for any subspace Π_I^A, it is possible to compute upper and lower bounds on the value of the best coalition structure in that subspace. More formally, let Max_s^A and Avg_s^A be the maximum and average values of all coalitions of size s, respectively. It turns out that one can compute the average value of the coalition structures in each $\Pi_I^A : I \in \mathcal{I}^n$ without inspecting these coalition structures [73]:

[7] While P is used in Theorem 5 to denote a *partition* of CS, it is used here to denote a *partition* of I. This is to simplify notation, as P stands for *Partition*.

ACCEPTED MANUSCRIPT

Theorem 6. *For every $I \in \mathcal{I}^n$, let $I(i)$ be the multiplicity of i in I. Then:*

$$\frac{\sum_{CS \in \Pi_I^A} V(CS)}{\left|\Pi_I^A\right|} = \sum_{i \in I} I(i) \cdot Avg_i^A. \tag{7}$$

Proof. For every coalition $C \subseteq A$, the number of coalition structures in Π_I^A that contain C depends solely on the size of C. In other words, this number is equal for any two coalitions that are of the same size. Let us denote this number by $\mathcal{N}_I^{|C|}$. Formally, for every $C \subseteq A$, we set $\mathcal{N}_I^{|C|} = \left|\{CS \in \Pi_I^A : C \in CS\}\right|$. Then, we have

$$\sum_{CS \in \Pi_I^A} V(CS) = \sum_{i \in I} \sum_{C:|C|=i} \mathcal{N}_I^i \cdot v(C) = \sum_{i \in I} \mathcal{N}_I^i \sum_{C:|C|=i} v(C) = \sum_{i \in I} \mathcal{N}_I^i \cdot \binom{n}{i} \cdot Avg_i^A,$$

where $\binom{n}{i}$ is the binomial coefficient (i.e., the number of possible coalitions of size i). Thus, to prove (7) it suffices to prove that:

$$\frac{\sum_{i \in I} \mathcal{N}_I^i \cdot \binom{n}{i} \cdot Avg_i^A}{\left|\Pi_I^A\right|} = \sum_{i \in I} I(i) \cdot Avg_i^A.$$

This can be done by proving that the following holds for all $i \in I$:

$$\mathcal{N}_I^i \cdot \binom{n}{i} = I(i) \cdot \left|\Pi_I^A\right|. \tag{8}$$

Observe that every $CS \in \Pi_I^A$ contains exactly $I(i)$ coalitions of size i. Then:

$$\sum_{C:|C|=i} \mathcal{N}_I^{|C|} = \sum_{C:|C|=i} \sum_{CS \in \Pi_I^A : C \in CS} 1 = \sum_{CS \in \Pi_I^A} \sum_{C \in CS:|C|=i} 1 = \sum_{CS \in \Pi_I^A} I(i) = |\Pi_I^A| \cdot I(i).$$

We have shown that $\sum_{C:|C|=i} \mathcal{N}_I^{|C|} = |\Pi_I^A| \cdot I(i)$. On the other hand, since $\mathcal{N}_I^{|C|}$ is any coalitions of size $|C|$, then: $\sum_{C:|C|=i} \mathcal{N}_I^{|C|} = \binom{n}{i} \cdot \mathcal{N}_I^i$. Thus, Equation (8) holds. □

Note that, for every $I \in \mathcal{I}^n$, the value of the best coalition structure in Π_I^A is always greater than, or equal to, the average value of all coalition structures in Π_I^A. Thus, based on Theorem 6, we obtain a *lower bound* on the value of the best coalition structure in Π_I^A, which is: $LB_I^A = \sum_{i \in I} I(i) Avg_i^A$. Furthermore, by replacing Avg_i^A with Max_i^A in this expression, we obtain an *upper bound* UB_I^A on the value of the best coalition structure in Π_I^A, which is: $UB_I^A = \sum_{i \in I} I(i) Max_i^A$. Using these bounds, the algorithm computes an upper bound $UB^* = \max_{I \in \mathcal{I}^n} UB_I^A$ and a lower bound $LB^* = \max_{I \in \mathcal{I}^n} LB_I^A$ on the value of the optimal coalition structure CS^*. By computing UB^*, we can establish a bound on the quality of the best coalition structure found at any point in time, denoted by CS^{**}; this bound is $\beta = UB^*/V(CS^{**})$. On the other hand, by computing LB^*, we can identify any subspaces that cannot possibly contain an optimal coalition structure, which are $\Pi_I^A : UB_I^A < LB^*$. These subspaces are pruned from the search space. As for the remaining subspaces, the algorithm searches them one at a time. During this search, if a solution is found whose value is greater than, or equal to, the upper bound of a not-yet-searched subspace, the algorithm safely skips searching that subspace. Next, we explain how a subspace is searched.

Let $\mathcal{C}_i^A$ denote the set of coalitions of size i. Generally speaking, given an integer partition $I = \{i_1, \ldots, i_{|I|}\}$, one way of searching the corresponding subspace Π_I^A is by simply going through the cartesian product of the sets $\mathcal{C}_{i_1}^A, \ldots, \mathcal{C}_{i_{|I|}}^A$ and checking every combination of coalitions to determine whether it is *invalid* or *redundant*. Here, an invalid combination is one that contains overlapping coalitions, and a redundant combination is one that has already been examined with a different ordering of the coalitions.

ACCEPTED MANUSCRIPT

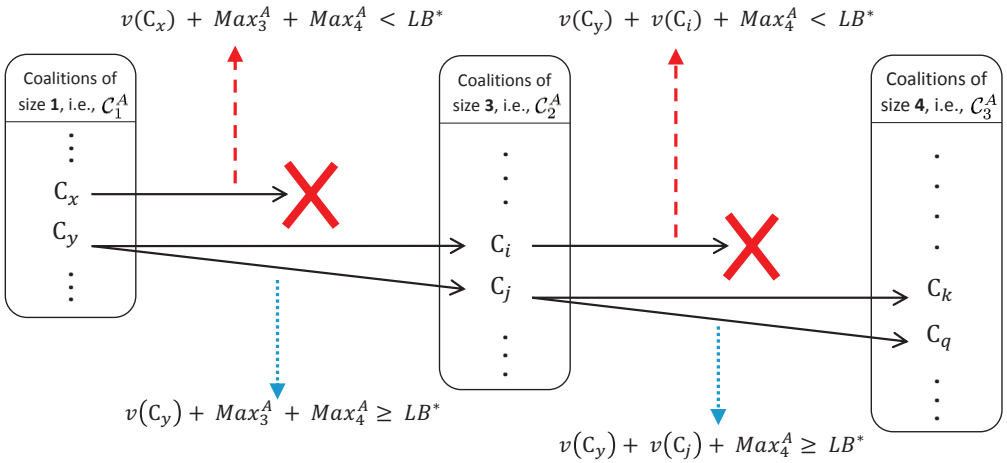


Figure 4: Illustrating IP's branch-and-bound technique while searching $\Pi^A_{\{1,3,4\}}$. Here, every coalition structure containing C_x or C_y, C_i cannot be optimal, and so IP does not proceed deeper in the search tree.

For example, given $\Pi^A_{\{1,2,2\}}$, and having examined $\{\{a_1\},\{a_2,a_3\},\{a_4,a_5\}\}$, the following combination is redundant and so no longer needs to be examined: $\{\{a_1\},\{a_3,a_4\},\{a_1,a_2\}\}$. Searching through the cartesian product, while correct, is inefficient because it involves examining a number of combinations that is significantly larger than the number of coalition structures in Π^A_I. To avoid this, IP uses a *depth-first search* technique that iterates over $\mathcal{C}^A_{i_1}$. For every coalition $C_1 \in \mathcal{C}^A_{i_1}$ that the algorithm encounters, it only iterates over the coalitions in $\mathcal{C}^A_{i_2}$ that do not overlap with C_1 nor lead to redundant coalition structures. Similarly, for every two coalitions $C_1 \in \mathcal{C}^A_{i_1}, C_2 \in \mathcal{C}^A_{i_2}$ that the algorithm encounters, it only iterates over the coalitions in $\mathcal{C}^A_{i_3}$ that do not overlap with $C_1 \cup C_2$ nor lead to redundant coalition structures. This is repeated until the coalition $C_{|I|} \in \mathcal{C}^A_{i_{|I|}}$ is encountered, in which case the algorithm would have a set $\{C_1 \in \mathcal{C}^A_{i_1}, \ldots, C_{|I|} \in \mathcal{C}^A_{i_{|I|}}\}$ which is guaranteed to be a unique coalition structure in Π^A_I. This process is repeated until all coalition structures in Π^A_I have been visited.

To speed up the search, IP applies a *branch-and-bound* technique. In particular, given some coalitions $C_1 \in \mathcal{C}^A_{i_1}, C_2 \in \mathcal{C}^A_{i_2}, \ldots, C_k \in \mathcal{C}^A_{i_k} : k < |I|$, and before iterating through the relevant coalitions in $\mathcal{C}^A_{i_{k+1}}, \ldots, \mathcal{C}^A_{i_{|I|}}$, the algorithm checks the following condition, where CS^{**} is the best coalition structure found so far by the algorithm:

$$\sum_{j=1}^{k} v(C_{i_j}) + \sum_{j=k+1}^{|I|} Max^A_{i_j} < V(CS^{**}). \tag{9}$$

Now if the above inequality holds, then none of the coalition structures containing $C_1, \ldots, C_k$ can improve upon the quality of the best solution found so far, in which case, the algorithm skips this part of the search. Figure 4 illustrates how IP searches $\Pi^A_{\{1,3,4\}}$ given 8 agents.

The IP algorithm runs in $O(n^n)$ time. In the worst case, it can end up constructing every possible coalition structure. In practice, however, IP is significantly faster than ODP for many coalition-value distributions (e.g., when tested against random characteristic functions on a modern desktop computer it took seconds to solve problems of 30 agents). Moreover, the bound that it generates, i.e., $\beta = UB^*/V(CS^{**})$, is significantly better than those obtained by searching particular subsets as per the previous subsection. Finally, we remark that a decentralized version of IP has been developed by Michalak et al. [47].

ACCEPTED MANUSCRIPT

5.3. Anytime, Dynamic Programming-based, Search

A number of algorithms have been developed that combine techniques from the dynamic programming approach and the anytime approach. The first such algorithm is called ODP-IP [62, 64]. As the name suggests, the algorithm combines ODP (from Section 4) with IP (from Section 5.2). Next, we explain the basic idea behind this combination.[8]

As discussed earlier in Section 4, the operation of ODP can be interpreted as the evaluation of movements through edges in the coalition structure graph (Figure 1). Furthermore, avoiding the evaluation of any movements can be visualized by removing the corresponding edges from that graph. Importantly, however, the way ODP works can also be visualized on the integer partition graph (Figure 2). Basically, by making a movement (in the coalition structure graph) from CS' to CS'', ODP is also making a movement (in the integer partition graph) from $I' : \Pi^A_{I'} \ni CS'$ to $I'' : \Pi^A_{I''} \ni CS''$. For instance, the movement from $\{\{a_1, a_2\}, \{a_3, a_4\}\}$ to $\{\{a_1\}, \{a_2\}, \{a_3, a_4\}\}$ in Figure 1 corresponds to the movement from $\{2, 2\}$ to $\{1, 1, 2\}$ in Figure 2. Moreover, by removing (from the coalition structure graph) all the edges that correspond to the splitting of a coalition of size s into two coalitions of sizes s' and s'', ODP is also removing (from the integer partition graph) the edges that connect every $I : I \ni s$ to $I' = (I \setminus \{s\}) \uplus \{s', s''\}$. This visualization provides the link between ODP and IP since the latter algorithm deals with subspaces that are represented by integer partitions.

Having presented the link between ODP and IP, we now show how to modify ODP so that it becomes compatible with IP. To this end, observe that for a given triple of positive integers s, s', s'' with $s' + s'' = s$, avoiding the evaluation of *all* possible ways of splitting *all* coalitions of size s into two coalitions of sizes s' and s'' corresponds to the removal of the edges from every $I : I \ni s$ to $I' = (I \setminus \{s\}) \uplus \{s', s''\}$. The problem with ODP is that it avoids evaluating *only some* of the movements from coalitions of a given size. For instance, given $n = 4$ and $s = 2$, ODP avoids evaluating the movements from the coalitions: $\{a_1, a_3\}$, $\{a_1, a_4\}$, $\{a_2, a_3\}$, $\{a_2, a_4\}$ and $\{a_3, a_4\}$, but evaluates the movement from $\{a_1, a_2\}$. To circumvent this, Rahwan et al. developed a *size-based version* of ODP (called **IDP**). Specifically, for any three sizes $s, s', s'' \in \{1, \ldots, n\}$ such that $s = s' + s''$, IDP evaluates either *all* or *none* of the movements in which a coalition of size s is split into coalitions of sizes s' and s''. More specifically, given two positive integers $s', s'' \in \mathbb{Z}^+$, let $M^{s',s''} \subseteq \mathcal{M}$ be the set of all movements in which a coalition of size $s' + s''$ is split into two coalitions of sizes s' and s''. That is, $M^{s',s''} = \{m^{C',C''} \in \mathcal{M} : |C'| = s', |C''| = s''\}$. Then, unlike ODP which evaluates the movements in M^* (see Equation 2), IDP evaluates the movements in M^{**}, where:

$$M^{**} = \left(\bigcup_{s',s'' \in \mathbb{Z}^+ : \max\{s',s''\} \leq n-s'-s''} M^{s',s''} \right) \cup \left(\bigcup_{s',s'' \in \mathbb{Z}^+ : s'+s''=n} M^{s',s''} \right).$$

Theorem 7. *Starting from the bottom node in the coalition structure graph, it is possible to reach every node in the graph through a series of movements in M^{**}.*

The above theorem is very similar to Theorem 3, except that M^* is now replaced with M^{**}. Thus, the proof is omitted due to space constraints.

Observe that the movements in M^{**} are evaluated in the following order: $m^{C',C''} \in M^{**} : |C'| + |C''| = s$, with s running from 1 to n. Now, let us introduce a parameter, m, which takes any value between 1 and $n-1$. This parameter controls a modified version of IDP, which only evaluates $m^{C',C''} \in M^{**} : |C'| + |C''| = s$ for $s = 1, \cdots, m, n$. This means that for every $C : |C| \in \{1, \cdots, m, n\}$, IDP computes $f(C)$ and $t(C)$ as usual, while for every $C : |C| \in \{m + 1, \cdots, n - 1\}$, it simply sets $f(C) = C$ and $t(C) = \{C\}$. This corresponds to the removal of the edges (in the integer partition graph) that involve splitting an integer $i : m < i < n$.

[8] An open-source implementation of IP, ODP and ODP-IP is publicly available at the following link: `https://github.com/trahwan/ODP-IP`

ACCEPTED MANUSCRIPT

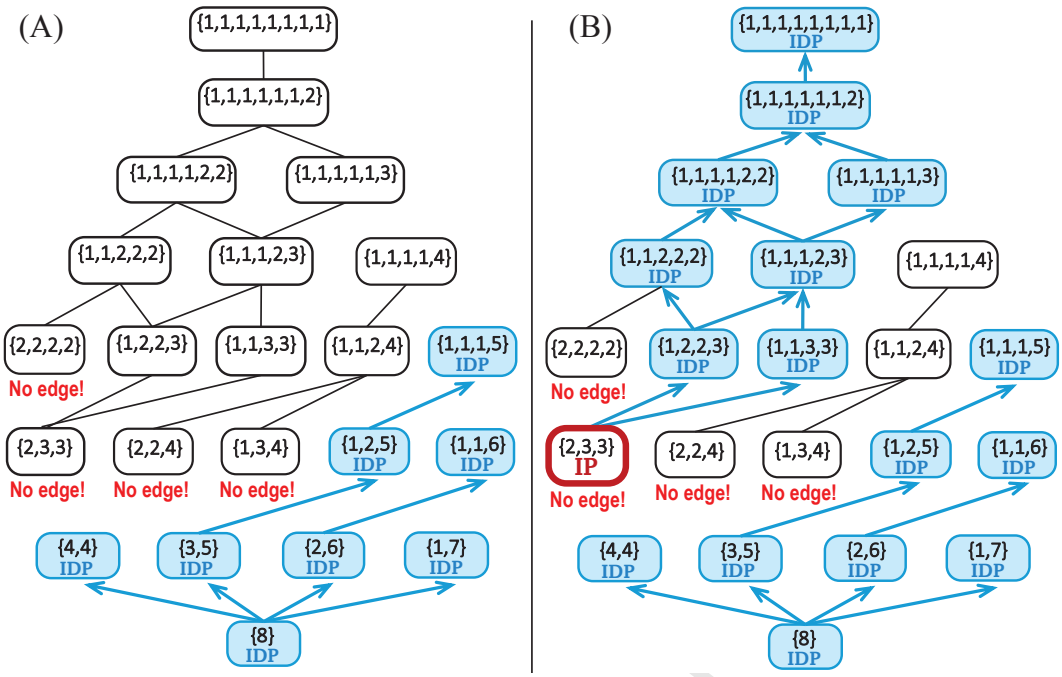


Figure 5: Illustration of how IDP-IP works for $n = 7$ and $m = 3$.

Importantly, even with the removal of those edges, IDP can still reach some nodes (i.e., subspaces) in the integer partition graph. For example, given $n = 8$ and $m = 3$, Figure 5(A) illustrates how, after removing every edge that involves splitting an integer $i : 3 < i < 8$, IDP can still reach some nodes in the integer partition graph. This means if IDP now keeps making the best movements upwards in the graph (out of all evaluated movements, starting from the bottom node), it would find *in linear time* the best coalition structure in all the subspaces highlighted in Figure 5(A). As for the remaining subspaces, IDP cannot search them as they are disconnected from the bottom component of the graph. Take, for example, the subspace $\Pi^A_{\{2,3,3\}}$. As can be seen in Figure 5(A), if only IDP could reach this subspace (i.e., if it could reach the integer partition $\{2, 3, 3\}$), it would be able to make further movements upwards. More precisely, IDP would be able to search *in linear time* all subspaces that correspond to integer partitions reachable from $\{2, 3, 3\}$. Fortunately, as we will show next, a modified version of IP could help IDP reach an otherwise unreachable subspace, such as $\Pi^A_{\{2,3,3\}}$ in Figure 5(A).

A key observation is that, for any $CS \in \Pi^A$ and any $m \in \{1, \cdots, n-1\}$, the value of the best coalition structure reachable from CS (using only the movements evaluated by IDP) is: $\sum_{C \in CS} f(C)$, where $f(C)$ is computed for every $C : |C| \in \{1, \cdots, m, n\}$ as usual, while for every $C : |C| \in \{m+1, \cdots, n-1\}$, we simply have $f(C) = C$. Based on this, out of all CS in some subspace Π^A_I, the most beneficial one for IDP to start moving from is a coalition structure in:

$$\arg\max_{CS \in \Pi^A_I} \sum_{C \in CS} f(C).$$

On the other hand, as we have seen in Section 5.2, IP can find a coalition structure in:

$$\arg\max_{CS \in \Pi^A_I} \sum_{C \in CS} v(C).$$

Based on the above, to find the best coalition structure in Π^A_I for IDP to start moving from, we can simply use a modified version of IP which searches Π^A_I as usual, except that the worth of every $CS \in \Pi^A_I$ is measured as $\sum_{C \in CS} f(C)$ instead of $\sum_{C \in CS} v(C)$. Then, IDP can proceed with the movements upward, and easily

ACCEPTED MANUSCRIPT

search all subspaces reachable from Π_I^A as illustrated in Figure 5(B). Similarly, every subspace that has no edge leading to it must first be searched by IP, followed by IDP.

When m is set to 1, every subspace will have no edge leading to it, and so must be searched by IP. In this case, the combination of IDP and IP becomes identical to IP. On the other hand, Rahwan and Jennings [63] proved that, when m is set to $\lfloor \frac{2\times n}{3} \rfloor$, every subspace will have an edge leading to it, and so IP will never be used. In this case, the combination of IDP and IP becomes identical to IDP. More importantly, by setting m anywhere between those two extremes, *one can determine how much of IDP, and how much of IP, to put in the mix*. Rahwan and Jennings showed that, for small values of m, IDP evaluates most of the edges in the graph (e.g., $> 99\%$) with ease. However, as m increases, the effectiveness of IDP drops exponentially, and it becomes more efficient to continue the search using IP.

A modified version of the above algorithm was subsequently proposed by Rahwan et al., which runs IDP and IP in parallel, thus enabling the partial outcome of IDP to be used to speed up the branch-and-bound technique of IP and to search multiple subspaces simultaneously [65, 64]. At the time of writing, we believe this is the fastest exact anytime algorithm in the literature (it still takes seconds to partition 30 agents given various random characteristic functions on a modern desktop computer, and is unlikely to solve problems of 40 agents or more).

A different algorithm was proposed by Service and Adams [80]. The basic idea is to compute, for every coalition C, the following values: $f^\dagger(C) = \max_{C' \subseteq C} v(C)$ and $t^\dagger(C) = \arg\max_{C' \subseteq C} v(C)$.[9] This computation is done using a dedicated dynamic programming algorithm that runs in $O(n2^n)$ time. After computing those values, a modified version of IDP is used. In particular, while iterating over all coalitions to compute f and t, the algorithm keeps track of the coalition that maximizes $f(C) + f^\dagger(A \backslash C)$. Let us denote this coalition by C^*. Now, suppose that this modified version of IDP was terminated prematurely, and that it had already computed f and t for every coalition of size $s \leq \lfloor \frac{n}{r} \rfloor$. Then:

- The best known partition of C^*, denoted by P^*, can be computed using $t(C^*)$ (just like the optimal partition of A was computed using $t(A)$ in Example 1). The value of this partition is $f(C^*)$.
- The best subset of $A \backslash C$ can obtained instantly from $t^\dagger(A \backslash C^*)$. The value of this subset is $f^\dagger(A \backslash C^*)$.

Based on the above, any coalition structure containing P^* and $t^\dagger(A \backslash C^*)$ has a value that is $\geq f(C^*) + f^\dagger(A \backslash C^*)$. This implies that any such coalition structure is guaranteed to be within a bound $\beta = r$ according to the following theorem:

Theorem 8.

$$f(C^*) + f^\dagger(A \backslash C^*) \geq \frac{V(CS^*)}{r}. \tag{10}$$

Proof. Let $\{P^1, \cdots, P^r\}$ be a partition of CS^* into r parts (i.e., each P^i is a possibly-empty set of coalitions).[10] Also, for each P^i, let $V(P^i) = \sum_{C \in P^i} v(C)$. Without loss of generality, let us assume that $V(P^1) \geq \cdots \geq V(P^r)$. Moreover, let the coalitions in P^i be denoted by $P^i_1, \cdots, P^i_{k_i}$, where $v(P^i_1) \geq \cdots \geq v(P^i_{k_i})$. The partition $\{P^1, \cdots, P^r\}$ is said to be *balanced* if $V(CS^*)$ is distributed equally among the parts, i.e., if $V(P^i) = V(CS^*)/r$ for all P^i. Based on this, let us define the *imbalance* of $\{P^1, \cdots, P^r\}$ as:

$$\sum_{i=1}^{r} \left(V(P^i) - \frac{V(CS^*)}{r} \right)^2.$$

[9] Note that $f(C)$ is the value of the best *partition* of C, while $f^\dagger(C)$ is the value of the best *subset* of C.

[10] Some parts will be empty if CS^* contains less than r coalitions

ACCEPTED MANUSCRIPT

Now, let $P = \{P^1, \cdots, P^r\}$ be a partition of CS^* that minimizes the imbalance. Then, we will prove that the following holds for any $P^i \in P$:

$$V(P^i) \geq V(P^1 \backslash \{P_1^1\}). \tag{11}$$

This can be proved by contradiction: if (11) does not hold, then it is possible to reduce the imbalance of P by replacing P^1 with $P^1 \backslash \{P_1^1\}$ and replacing P^i with $P^i \cup \{P_1^1\}$, which contradicts the definition of P.

Having proved the correctness of (11), let us now add $v(P_1^1)$ to both of its sides. We get: $v(P_1^1)+V(P^i) \geq V(P^1)$ for all $P^i \in P$. On the other hand, it is easy to see that the following holds no matter how small the imbalance of P is: $V(P^1) \geq \frac{V(CS^*)}{r}$. Thus, for all $P^i \in P$:

$$v(P_1^1) + V(P^i) \geq \frac{V(CS^*)}{r}. \tag{12}$$

Since there are n agents and r parts, there exists a part, say P^j, that contains at most $\lfloor \frac{n}{r} \rfloor$ agents. Let us denote the set of those agents by C^j, i.e., $C^j = \bigcup P^j$. Since $f(C^j)$ is the value of the optimal partition of C^j, then:

$$f(C^j) \geq V(P^j). \tag{13}$$

On the other hand, since the agents in P^j do not overlap with those in P^1, then P_1^1 must be a subset of $A \backslash C^j$, and so:

$$f^\dagger(A \backslash C^j) \geq V(P_1^1). \tag{14}$$

From (12), (13) and (14), we find that:

$$f^\dagger(A \backslash C^j) + f(C^j) \geq \frac{V(CS^*)}{r}. \tag{15}$$

Since C^* maximizes the right-hand side of (15), then (10) holds. □

Service and Adams showed that the total number of operations that is required to compute P^* is:

$$O\left(n2^n + \sqrt{n}\left(\frac{r(2(r-1))^{\frac{1}{r}}}{r-1}\right)^n\right).$$

For instance, to get the bounds $r = 2$, 3, 4, the algorithm runs in $O(\sqrt{n}2.83^n)$, $O(\sqrt{n}2.38^n)$, $O(\sqrt{n}2.09^n)$ time respectively. Moreover, to get any bound $r \geq 5$, the algorithm runs in $O(n2^n)$ time.

5.4. Integer Programming

A fundamentally different approach to those that we have seen so far is to formulate the coalition structure generation problem as an integer program. More specifically, let $C_1, C_2, \ldots, C_{2^n}$ denote the possible coalitions. Moreover, let z be an $n \times 2^n$ binary matrix, where every row represents an agent and every column represents a coalition, such that $z_{i,j} = 1$ if and only if $a_i \in C_j$. Finally, let us have 2^n decision variables, $x_1, x_2, \ldots, x_{2^n}$, where $x_j = 1$ corresponds to C_j being selected in the solution. The coalition structure generation problem can then be modeled as:

$$\begin{aligned} &\textit{Maximize} \sum_{j=1}^{2^n} v(C_j) \cdot x_j \\ &\textit{subject to} \sum_{j=1}^{2^n} z_{i,j} \cdot x_j = 1 \quad \text{for } i = 1, 2, \ldots, n \\ &\qquad x_j \in \{1, 0\} \qquad \text{for } j = 1, 2, \ldots, 2^n \end{aligned}$$

ACCEPTED MANUSCRIPT

With this formulation, it is possible to apply any integer programming solver. However, this approach has been shown to be inefficient, e.g., even an industrial-strength solver such as ILOG's CPLEX was shown to be significantly slower than both IDP and IP, and quickly runs out of memory given problems consisting of about 20 agents on a modern desktop computer [72].

6. Metaheuristic Algorithms

In all the algorithms that were presented so far, the focus was on finding an optimal solution, or a solution that is within a known bound from optimum. However, as the number of agents increases, the problem becomes too hard, and the only practical option is to use metaheuristic algorithms. Such algorithms do not guarantee that an optimal solution is ever found, nor do they provide any guarantees on the quality of their solutions. However, they can usually be applied with very large problems (e.g., consisting of thousands of agents). We now describe some of these algorithms.

Among the first researchers to propose algorithms for coalition formation were Shehory and Kraus [81, 82, 83, 84]. The algorithms that they proposed included a decentralized, greedy algorithm for coalition structure generation [84]. This algorithm ignores coalitions containing more than a certain number of agents. It returns a coalition structure CS that is constructed iteratively in a greedy manner; at every iteration, the best of all candidate coalitions is added to CS, where a candidate coalition is one that does not overlap with any of the coalitions that were added to CS in previous iterations. The search for the best candidate coalition is done in a distributed fashion; the agents negotiate over which one of them searches which coalitions.[11] Although this algorithm was rather simple, it made the first step towards addressing the need to develop algorithms that can produce a feasible coalition structure—a need that was first highlighted in the multi-agent community by Shehory and Kraus.

Another heuristic algorithm was later on proposed by Sen and Dutta [79]. This is a genetic algorithm that starts with an initial, randomly generated, set of coalition structures, called a *population*. After that, the algorithm repeats the following three steps: (1) evaluation, (2) selection, and (3) recombination. More specifically, the algorithm evaluates every member of the current population, selects members based on the outcome of the evaluation, and constructs new members from the selected ones by exchanging and/or modifying their contents.

A few years later, Keinänen [36] subsequently proposed an algorithm based on Simulated Annealing—a generic, stochastic local search technique. At every iteration, the algorithm moves from the current coalition structure to a coalition structure in its neighborhood, where neighborhoods can be defined using a variety of criteria. More specifically, the algorithm starts by generating a random coalition structure CS. Then, at every iteration, it samples a random coalition structure CS' in the neighborhood of CS. If CS' is better than CS, then the algorithm sets $CS = CS'$. Otherwise, it sets $CS = CS'$ with a probability $e^{(V(CS')-V(CS))/temp}$, where $temp$ is the *temperature* parameter that decreases after each iteration according to an *annealing schedule* $temp = \alpha temp$, where $0 < \alpha < 1$.

Another greedy algorithm was put forward by Mauro et al. [44] based on GRASP—a general purpose greedy technique which, after each iteration, performs a quick local search to try and improve its solution [27]. In the coalition structure generation version of GRASP, a coalition structure CS is formed iteratively as follows. Initially, the algorithm sets $CS := \emptyset$. Every iteration afterwards consists of two phases: a *constructive* phase and a *local search* phase.

- *The constructive phase* involves choosing an agent $a_i \in N \setminus \bigcup CS$, who will either be added to CS as a singleton, i.e., $CS := CS \cup \{\{a_i\}\}$, or added to some coalition $C_j \in CS$, i.e., $CS := (CS \setminus \{C_j\}) \cup$

[11] A significantly improved distribution mechanism was subsequently proposed in [61].

ACCEPTED MANUSCRIPT

$(\{C_j \cup \{a_i\}\})$. The candidate coalitions are then:

$$\bigcup_{a_i \in N \setminus \bigcup CS} \left\{ \{\{a_i\}\} \cup \bigcup_{C_j \in CS} \{C_j \cup \{a_i\}\} \right\}.$$

This set is sorted according to the impact of each candidate coalition on the value of CS. A coalition is then chosen uniformly at random from the top $x\%$ of the list, where x is a parameter that can be set by the user to control greediness; the smaller x is, the greedier the algorithm becomes.

- *The local search phase* explores different neighborhoods of CS. A *"neighbor"* of CS is defined as a coalition structure that results from one of the following five operations: (i) split a coalition in CS in two, (ii) merge two coalitions into one, (iii) swap two agents belonging to different coalitions, (iv) move an agent from one coalition to another, or (v) take an agent out of a coalition, and put that agent as a singleton.

Those two steps are repeated until $\cup_{C \in CS} C = A$. Furthermore, the whole process of forming a coalition structure is repeated over and over. This algorithm has been shown to work particularly well, with empirical results suggesting that it is the best metaheuristic algorithm for coalition structure generation to date.

We conclude with yet another greedy algorithm, called C-Link, which was recently proposed by Farinelli et al. [26]. It starts at the top node in the coalition structure graph, and then moves downwards in a greedy fashion. That, is, out of all downward movements from the current node, the algorithm picks the one that has the highest immediate reward, without taking into consideration the future consequences of this choice. The algorithm terminates when no downward movement is immediately beneficial. Since there are n levels, and the number of downward movements from any node is at most $n(n-1)/2$, the run time is $O(n^3)$. However, like other algorithms in this class, no guarantee can be placed on solution quality.

7. Coalition Structure Generation Under Compact Representations

An implicit assumption in the survey thus far has been that one can obtain the value $v(C)$ of a coalition C in unit time. This amounts to assuming that we have access to an oracle for v; we have remained silent until now on how the characteristic function v might actually be represented in a form suitable for practical computation. The oracle representation is, in fact, quite a strong assumption: it amounts to assuming that the optimization problem facing coalition C (i.e., the problem of C determining how to collectively act so as to obtain maximal benefit) can be solved in unit time. And yet, even with this strong assumption in place, as we noted earlier, the coalition structure generation problem is NP-hard. In this section, we survey algorithms that solve this problem given specific concrete representations for characteristic functions v. Such representations have been the subject of intense research over the past decade [16].

This section is structured as follows. Section 7.1 considers a setting in which the computation of a coalition's value requires solving a distributed constraint optimization problem. Section 7.2 considers a setting where every agent has a set of skills, and a coalition's value depends on the skills that are collectively possessed by its members. Section 7.3 considers a game in which some of the agents are identical. Section 7.4 considers a representation whereby only the coalitions with synergy are explicitly modelled. Finally, Section 7.5 considers a setting in which the characteristic function is expressed using the Marginal Contribution net (MC-net) representation.

7.1. Distributed Constraint Optimization

The Distributed Constraint Optimization Problem (DCOP) framework has recently become a popular approach for modelling cooperative agents [52]. In this framework: (1) each agent has a choice of actions, (2)

ACCEPTED MANUSCRIPT

the possibly-negative reward is determined by the combination of actions, and (3) the goal is for every agent to choose an action so as to maximize the sum of the rewards. Ueda et al. [89] considered the coalition structure generation problem where the multi-agent system is represented as one big DCOP, and every coalition's value is computed as the optimal solution of the DCOP among the members of that coalition.

At first glance, this might seem too computationally expensive since there are 2^n possible coalitions. Thus, to find the optimal coalition structure, one might need to solve 2^n instances of the NP-hard DCOP problem. Interestingly, however, Ueda et al. showed that the process of finding an optimal, or near optimal, coalition structure does not have to be divided into two independent stages: (1) computing all coalition values, and (2) finding an optimal combination of disjoint and exhaustive coalitions. Instead, the big DCOP that represents the multi-agent system can be modified such that those two stages are merged. This means the desired coalition structure can be obtained by solving a single, modified, DCOP.

The modification is controlled by a single parameter, called σ, which specifies the maximum number of coalitions that are allowed to contain more than one agent. We will call these *multi-agent* coalitions. The basic idea behind the modification is to change every agent's *domain*, i.e., set of possible actions. Specifically, every action d_j in the original domain is replaced by σ actions, $d_{j,1}, \ldots, d_{j,\sigma}$, where $d_{j,i}$ means that the agent performs action d_j while joining the i^{th} multi-agent coalition. The new domain also contains an action called "*independent*", which means that the agent acts independently. The modified DCOP can be solved using any existing algorithms that can obtain an optimal solution, e.g., ADOPT [52] or DPOP [59]. Assuming that the original number of possible actions per agent is d, the search space size for the original DCOP is d^n, while for the modified DCOP it is $(\sigma d + 1)^n$. Assuming that the value of each coalition is greater than, or equal to, zero, the following theorem implies that the optimal solution of the modified DCOP is within a bound $\beta = \left\lfloor \frac{n}{2} \right\rfloor / \sigma$ from optimum.

Theorem 9. *Let $\mathcal{I}_k^n \subseteq \mathcal{I}^n$ be the set in which every integer partition contains at most k integers that are greater than 1. Then, the best coalition structure in $\cup_{I \in \mathcal{I}_k^n} \Pi_I^A$ is within a bound $\beta = \left\lfloor \frac{n}{2} \right\rfloor / k$ from optimum.*

Proof. Assume that the optimal coalition structure, CS^*, does not belong to $\mathcal{I}_k^n$. In other words, assume that CS^* contains exactly l multi-agent coalitions, $C_1, \ldots, C_l$, where $l > k$. Furthermore, without loss of generality, assume that $v(C_1) \geq \cdots \geq v(C_l)$. This latter assumption implies that $\sum_{i=k+1}^{l} v(C_i) \leq \frac{l-k}{l} V(CS^*)$.

Now, consider a new coalition structure, CS', which is identical to CS^* except that every $C_i : i \in \{k+1, \ldots, l\}$ is split into single-agent coalitions. Clearly, CS' contains exactly k multi-agent coalitions, namely $C_1, \ldots, C_k$, and so $CS' \in \mathcal{I}_k^n$. Furthermore, the total value of $C_{k+1}, \ldots, C_l$ is at most $\frac{l-k}{l} V(CS^*)$, while the total value of the single-agent coalitions (that result from splitting $C_{k+1}, \ldots, C_l$) is at least 0. Thus, the maximum possible difference in value between CS' and CS^* is: $\frac{l-k}{l} V(CS^*)$. This implies that $\frac{V(CS^*)}{V(CS')} \leq \frac{l}{k}$. It remains to observe that $l \leq \left\lfloor \frac{n}{2} \right\rfloor$ since CS^* cannot possibly contain more than $\left\lfloor \frac{n}{2} \right\rfloor$ multi-agent coalitions. □

Ueda et al. [89] proved that, under the DCOP representation, the *decision version* of the coalition structure generation problem is NP-complete (this version involves checking whether a coalition structure exists whose value is greater than or equal to some given constant, q). The proof starts by highlighting the fact that, for a given coalition structure CS, checking whether $V(CS) \geq q$ can be done in polynomial time, meaning that the problem is in NP. The proof then proceeds based on a reduction from a Constraint Optimization Problem (COP) in which all rewards are non-negative, to a coalition structure generation problem consisting of the same variables and reward functions. Since all rewards are non-negative, the coalition structure containing the grand coalition is optimal. This implies that the solution to the COP is identical to the solution of the corresponding coalition structure generation problem. Finally, since the decision version of the original problem is NP-complete, the solution to the corresponding problem is also NP-complete.

ACCEPTED MANUSCRIPT

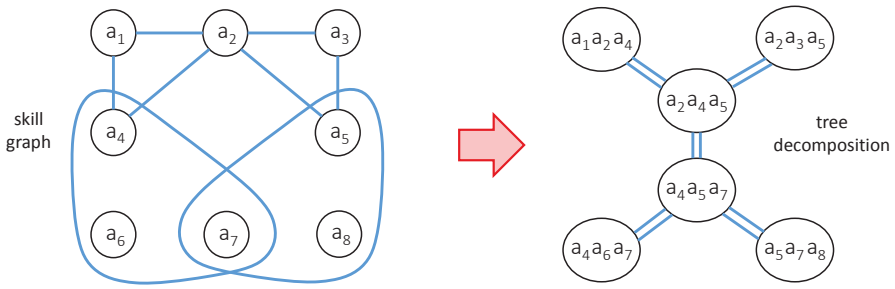


Figure 6: A *skill graph* and a possible *tree decomposition* with *treewidth* = 2.

7.2. Coalitional Games with Skills

In many settings, the value of a coalition can be defined in terms of the skills that are possessed by the agents. A simple representation formalism that is based on this idea was proposed by Ohta et al. [57]: there is a set of *skills* S, each agent $a_i \in A$ has a subset of the skills $S^{a_i} \subseteq S$, and there is a function $u : 2^S \to \mathbb{R}$ which for every subset of skills $S' \subseteq S$ specifies the payoff of a coalition that collectively possesses all the skills in S' and none of the skills in $S \setminus S'$. The value of a coalition $C \subseteq A$ is then:

$$v(C) = u(\cup_{a_i \in C} S^{a_i}).$$

Clearly, this representation is complete (i.e., it can represent any characteristic function game), as we can in the worst case identify each agent a_i with a unique skill s^{a_i} and set $u(S') = v(\{a_i \mid s^{a_i} \in S'\})$ for any subset S' of the skill set. However, this representation can be succinct, especially when the performance of each coalition can be expressed in terms of a small number of skills possessed by its members. A more structured representation was later on proposed by Bachrach and Rosenschein [7], where coalition values are expressed in terms of skills and tasks. Specifically, in addition to the set of skills S, there is a set of *tasks* Γ, and every task $\tau \in \Gamma$ has a *skill requirement* $S^\tau \subseteq S$ and a payoff. As before, each agent $a_i \in A$ has a set of skills $S^{a_i} \subseteq S$. A coalition $C \subseteq A$ *achieves* a task τ if it has all skills that are required for τ, i.e., if $S^\tau \subseteq \cup_{a_i \in C} S^{a_i}$. Finally, there is a *task value function* $F : 2^\Gamma \to \mathbb{R}$, which for every subset $\Gamma' \subseteq \Gamma$ of tasks specifies the payoff that can be obtained by a coalition that *achieves* all tasks in Γ'. The value of a coalition C is then given by

$$v(C) = F(\{\tau \mid S^\tau \subseteq \cup_{a_i \in C} S^{a_i}\}).$$

Bachrach et al. [6] considered the coalition structure generation problem in coalitional skill games. While this problem is, in general, very hard computationally, Bachrach et al. showed that it admits an efficient algorithm as long as the number of tasks and the *treewidth* of a certain associated hypergraph are small. To describe their algorithm, we need a few additional definitions.

Given a skill set S, its *skill graph* is a hypergraph $g = \langle V, E \rangle$ in which every agent is represented as a node, and every skill s_i is represented as a hyperedge $e_{s_i} \in E$ that connects all agents that possess this skill. The "complexity" of a hypergraph can be measured using the notion of *treewidth*. The following definitions are reproduced from [21], and an illustration is provided in Figure 6:

Definition 4. *Given a hypergraph $g = \langle V, E \rangle$, a* ***tree decomposition*** *of g is a pair $(Q, \mathbf{B})$, where $\mathbf{B}$ is a family of subsets of V (each such subset $B_i \in \mathbf{B}$ is called a* bag*), and Q is a tree whose node set is $\mathbf{B}$, such that: (1) for each $e \in E$, there is a bag $B_i \in \mathbf{B}$ such that $e \in B_i$; (2) for each $v_j \in V(g)$ the set $\{B_i \in \mathbf{B} | v_j \in B_i\}$ is non-empty and connected in Q. The* treewidth *of $(Q, \mathbf{B})$ is $\max_{B_i \in \mathbf{B}} |B_i| - 1$. The* treewidth *of g is the minimum treewidth of $(Q, \mathbf{B})$ over all possible tree decompositions $(Q, \mathbf{B})$ of g.*

ACCEPTED MANUSCRIPT

Let $CSG(m, w)$ be the class of all coalitional skill games where the number of tasks is at most m, and the treewidth of the corresponding skill graph is at most w. We will show that, for a fixed m and w, the coalition structure generation problem for a game in $CSG(m, w)$ can be solved in time polynomial in the number of agents n and the number of skills $|S|$ (but exponential in m and w).[12] The basic idea is as follows:

- **Step 1:** show that the coalition structure generation problem is equivalent to the problem of solving multiple *constraint satisfaction problems*[13] (CSP for short), whose underlying graph is the skill graph g;
- **Step 2:** show that each of the above CSPs can be solved by solving a *"dual"* CSP whose underlying graph is the tree decomposition of g. This is important because the tree decomposition of g is a tree, and CSPs whose underlying graph is a tree are solvable in polynomial time [75].

Next, we describe the above two steps in detail.

Step 1: Observe that a single task can be achieved multiple times by a single coalition structure CS. To be more precise, a task that requires a single skill which only x agents share can be achieved at most x times (this is when each one of those x agents appears in a different coalition in CS). Thus, if we denote by h' the largest number of agents sharing a single skill, then a coalition structure can achieve at most $h'|\Gamma|$ tasks. Based on this, we will define a *candidate task solution* as a set $\{\Gamma_i\}_{i=1}^h$ where each Γ_i is a subset of tasks, and $h \leq h'|\Gamma|$. For every coalition structure $CS = \{C_i\}_{i=1}^h$, we say that CS *achieves* $\{\Gamma_i\}_{i=1}^h$ if C_i achieves Γ_i for all $i = 1, \ldots, h$. We say that $\{\Gamma_i\}_{i=1}^h$ is *feasible* if there exists at least one coalition structure that achieves it. Clearly, the total value obtained by achieving these tasks is $\sum_{i=1}^h F(\Gamma_i)$. The problem of finding an optimal coalition structure is thus equivalent to the problem of finding a feasible $\{\Gamma_i\}_{i=1}^h$ that maximizes $\sum_{i=1}^h F(\Gamma_i)$. One way of doing this is to iterate over all possible choices of $\{\Gamma_i\}_{i=1}^h$ and, for each such choice, solve the following problem: *Determine whether it is feasible and, if so, find a coalition structure that achieves it.* This problem can be represented as a CSP whose underlying graph is the skill graph g, based on the observation that every coalition structure can be viewed as a *coloring* of the agents, where all agents with the same color form a coalition. Generally speaking, a CSP is a problem of finding an assignment of *variables* that satisfies some *constraints*. Here:

- the **variables** correspond to the agents;
- the **domain** (i.e., the possible values) of each variable (i.e., agent) consists of the possible colors (i.e., the possible coalitions that the agent can join);
- For each skill s, we have the following **constraint**: *For each $i = 1, \ldots, h$, if some task in Γ_i requires s, then at least one agent in C_i possesses s.*

Step 2: To solve the above "*primal*" CSP, we first check if the treewidth of g is bounded by w, and if so return a tree decomposition (this can be done in time polynomial in n and $|S|$, see [30]). Then, to solve the primal problem, we define a "*dual*" problem. This is another CSP whose underlying graph is the *tree decomposition* of g and:

- the **variables** correspond to the bags in the tree decomposition;
- the **domain** of every bag consists of the possible colorings of the agents in the bag. The size of this domain is $O((h)^{w+1})$ since every bag contains at most $w + 1$ agents, and every agent has h possible colors;

[12] See also the work by Aziz and de Keijzer [4] for further polynomial-time results for coalitional skill games with a constant number of skills.

[13] For more details on constraint satisfaction problems, see [75].

ACCEPTED MANUSCRIPT

- the **constraints** are of two types. The first prevents an agent from getting different colors in two neighboring bags. This, in turn, ensures that every agent gets the same color in *all* bags (due to the structure of the tree decomposition). The second type of constraints is exactly the same as the one in the primal problem (i.e., *if a skill is required for at least one task in Γ_i, then at least one agent in C_i possesses that skill*).

Note that a solution to the dual problem is in fact a valid solution to the primal problem. Since the underlying graph of the dual problem is a tree, it can be solved in time polynomial in n and $|S|$ [75, 6].

Having described how to handle skills in polynomial time under the Coalitional Skill Game representation of Bachrach and Rosenschein [7], we end this subsection by briefly mentioning a generalization of this representation, called the *Coalitional Skill Vector (CSV)* representation [86]. The key novelty here is that each agent is characterized not by a subset of available skills but by a *skill vector*. Formally, each agent a_i is assigned an $|S|$-dimensional vector of skills, $\mathbf{r}_i = (r_{i1}, \ldots, r_{i|S|})$, where S is the set of skills. The values in this vector reflect the level at which the agent has mastered the corresponding skills. In other words, while in Coalitional Skill Games the relation between an agent and a task is binary, i.e., any agent either possesses or does not possess a certain skill, in the CSV representation it is possible to express any intermediate state, i.e., that an agent has mastered any skill to a certain degree.

To define the value function for the CSV representation, we first define the skill vector of a coalition $C \subseteq A$ as $\mathbf{r}(C) = \sum_{a_i \in C} \mathbf{r}_i$. We also need $d(\mathbf{r}_1, \mathbf{r}_2)$ which is the distance between vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ in some norm L. The agents aim to achieve a set of goals and, in order to do so, they have to possess the level of skill required to achieve the different goals. More specifically, the list of goal-requirements is expressed as a set of skill vectors $\mathbf{G} \subseteq \mathbb{R}^{|S|}$. The value of coalition C is then:

$$v(\mathbf{r}(C)) = f(d(\mathbf{r}(C), \mathbf{G})),$$

where $d(\mathbf{r}(C), \mathbf{G}) = \min_{\mathbf{g} \in \mathbf{G}} d(\mathbf{r}(C), \mathbf{g})$ is the distance from $\mathbf{r}(C)$ to $\mathbf{G}$, and $f : \mathbb{R}^+ \cup \{0\} \to \mathbb{R}$ is the value function of d. The intuitive interpretation is as follows: the value of a coalition is a function over the distance between the coalition's skill vector and the set of goals $\mathbf{G}$. In this way, an agent that has relevant skills can pull its coalition closer towards a desired goal.

In addition to being *fully expressive*[14] and significantly more concise for certain classes of games than the classical representations, Tran-Thanh et al. [86] showed that the CSV representation facilitates the development of a comparatively efficient MIP-based algorithm to solve the coalition structure generation problem using linear relaxation (e.g., their algorithm minutes on a modern desktop computer to solve problems consisting of hundreds of agents and hundreds of constraints).

7.3. Agent-type Representation

Aziz and de Keijzer [4] and Ueda et al. [90] studied the coalition structure generation problem under the *agent-type* representation, where there is a set of *types*, $T = \{t^1, \ldots, t^{|T|}\}$, and a partition of A, denoted by $\{A^1, \ldots, A^{|T|}\}$, such that all agents in A^i are of type t^i. Intuitively, agents of the same type have the same contribution to any coalition they belong to. That is, for any $a_j, a_k \in A^i$, and any coalition C such that $a_j, a_k \notin C$, we have: $v(C \cup \{a_j\}) = v(C \cup \{a_k\})$. This implies that the value of any coalition depends solely on the number of agents that it contains from each type. More formally, for any coalition C, let us define the *coalition-type* of C as a vector, $\psi = \langle n^1, \ldots, n^{|T|} \rangle$, where each n^i denotes the number of agents in C that are of type t^i. Then, coalitions of the same coalition-type have the same value. This means the conventional characteristic function, $v : 2^A \to \mathbb{R}$, can be replaced with the more concise *type-based characteristic function*, $v^t : \Psi \to \mathbb{R}$, where Ψ is the set of all possible coalition-types, i.e., $\Psi = \{\langle n^1, \ldots, n^{|T|} \rangle : 0 \leq n^i \leq |A^i|\}$. This

[14] A representation is fully expressive if it is capable of representing any characteristic function game.

ACCEPTED MANUSCRIPT

function requires $O(|A|^{|T|})$ space, since $|\Psi| = (|A^1|+1) \times \cdots \times (|A^{|T|}|+1) < |A|^{|T|}$.[15] Now, let us introduce the following definitions:

Definition 5. *For every coalition-type, $\psi = \langle n^1, \ldots, n^{|T|}\rangle$, a **type-partition** of ψ is a set of coalition-types, $\lambda = \{\langle n_i^1, \ldots, n_i^{|T|}\rangle\}_{i=1}^{|\lambda|}$ such that the following holds: $\langle \sum_{i=1}^{|\lambda|} n_i^1, \ldots, \sum_{i=1}^{|\lambda|} n_i^{|T|}\rangle = \psi$. For example, $\{\langle 0,1,2\rangle, \langle 4,3,2\rangle\}$ is one of the possible type-partitions of $\langle 4,4,4\rangle$ because $\langle 0+4, 1+3, 2+2\rangle = \langle 4,4,4\rangle$.*

Definition 6. *The **value of a type-partition** $\lambda = \{\langle n_i^1, \ldots, n_i^{|T|}\rangle\}_{i=1}^{|\lambda|}$, is computed in the following way: $V^t(\psi) = \sum_{i=1}^{|\psi|} v^t(\langle n_i^1, \ldots, n_i^{|T|}\rangle)$. For example, $V^t(\{\langle 0,1,2\rangle, \langle 4,3,2\rangle\}) = v^t(\langle 0,1,2\rangle) + v^t(\langle 4,3,2\rangle\})$.*

Thus, while we typically deal with "coalitions" and "coalition structures", in an agent-type representation we deal with "coalition-types" and "type-partitions". The problem of finding an optimal coalition structure is then equivalent to that of finding an optimal type-partition of $\langle |A^1|, \ldots, |A^{|T|}|\rangle$. For example, if we have four types and five agents of each type, we need to find an optimal type-partition of $\langle 5,5,5,5\rangle$. Two dynamic programming algorithms were proposed to solve this problem; both run in $O(n^{2|T|})$ time [4, 90]. We will present the one given by Aziz and de Keijzer [4] since it is less heavy on notation.

For any coalition-type $\psi \in \Psi$, let $f^t(\psi)$ denote the value of the optimal type-partition of ψ. Then, we can compute $f^t(\psi)$ recursively as follows [4]:

$$f^t(\psi) = \begin{cases} 0 & \text{if } n^i = 0 \text{ for } 1 \le i \le |T| \\ \max\{f^t(\langle n^1 - x^1, \ldots, n^{|T|} - x^{|T|}\rangle) + v^t(\langle x^1, \ldots, x^{|T|}\rangle) & \\ \quad \mid x^i \le n^i \text{ for } i = 1, \ldots, T\} & \text{otherwise} \end{cases} \tag{16}$$

Based on this recursive formula, we can compute the optimal type-partition by dynamic programming. Specifically, the algorithm works by filling two tables, namely R and Q, each with an entry for every coalition-type. Entry $R[\langle n^1, \ldots, n^T\rangle]$ of table R stores an optimal type-partition of $\langle n^1, \ldots, n^T\rangle$, whereas entry $Q[\langle n^1, \ldots, n^T\rangle]$ of table Q stores the value of this type-partition. The algorithm fills out these tables using Equation (16), where "lower" entries are filled in first, i.e., if $m^i \le n^i$ for all $i = 1, \ldots, T$, then $\langle m^1, \ldots, m^T\rangle$ is dealt with before $\langle n^1, \ldots, n^T\rangle$. For each $\langle n^1, \ldots, n^{|T|}\rangle$, the algorithm finds a coalition-type $\langle x^1, \ldots, x^{|T|}\rangle$ that maximizes the max-expression of (16), and then sets

$$Q[\langle n^1, \ldots, n^{|T|}\rangle] = Q[\langle n^1 - x^1, \ldots, n^{|T|} - x^{|T|}\rangle] + v^t(\langle x^1, \ldots, x^{|T|}\rangle),$$
$$R[\langle n^1, \ldots, n^{|T|}\rangle] = R[\langle n^1 - x^1, \ldots, n^{|T|} - x^{|T|}\rangle], \langle x^1, \ldots, x^{|T|}\rangle.$$

By the end of this process, we compute $Q[\langle |A^1|, \ldots, |A^{|T|}|\rangle]$ and $R[\langle |A^1|, \ldots, |A^{|T|}|\rangle]$, which provide the solution to the coalition structure generation problem. Filling out each cell of R and Q requires $O(n^{|T|})$ operations, and the size of each table is $|\Psi| < n^{|T|}$. Hence, the algorithm runs in time $O(n^{2|T|})$. This algorithm has been shown to solve problems consisting of 100 agents and 5 agent-types in a matter of seconds on a modern desktop computer [90].

We conclude this subsection by mentioning the work of Rahwan et al. [70] who proposed a network flow-based representation, which conveniently models coalitional games with agent types. The authors showed that under this representation, the coalition structure generation problem in a task-based setting can be reformulated as a mathematical program related to the widely-studied production-transportation problem [87]. This produces approximate solutions to problems consisting of 5000 agents and 100 tasks in a matter of seconds on a modern desktop computer.

[15] For every type t^i, we have $(|A^i|+1)$ possible values because the number of agents that are of type t^i can be 0 or 1 or ... or $|A^i|$.

ACCEPTED MANUSCRIPT

7.4. Synergy Coalition Groups

Another concise representation under which the coalition structure generation problem was considered is *Synergy Coalition Groups (SCG)* [17]. An interesting property of this representation is that the very basic operation of computing a coalition's value may itself require solving a small coalition structure generation problem. In more detail, the SCG representation is built upon the observation that in certain settings cooperation within many coalitions does not produce any value added. In other words, there is no synergy achieved by creating such coalitions. Thus, the rational choice of the agents involved in such coalitions is to work in smaller (and possibly more effective) sub-coalitions. In such settings, instead of explicitly representing the values of all possible coalitions, the main idea behind SCG is to only represent the values of coalitions in which there is synergy among the members. More formally, SCG consists of a set of pairs of the form: $(C, v(C))$, where C is a coalition the formation of which produces synergy.[16] If there is no synergy from the creation of a coalition, then it is assumed that agents in such a coalition partition themselves into sub-coalitions in the best possible way. Formally, if $(C, v(C)) \notin$ SCG then $v(C) = \max_{CS \in \Pi^C} \sum_{C_i \in CS} v(C_i)$, where the following two conditions are met:

- for all the C_i it holds that $(C_i, v(C_i)) \in$ SCG; and
- for all $CS' \subseteq CS$, where $|CS'| \geq 2$, it holds that $\left(\bigcup_{C_i \in CS'} C_i, v\left(\bigcup_{C_i \in CS'} C_i\right)\right)$ is not an element of SCG.

Intuitively, the latter condition states that, while computing the value of a coalition C that does not appear in SCG, we cannot choose $CS \in \Pi^C$ such that union of any of the coalitions in CS appears in SCG. Ohta et al. [56] showed that this condition is needed to ensure that the SCG representation is fully expressive.

Example 2. *Let* $A = \{a_1, a_2, a_3, a_4, a_5\}$ *and* $SCG = \{(\{a_1\}, 0), (\{a_2\}, 0), (\{a_3\}, 1), (\{a_4\}, 2), (\{a_5\}, 1), (\{a_1, a_2\}, 3), (\{a_1, a_2, a_3\}, 3)\}$. *In this case,* $v(\{a_4, a_5\}) = v(\{a_4\}) + v(\{a_5\}) = 3$, *and* $v(\{a_1, a_2, a_3, a_4, a_5\}) = v(\{a_1, a_2, a_3\}) + v(\{a_4\}) + v(\{a_5\}) = 6$. *For* $v(\{a_1, a_2, a_3, a_4, a_5\})$, *we cannot use* $v(\{a_1, a_2\} + v(\{a_3\}) + v(\{a_4\}) + v(\{a_5\}) = 7$, *because* $\{a_1, a_2\} \cup \{a_3\} = \{a_1, a_2, a_3\}$ *appears in SCG.*

Ohta et al. [56] proved that following theorem:

Theorem 10. *There exists an optimal coalition structure* CS^* *such that* $\forall C_i \in CS^*,\ (C_i, v(C_i)) \in SCG$.

Proof. If there existed some CS^* such that $V(CS^*)$ was strictly greater than any coalition structure whose coalitions are all in SCG, then it would be possible to construct some CS from CS^* by replacing all coalitions $C_i \in CS^* : C_i \notin$ SCG by their best possible partitions into sub-coalitions. By doing so, the value of CS would be at least equal to the value of CS^*—we obtain a contradiction. □

Interestingly, due to Theorem 10, the coalition structure generation problem under the SCG representation becomes equivalent to a *weighted set packing problem*, also known as the *winner-determination problem* in combinatorial auctions [77]. As such, all the results for these two problems are directly applicable in our context. In particular:

Theorem 11 ([77]). *Under the SCG representation, the optimal coalition structure generation problem is NP-hard. Moreover, unless NP = ZPP, there exists no polynomial-time* $O(|SCG|^{1-\epsilon})$ *approximation algorithm for any* $\epsilon > 0$.

[16] To avoid senseless cases without feasible partitions, it is assumed that $(\{a_i\}, 0) \in$ SCG whenever $\{a_i\}$ does not receive a value elsewhere in SCG.

ACCEPTED MANUSCRIPT

Proof. Follows directly from the corresponding results on the winner determination problem [77] and on the maximum independent set problem [97]. □

Also the MIP formulation directly corresponds to a standard winner determination formulation:

$$\max \sum_{(C,v(C))\in \mathrm{SCG}} v(C)\cdot x(S) \quad \text{subject to: } \forall a_i \in A,\ \sum_{C \ni a_i} x(C) = 1,$$

where $x(C)$ is 1 if C is included in CS^*, and 0 otherwise. Ohta et al. [56] report solving problems consisting of around a hundred agents, and a few hundred pairs in the corresponding SCG, in a few milliseconds on a modern desktop computer.

In the spirit of a recent literature on representing coalitional games using decision diagrams [10, 1], Sakurai et al. [76] proposed a representation that uses Multi-Terminal Zero-suppressed Decision Diagrams (ZDD, [51]) to reduce the size of the SCG representation. The resulting combined ZDD-SCG representation also admits an MIP formulation to solve the coalition structure generation problem.

7.5. Marginal Contribution Nets

Ieong and Shoham [34] proposed a concise representation called *Marginal contribution nets*, or *MC-nets*, where a game is described by a collection of rules $\mathcal{R}$. Each rule $r \in \mathcal{R}$ is of the form $\mathcal{B}_r \to \vartheta_r$, where $\mathcal{B}_r$ is a Boolean formula over a set of variables $\{b_1, \ldots, b_n\}$ and ϑ_r is a real value. We say that a rule $r \in \mathcal{R}$ is *applicable* to coalition C if $\mathcal{B}_r$ is satisfied by the following truth assignment: $b_i =$ **true** if $a_i \in C$ and $b_i =$ **false** if $a_i \notin C$. Let $\mathcal{R}^C$ denote the set of rules that are applicable to C. Then, the characteristic function of the game described by $\mathcal{R} = \{\mathcal{B}_1 \to \vartheta_1, \ldots, \mathcal{B}_k \to \vartheta_k\}$ is computed as follows:

$$v(C) = \sum_{r\in\mathcal{R}^C} \vartheta_r.$$

Example 3. *The MC-net that consists of the rules* $\mathcal{R} = \{b_2 \to 3, b_1 \wedge b_2 \to 5\}$*, corresponds to a coalitional game* $G = (A, v)$*, where* $A = \{a_1, a_2\}$*,* $v(\{a_1\}) = 0$*,* $v(\{a_2\}) = 2$*,* $v(\{a_1, a_2\}) = 8$*. The intuitive interpretation of the rules in* $\mathcal{R}$ *is as follows: whenever* a_2 *is in a coalition,* C*, it improves the performance of* C *by 3 units of utility, and whenever* a_1 *and* a_2 *are together in a coalition, they improve its performance by* 5 *units of utility.*

An MC-net is said to be *basic* if the left-hand side of any rule is a conjunction of literals, i.e., a conjunction of variables or their negations. In this case, we write a rule $r \in \mathcal{R}$ as $(P_r, N_r) \to \vartheta_r$, where P_r and N_r are the sets of positive and negative literals, respectively. Thus, r is applicable to coalition C if C contains every agent in P_r and none of the agents in N_r. It is not hard to see that any coalitional game $G = (A, v)$ with $|A| = n$ can be represented by a basic MC-net with $2^n - 1$ rules: for each coalition $C \subseteq A$ we create a rule

$$(\wedge_{i:a_i\in C} b_i) \bigwedge (\wedge_{i:a_i\notin C} \neg b_i) \to v(C).$$

However, many interesting games admit a more succinct representation (i.e., they do not require as many as $2^n - 1$ rules), especially if we allow MC-nets that are not basic.

Ohta et al. [56] studied the coalition structure generation problem given a restricted class of basic MC-nets, where $P_r \neq \emptyset$ and $\vartheta_r > 0$ for every r.[17] In particular, they define a set of rules $\mathcal{R}' \subseteq \mathcal{R}$ to be *feasible* if all the rules in $\mathcal{R}'$ are applicable *at the same time* to some coalition structure. In other words, $\mathcal{R}'$ is feasible if there exists a coalition structure CS such that every rule $r \in \mathcal{R}'$ is applicable to some $C \in CS$.

[17] The authors showed that any characteristic function can be represented by such a restricted MC-net.

ACCEPTED MANUSCRIPT

The problem of finding an optimal coalition structure is then equivalent to the problem of finding a feasible set of rules $\mathcal{R}'$ such that $\sum_{r\in\mathcal{R}'}\vartheta_r$ is maximized. While this problem is NP-hard, Ohta et al. showed that it admits a mixed integer programming (MIP) formulation. The remainder of this section will first explain the intuition behind the MIP, and then explain why (and how) it should be modified in order to handle rules with negative values. After that, it will briefly discuss the work by Liao et al. [40] that builds upon MC-Nets to solve the coalition structure generation problem with an off-the-shelve MaxSAT solver. Finally, it will consider the *Induced Subgraph* [22] representation, which is basically a simplified, graph-based version of MC-Nets.

The MIP of Ohta et al. [56] is based on the observation that, for any two rules r, r', the possible relations between r and r' can be classified into the following four cases:

- **Incompatible (IC):** This is when $P_r \cap P_{r'} \neq \emptyset$ and ($P_r \cap N_{r'} \neq \emptyset$ or $P_{r'} \cap N_r \neq \emptyset$). For example, $(\{a_1, a_2\}, \emptyset) \rightarrow \vartheta_1$ and $(\{a_2, a_3\}, \{a_1\}) \rightarrow \vartheta_2$ are not applicable at the same time, because the first requires a_1 and a_2 to appear together in a coalition, while the second required a_2 and a_3 to appear together in a coalition that does not contain a_1.

- **Compatible on the same coalition (CS):** This is when $P_r \cap P_{r'} \neq \emptyset$ and $P_r \cap N_{r'} = P_{r'} \cap N_r = \emptyset$. For example, $(\{a_1, a_2\}, \emptyset) \rightarrow \vartheta_1$ and $(\{a_2, a_3\}, \{a_4\}) \rightarrow \vartheta_2$ are applicable at the same time in some coalition structure CS as long as there exists $C \in CS$ such that $\{a_1, a_2, a_3\} \subseteq C$ and $a_4 \notin C$. Note that both rules apply to the same coalition.

- **Compatible on different coalitions (CD):** This is when $P_r \cap P_{r'} = \emptyset$ and ($P_r \cap N_{r'} \neq \emptyset$ or $P_{r'} \cap N_r \neq \emptyset$). For example, $(\{a_1, a_2\}, \emptyset) \rightarrow \vartheta_1$ and $(\{a_3, a_4\}, \{a_1\}) \rightarrow \vartheta_2$ are applicable at the same time in some CS as long as a_1, a_2 appear in a coalition $C \in CS$ and a_3, a_4 appear in a different coalition $C' \in CS$.

- **Independent (ID):** This is when $P_r \cap P_{r'} = P_r \cap N_{r'} = P_{r'} \cap N_r = \emptyset$.

Consider a graphical representation of an MC-net in which every node is a rule, and between any two nodes there exists an edge whose type is one of the four cases described above (see, e.g., Figure 7). Then, the following holds:

Theorem 12. *A set of rules $\mathcal{R}'$ is feasible if and only if (1) it includes no pair of rules that are connected by an edge of type* **IC***, and (2) for any two rules $r_1, r_2 \in \mathcal{R}'$, if r_1 can be reached from r_2 via a series of edges of type* **CS***, then the edge (r_1, r_2) must not be of type* **CD***.*

To understand the intuition behind the proof, consider the example from Figure 7. Here, r_1 and r_2 are connected by an edge of type **CS**. Thus, they must be applicable to a single coalition in CS, say C', such that $P_1 \cup P_2 \subseteq C'$. Similarly, an edge of type **CS** connects r_2 and r_3, and so they must be applicable to a single coalition in CS, say C'', such that $P_2 \cup P_3 \subseteq C''$. Now, since $P_1 \cup P_2$ overlaps with $P_2 \cup P_3$, and since the coalitions in CS are pairwise disjoint, we must have $C' = C''$. This means that r_1, r_2, r_3 must all be applicable to the same coalition. In other words, the edge between r_1 and r_3 must *not* be of the type **IC** or **CD**. However, in our example we happen to have an edge of type **CD** between r_1 and r_3. Therefore, any rule set containing r_1, r_2, r_3 is not feasible.

The above example can be generalized to any set of rules, say $r_1, r_2, \ldots, r_m$, where an edge of type **CS** connects every $r_{i<m}$ to r_{i+1}; these rules must all be applicable to a single coalition, which contains $P_1 \cup \cdots \cup P_m$.

Based on Theorem 12, Ohta et al. proposed the following MIP formulation.

$$\max \quad \sum_{r\in R} \vartheta_r \cdot x_r \quad \text{subject to:}$$

ACCEPTED MANUSCRIPT

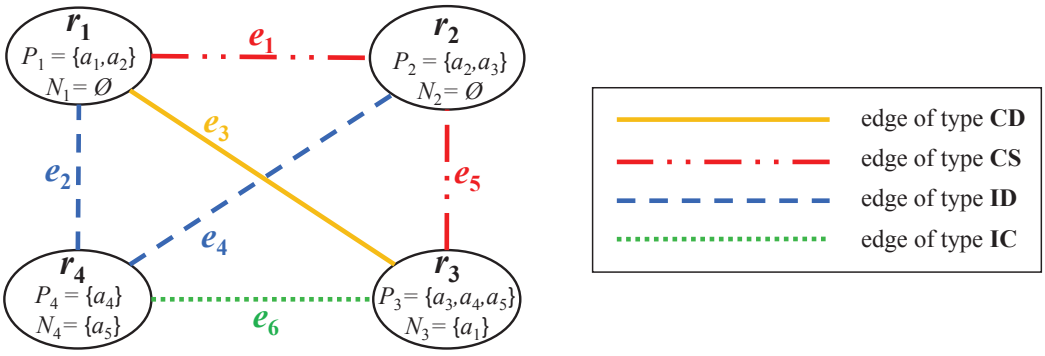


Figure 7: A four-rule example of the graphical representation of MC-nets.

$$
\begin{aligned}
x_{r_i} + x_{r_j} &\leq 1 \quad \text{for each edge } (r_i, r_j) \text{ of type } \mathbf{IC} && (17)\\
y^e_{r_i} &= 0 \quad \text{for each edge } e = (r_i, r_j) \text{ of type } \mathbf{CD} \text{ with } j > i && (18)\\
y^e_{r_j} &\geq 1 \quad \text{for each edge } e = (r_i, r_j) \text{ of type } \mathbf{CD} \text{ with } j > i && (19)\\
y^e_{r_k} &\leq y^e_{r_\ell} + (1 - x_{r_k}) + (1 - x_{r_\ell}) \\
&\qquad \text{for each edge } e \text{ of type } \mathbf{CD}, \text{ and each edge } (r_k, r_\ell) \text{ of type } \mathbf{CS} && (20)\\
y^e_{r_\ell} &\leq y^e_{r_k} + (1 - x_{r_k}) + (1 - x_{r_\ell}) \\
&\qquad \text{for each edge } e \text{ of type } \mathbf{CD}, \text{ and each edge } (r_k, r_\ell) \text{ of type } \mathbf{CS} && (21)\\
x_r &\in \{0, 1\} \text{ for each } r \in R
\end{aligned}
$$

Here, we have a binary variable x_r for every rule r, where $x_r = 1$ means that r is selected in the solution. Thus, condition (1) of Theorem 12 is enforced by the constraint (17), which ensures that two rules connected by an edge of type **IC** are never selected at the same time. What remains is to enforce condition (2) of the theorem. To do this, for every edge e of type **CD**, and for every rule r that is an endpoint of e or an endpoint of some other edge of type **CS**, we define a variable y^e_r. For example, since e_3 in Figure 7 is an edge of type **CD**, we define the variables $y^{e_3}_{r_1}, y^{e_3}_{r_2}, y^{e_3}_{r_3}$ (because every rule r_1, r_2, r_3 is either an endpoint of e_3 or an endpoint of some edge of type **CS**). These variables are used in constraints (18)–(21) to enforce condition (2) of Theorem 12. In more detail, for every edge $e = (r_i, r_j)$ of type **CD**, we have:

- Constraints (18) and (19) ensure that $y^e_{r_i} \neq y^e_{r_j}$.
- Constraints (20) and (21) ensure that, for any edge (r_k, r_ℓ) of type **CS**, we have: $y^e_{r_k} = y^e_{r_\ell}$ whenever r_k and r_ℓ are selected (i.e., whenever $x_k = 1$ and $x_\ell = 1$).

This way, if r_i and r_j are connected by a path of edges of type **CS**, then the rules on this path cannot be selected at the same time. In Figure 7, for example, $e_3 = (r_1, r_3)$ is an edge of type **CD**, and r_1 is connected to r_3 by a path of edges of type **CS**. Therefore, the rules on this path, i.e., r_1, r_2, r_3, cannot be selected at the same time. This is enforced by constraints (18) and (19), which ensure that $y^{e_3}_{r_1} \neq y^{e_3}_{r_3}$, as well as the constraints (20) and (21), which ensure that $y^{e_3}_{r_1} = y^{e_3}_{r_2}$ (whenever r_1 and r_2 are selected) and $y^{e_3}_{r_2} = y^{e_3}_{r_3}$ (whenever r_2 and r_3 are selected). By enforcing both conditions of Theorem 12, every solution of the MIP formulation is guaranteed to be a feasible rule set.

Ueda et al. [88] highlighted the fact that the above MIP cannot handle rules with negative values, which is unfortunate since the incorporation of negative-valued rules can potentially make the MC-net representation exponentially smaller for some games. Recall that the value of a coalition in MC-nets is the sum of *all* the values of the rules that are applicable to that coalition. Therefore, when evaluating a coalition structure, CS, the rules that are applicable to the coalitions in CS *must all be taken into consideration*, even if some of those

ACCEPTED MANUSCRIPT

rules have a negative value. However, the constraints in the above MIP only specify the rules that *cannot all be taken into consideration*. With such constraints, the solver will never incorporate a negative-valued rule, because doing so decreases the solution quality, and is *not necessary as far as the solver is concerned*. To handle such rules correctly, the authors proposed (and compared) three alternative modifications to the above MIP. Here, we explain the most effective one according to their evaluation. Basically, assume without loss of generality that every negative-valued rule is of the form:

$$r_n : a_1 \wedge \cdots \wedge a_k \wedge \neg a_{k+1} \cdots \wedge \neg a_m \rightarrow v_n \quad (\text{where } v_n < 0).$$

For every such rule, the authors add to the MIP the following *dummy* rules: $r_{n,i} : a_1 \wedge \neg a_i \rightarrow 0$ for every $i = 2, \ldots, k$ and $r_{n,i} : a_1 \wedge a_i \rightarrow 0$ for every $i = k+1, \ldots, m$. They also add a binary variable x_n for the rule r_n, and a binary variable $x_{n,i}$ for every dummy rule $r_{n,i}$. Importantly, the authors proved that r_n is applicable to a coalition, C, if and only if every $r_{n,i}$ is not applicable to C. Therefore, to force the solver to select the negative-valued rule r_n whenever it is applicable, it suffices to add the following constraint to the MIP: $x_{r_n} + x_{r_{n,1}} + \cdots + x_{r_{n,m}} \geq 1$. This implies that if every $r_{n,i}$ were not selected in the solution, then r_n *must* be selected in the solution.

We now turn our attention to the work by Liao et al. [40], who built upon the results by Ohta et al. [56] and Ueda et al. [88] to tackle the coalition structure generation problem with off-the-shelf MaxSAT solvers. To this end, the relations **IC**, **CS**, **CD**, and **ID** were used to encode MC-nets into propositional Boolean logic so that the coalition structure generation problem becomes the Weighted Partial MaxSAT problem [74]. Simulation results presented by Liao et al. [40] suggest that the MaxSAT solver SAT4j [11] returns a solution significantly faster than ILOG's CPLEX used by Ohta et al. [56] and Ueda et al. [88] to solve the MIP formulations outlined above (e.g., it solves instances of 150 agents and 150 rules in a few seconds on a modern desktop computer, and seems to scale very well).

Finally, we briefly mention the work by Bachrach et al. [5] who considered the coalition structure generation problem under the *induced-subgraph* representation of Deng and Papadimitriou [22] which, in fact, can be interpreted as a simplified version of the MC-Nets representation. In more detail, the induced-subgraph representation is built upon weighted graphs, where a node is interpreted as an agent, and the (possibly negative) weight of an edge is interpreted as the value of cooperation between the two connected agents.[18] The value of any coalition is then defined as the sum of weights of all its internal edges, i.e., the weights of edges belonging to a subgraph induced by members of the coalition. It is not difficult to see that any weighted edge can be readily translated to an MC-Net rule. In particular, an edge that connects two agents a_i and a_j can be written as:

$$a_i \wedge a_j \rightarrow \text{ weight of the edge.}$$

On the other hand, a self-loop of an agent a_i becomes:

$$a_i \rightarrow \text{ weight of the self loop.}$$

Example 4. *A three-agent example of this formalism from Michalak et al. [48] is presented in Figure 8. In this game, values of coalitions* $\{v_1\} = 1$, $\{v_2\} = 1$, $\{v_3\} = 1$, $\{v_1, v_2\} = 1 + 1 + 2$, $\{v_1, v_3\} = 1 + 1 + 3$, $\{v_2, v_3\} = 1 + 1 + 4$, *and* $\{v_1, v_2, v3\} = 1 + 1 + 1 + 2 + 3 + 4$. *This game can be represented with the following MC-Net rules:* $a_1 \rightarrow 1$, $a_2 \rightarrow 1$, $a_3 \rightarrow 1$, $a_1 \wedge a_2 \rightarrow 2$, $a_1 \wedge a_3 \rightarrow 4$, *and* $a_2 \wedge a_3 \rightarrow 3$.

Bachrach et al. [5] proved that the coalition structure generation problem under the induced-subgraph representation is NP-complete. The proof is based on a reduction from the *independent-set* problem, which involves testing whether there exists a subset of k nodes with no edges between them in a graph G. More

[18] Self-loops are allowed.

ACCEPTED MANUSCRIPT

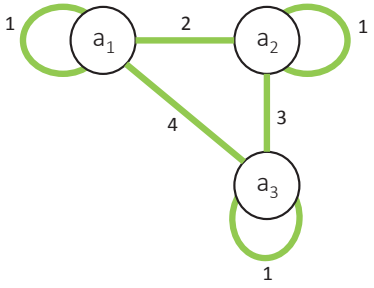


Figure 8: The induced-subgraph representation of a sample coalitional game of 3 agents.

specifically, the authors reduce this problem to a coalition structure generation problem, by first setting all the weights in G to be $-k$ each. After that, they add to G one additional node, and connect it to every other node in G with an edge whose weight is 1. One can see how a coalition structure whose value is k exists if and only if there exists a set of k nodes with no edges between them in G.

On the up side, the authors proposed constant factor approximation algorithms for planar, minor-free and bounded degree graphs.

8. Constraints on the Set of Feasible Coalitions

So far, we assumed that agents can split into teams in any way they like. However, in practice some coalition structures may be inadmissible due to various constraints present in the problem domain. This section discusses some of the works have addressed this issue. More specifically, Section 8.1 focuses on a setting where constraints are expressed in terms of subsets of agents whose presence in a coalition is desirable, and other subsets whose presence in a coalition is prohibited. Section 8.2, on the other hand, focuses on a setting where we are given a graph specifying which agents are connected to one another, and the feasible coalitions are those that induce a connected subgraph.

8.1. Constrained Coalition Formation Model

Rahwan et al. [68] proposed the *constrained coalition formation (CCF)* framework, which allows one to impose constraints on the coalition structures that can be formed. Formally, a CCF game is a tuple $\langle A, \ddot{\Pi}^A, v\rangle$ where A is the set of agents, $\ddot{\Pi}^A$ is the set of coalition structures that are *feasible* (i.e., allowed to form), and v is the characteristic function that assigns a real value to every coalition that appears in some feasible coalition structure. Note that, in the general case, the notion of feasibility is defined for coalition structures rather than coalitions. For instance, if $A = \{a_1, a_2, a_3, a_4\}$ and we define $\ddot{\Pi}^A$ as the set of all coalition structures in which all coalitions have the same size, then the coalition structure $\{\{a_1\}, \{a_2\}, \{a_3, a_4\}\}$ is not feasible, even though each of its component coalitions may be a part of a feasible coalition structure. There are, however, many settings of interest where the constraints implied by $\ddot{\Pi}^A$ can be reduced to constraints on individual coalitions. More formally, a CCF game $G = \langle A, \ddot{\Pi}^A, v\rangle$ is *locally constrained* if there exists a set of coalitions $\ddot{\mathcal{C}}^A \subseteq 2^A$ such that $\ddot{\Pi}^A = \{CS \in \Pi^A \mid CS \subseteq \ddot{\mathcal{C}}^A\}$. We will refer to the coalitions in $\ddot{\mathcal{C}}^A$ as *feasible coalitions*.

To represent the constraints succinctly, the authors propose the use of propositional logic. More formally, let $B_A = \{b_i \mid a_i \in A\}$ be a set of Boolean variables, and let ϕ be a propositional formula over B_A, constructed using the usual classical connectives $(\wedge, \vee, \neg, \rightarrow, \ldots)$. A coalition C *satisfies* ϕ if ϕ is satisfied under the truth assignment that sets all b_i with $a_i \in C$ to **true** and all b_i with $a_i \notin C$ to **false**. For example, any coalition containing a_1 and a_2 satisfies $\phi = b_1 \wedge b_2$. It has been shown that this language can represent any *locally constrained* CCF game, and that it can be extended so as to represent any CCF game [68].

ACCEPTED MANUSCRIPT

The authors defined a natural subclass of *locally constrained* CCF games, which they called *basic* CCF games. Intuitively, the constraints in a *basic* CCF game are expressed in the form of (1) *sizes* of coalitions that are allowed to form, and (2) *subsets* of agents whose presence in any coalition is viewed as desirable/prohibited. The constraints of the former type are called *size constraints*, denoted by $\mathcal{S} \subseteq \{1, \ldots, n\}$. As for the latter type of constraints, the desirable subsets of agents are called *positive constraints*, denoted by $\mathcal{P} \subseteq 2^A$, while the prohibited subsets are called *negative constraints*, denoted by $\mathcal{N} \subseteq 2^A$. Thus, a coalition C is feasible if (1) its size is permitted, i.e., $|C| \in \mathcal{S}$, and (2) it contains at least one of the desirable subsets and none of the prohibited ones, i.e., $\exists P \in \mathcal{P} : P \subseteq C$ and $\forall N \in \mathcal{N}, N \not\subseteq C$. We will denote the set of all such feasible coalitions by $c(A, \mathcal{P}, \mathcal{N}, \mathcal{S})$.

The set of constraints in a basic CCF game can be transformed into another, isomorphic set so as to facilitate both the process of identifying feasible coalitions and the process of searching for an optimal, feasible coalition structure. This transformation is based on the observation that, for any agent $a_i \in A$, the coalitions in $c(A, \mathcal{P}, \mathcal{N}, \mathcal{S})$ can be divided into:

- coalitions that contain a_i. For those, any constraint $P \in \mathcal{P} : a_i \in P$ has the same effect as $P \backslash \{a_i\}$. Similarly, any $N \in \mathcal{N} : a_i \in N$ has the same effect as $N \backslash \{a_i\}$. Thus, every such P or N can be replaced with $P \backslash \{a_i\}$ or $N \backslash \{a_i\}$, respectively;
- coalitions that do not contain a_i. For those, every positive or negative constraint that contains a_i has no effect, and so can be removed.

Thus, the problem of dealing with $c(A, \mathcal{P}, \mathcal{N}, \mathcal{S})$ can be replaced with two simpler problems; we can then apply the same procedure recursively. This can be visualized as a tree, where the root is $c(A, \mathcal{P}, \mathcal{N}, \mathcal{S})$, and each node has two outgoing edges; one leads to a subtree containing some agent a_i and the other leads to a subtree that does not contain a_i. As we go further down the tree, the problem becomes simpler and simpler, until one of the following two cases is reached: (1) a case where one can easily generate the feasible coalitions, which is called a *base case*, or (2) a case where one can easily verify that there are no feasible coalitions (i.e., the constraints cannot be satisfied), which we call an *impossible case*. This is illustrated in Figure 9 (A), where the edge-labels a_i and $\overline{a_i}$ indicate whether the branch contains, or does not contain, a_i respectively. By generating the feasible coalitions in all base cases, one ends up with all the feasible coalitions in $c(A, \mathcal{P}, \mathcal{N}, \mathcal{S})$.

The tree structure described above also facilitates the search for an optimal, feasible, coalition structure. Indeed, observe that every such tree contains exactly one path that (1) starts with the root, (2) ends with a leaf, and (3) consists of edges that are each labelled with $\overline{a_i}$ for some $a_i \in A$. In Figure 9, for example, this path is the one connecting $c(A, \mathcal{P}, \mathcal{N}, \mathcal{S})$ to $baseCase_{15}$. Now, let us denote by A^* the sequence of agents that appear in the labels of this path. For instance, in Figure 9, we have $A^* = \langle a_5, a_2, a_1, a_8 \rangle$. Finally, let us denote by a_i^* the i^{th} agent in A^*.

With these definitions in place, we can now present the coalition structure generation algorithm of Rahwan et al. [68]; we will call this algorithm DC as it uses a *divide-and-conquer* techniques. The basic idea is to create lists, $L_1^*, \cdots, L_{|A^*|+1}^*$, where L_1^* consists of the base cases that contain a_1^*, and each $L_i^* : i \in \{2, \ldots, |A^*|\}$ consists of the base cases that contain a_i^* but none of $a_1^*, \ldots, a_{i-1}^*$, while $L_{|A^*|+1}^*$ consists of the base cases that do not contain any of $a_1^*, \ldots, a_{|A^*|}^*$. This is illustrated in Figure 9 (B). Importantly, by constructing the lists in this way, every feasible coalition structure contains *exactly* one coalition from L_1^*, and *at most* one coalition from $L_i^*, i > 1$. Thus, the algorithm picks a coalition, say C_1, from some base case in L_1^*, and checks whether $\{C_1\}$ is a feasible coalition structure. If not, then the agents in C_1 are added to the negative constraints of all base cases in L_2^*. This places further constraints on the coalitions in those base cases, so as to ensure that they do not overlap with C_1. Next, the algorithm picks a coalition, say C_2, from some base case in (the now modified) list L_2^*, and checks whether $\{C_1, C_2\}$ is a feasible coalition, and so on. Eventually, all feasible coalition structures are examined. To speed up the search, the algorithm applies

ACCEPTED MANUSCRIPT

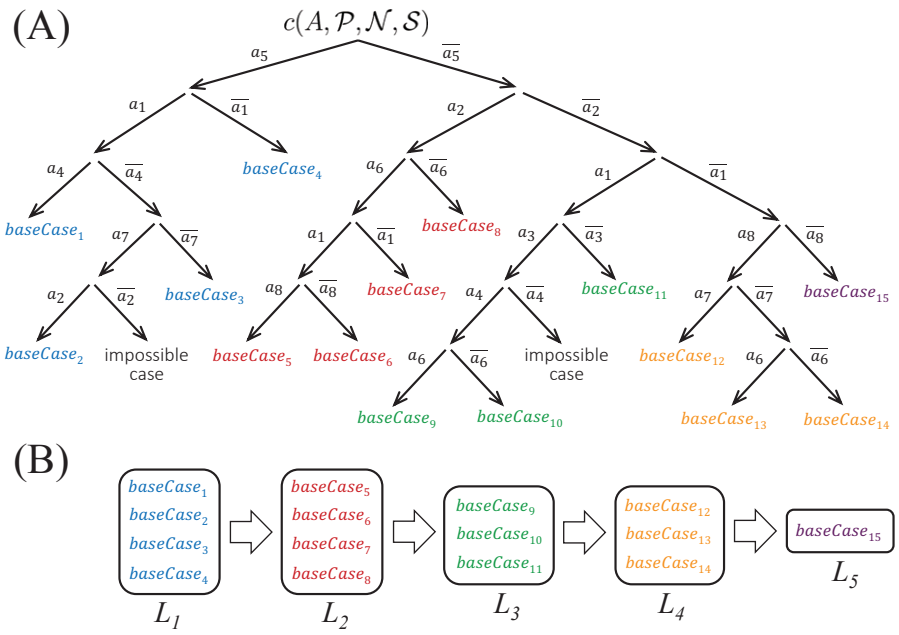


Figure 9: Given a basic CCF, Figure (A) shows how to generate feasible coalitions, while Figure (B) shows how to generate feasible coalition structures.

a *branch-and-bound* technique (see [68] for more details). This algorithm was compared against a modified version of the integer programming formulation that we presented earlier in Section 5.4, where z contains a column for every *feasible* coalition, instead of a column for every *possible* coalition. This comparison showed that DC outperforms the integer programming approach by orders of magnitude (DC can solve random instances consisting of 30 agents and 1000 constraints in a few seconds on a modern desktop computer, but seems unlikely to scale beyond tens of agents).

8.2. Games Restricted by Graphs

Myerson [54] introduced *graph-restricted games*—a class of games where the set of feasible coalitions is specified by an underlying graph $G(A, E)$. More specifically, the nodes of the graph represent the agents, while the edges can be interpreted as communication channels, or trust relationships, which facilitate the cooperation. As such, a coalition is feasible if and only if it induces a connected subgraph of G. Following convention, we assume that G is connected.[19]

Voice et al. [92] proposed a version of the IDP algorithm (from Section 5.3) for graph-restricted games. This algorithm, called DyCE, is very similar to IDP except that $f(C)$ is set to $-\infty$ if C is not feasible. Otherwise, if C is feasible, then out of all possible splits of C that are considered by IDP to compute $f(C)$, DyCE only considers a split $\{C', C \setminus C'\}$ if C' is feasible.[20]

An alternative algorithm was proposed by Bistaffa et al. [12], based on *edge contraction*—a basic operation in graph theory which involves: (i) *removing* an edge from a graph, and (ii) *merging* the two nodes that were previously joined by that edge. In our context of graph-restricted games, since every node represents an agent (i.e., a singleton coalition), *"merging the two nodes"* corresponds to *merging the two coalitions* that

[19] If G is not connected, the coalition structure generation problem can be decomposed into smaller independent subproblems, each with a connected graph.

[20] To do so, one needs the ability to iterate over all such C', i.e., over all connected induced subgraphs of G. This can be done, e.g., using the algorithm by Moerkotte [53].

ACCEPTED MANUSCRIPT

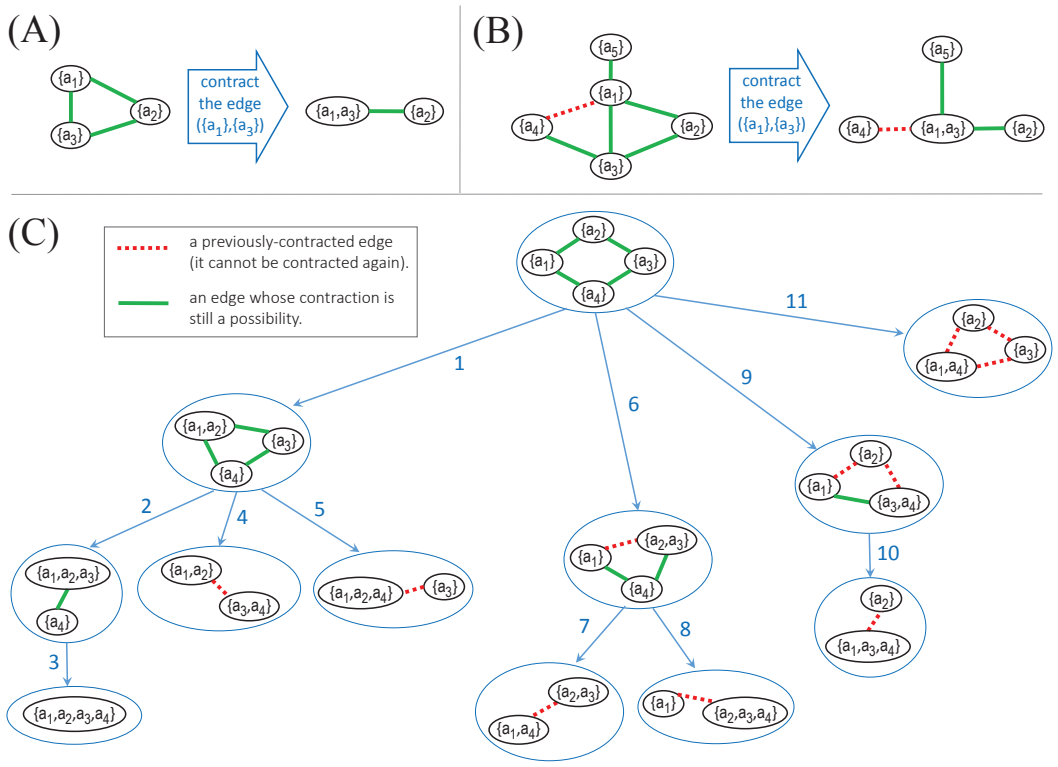


Figure 10: Illustration of how Bistaffa et al.'s algorithm works.

were represented by those nodes. An example is illustrated in Figure 10 (A).

Taking the entire graph into consideration, the contraction of an edge can be interpreted as a transition from one coalition structure to another. For instance, the contraction of the edge $(\{a_1\}, \{a_3\})$ in Figure 10 (B) corresponds to the transition from $\{\{a_1\}, \{a_2\}, \{a_3\}, \{a_4\}, \{a_5\}\}$ to $\{\{a_1, a_3\}, \{a_2\}, \{a_4\}, \{a_5\}\}$. Based on this observation, the algorithm repeats the process of contracting different edges, in order to eventually visit all coalition structures. During this process, to ensure that each coalition structure is visited no more than once, the algorithm marks all previously-contracted edges to avoid contracting them again in the future. In Figure 10, the marked edges are illustrated as dashed lines. Here, it is important to note that the contraction of an edge may result in merging other edges. In Figure 10 (B) for example, contracting $(\{a_1\}, \{a_3\})$ results in merging $(\{a_1\}, \{a_4\})$ with $(\{a_3\}, \{a_4\})$, as well as merging $(\{a_1\}, \{a_2\})$ with $(\{a_3\}, \{a_2\})$. Whenever this happens, if one of the merged edges happens to be dashed, the edge that results from the merger must also be dashed, again see Figure 10 (B). This ensures that the agents appearing at the two ends of a dashed edge never appear together in the same coalition.

Figure 10 (C) illustrates the sequence in which the algorithm visits all possible coalition structures given the graph $G = (A, E)$ where $A = \{a_1, a_2, a_3, a_4\}$ and $E = \{(a_1, a_2), (a_1, a_4), (a_3, a_2), (a_3, a_4)\}$. Each coalition structure is represented as a node in the illustrated search tree, and the numbers on the edges represent the order in which the algorithm visits the different coalition structures.

To speed up the search, a branch-and-bound technique is used whenever the algorithm visits a node CS in the search tree. The purpose of this technique is to determine whether it is worthwhile to search T^{CS}—the sub-tree rooted at CS. The general idea is to compute an upper bound, denoted $UB(T^{CS})$, on the values of all solutions in T^{CS}. Then, if this upper bound was not greater than the value of the best solution found so far, the algorithm skips searching $T(CS)$. Bistaffa et al. proposed a way of computing $UB(T^{CS})$ for cases where the game under consideration happens to be the *sum* of two games: a *weakly superadditive*

ACCEPTED MANUSCRIPT

game, and a *weakly subadditive* game.[21] In this case, we write $(A, v) = (A, v^{sup}) + (A, v^{sub})$, where (A, v^{sup}) denotes the weakly superadditive game, while (A, v^{sub}) denotes the weakly subadditive game. Here, it is possible to compute an upper bound $UB(T^{CS})$ based on the following observations:

- Every coalition structure in T^{CS} is the result of merging some (if not all) of the coalitions in CS that are connected via solid edges. Here, the only constraint is that agents appearing at the two ends of a dashed edge must not appear together in the same coalition.
- Merging coalitions in CS can never improve solution quality in a weakly subadditive game. Thus, $V^{sub}(CS) = \max_{CS' \in T^{CS}} V^{sub}(CS')$.
- Merging coalitions in CS can never reduce solution quality in a weakly superadditive game. Thus, no solution in T^{CS} can be better than the solution obtained by (i) removing all dashed edges, and (ii) merging all coalitions in CS that are connected via solid edges. Let us denote this solution as CS^{merge}. Then, $V^{sup}(CS^{merge}) \geq \max_{CS' \in T^{CS}} V^{sup}(CS')$.

Based on the above observations, we can establish the following upper bound on solution quality: $UB(T^{CS}) = V^{sub}(CS) + V^{sup}(CS^{merge})$. The authors experiment with two specific characteristic functions that satisfy the above conditions (i.e., each being the sum of a weakly superadditive function and a weakly subadditive function). Results show that the algorithm solves problems of 60 nodes in a matter of seconds, but is unlikely to scale beyond tens of agents.

So far in this subsection, we looked at coalition structure generation when feasible coalitions are represented as a graph. Voice et al. [91] worked on a different model, which is also based on a given underlying graph $G = (A, E)$. In their model, *all the coalitions are feasible*, and so the graph G is not used as before (to determine the feasible coalitions). Instead, G is used to place the following constraint on the *marginal contributions*[22] of agents:

$$\forall a_i, a_j \in A : (a_i, a_j) \notin E,\ \forall C \subseteq A \setminus \{a_i, a_j\},\quad mc(a_i, C) = mc(a_i, C \cup \{a_j\}).$$

That is, if two agents a_i and a_j are not connected in G, then the presence of a_i does not affect the marginal contribution of a_j to any coalition C. This property is called *independence of disconnected members (IDM)* [91]. Theorem 13 implies the following: Given a characteristic function game (A, v), if v satisfies IDM, then we can assume (A, v) to be a graph-restricted game without losing the guarantee of finding an optimal coalition structure. This is particularly desirable if G is sparse, as it means we could restrict our attention to a much smaller set of possible coalitions (instead of considering every subset of A to be a feasible coalition).

Theorem 13. *Let (A, v) be a characteristic function game. If v satisfies IDM, then there exists an optimal coalition structure in which every coalition induces a connected subgraph of G.*

Proof. First, observe that for any coalition C, and any permutation of the members of C, denoted as $\pi = (a_1^\pi, \ldots, a_C^\pi)$, the following holds:

$$v(C) \ = \ \sum_{i=1}^{|C|} mc(a_i^\pi, \bigcup_{j=1}^{i-1} \{a_j^\pi\}). \tag{22}$$

[21] Recall that a game (A, v) is said to be the *sum* of two games, (A, v_1) and (A, v_2), if $v(C) = v_1(C) + v_2(C)$ for all $C \subseteq A$. Furthermore, a game (A, v) is *weakly superadditive* if: $v(C \cup C') \geq v(C) + v(C')$ for any two disjoint coalitions C, C'. Conversely, a game (A, v) is *weakly subadditive* if: $v(C \cup C') \leq v(C) + v(C')$ for any two disjoint coalitions C, C'.

[22] Recall that the *marginal contribution* of agent a_i to coalition C, denoted as $mc(a_i, C)$, is the difference in value that a_i makes when joining C. More formally, $mc(a_i, C) = v(C \cup \{a_i\}) - v(C)$.

ACCEPTED MANUSCRIPT

In other words, imagine that the members of C arrived one at a time, in the order specified by π (i.e., a_1^π arrived first, followed by a_2^π, then a_3^π and so on). Then, it is possible to compute $v(C)$ by adding up the marginal contribution that each agent makes when joining the agents who already arrived at C before it.

Now, let CS^* be an optimal coalition structure, and let $C^* \in CS^*$ be a coalition that induces a *disconnected* subgraph of G. This implies that there exists a partition of C, namely $\{C^1, \ldots, C^{k>1}\}$, such that every C^x induces a connected subgraph of G, and every $C^x, C^y : x \neq y$ are disconnected. Thus, due to the IDM property, we have:

$$\forall a_i \in C^x, \quad mc\big(a_i, C\big) = mc\big(a_i, C \setminus (C^1 \cup \ldots \cup C^{x-1} \cup C^{x+1} \cup \cdots \cup C^k)\big). \tag{23}$$

Based on (23) and (22), the following holds, where $\pi^x = (a_1^{\pi^x}, \ldots, a_{|C^*|}^{\pi^x})$ denotes a permutation of the members of C^x:

$$v(C) \;=\; \sum_{x=1}^{k} \sum_{i=1}^{|C^x|} mc(a_i^{\pi^x}, \bigcup_{j=1}^{i-1} \{a_j^{\pi^x}\}) \;=\; \sum_{x=1}^{k} v(C^x).$$

This implies that the value of CS^* is the same as that of $(CS^* \setminus \{C\}) \cup \{C^1\} \cup \ldots \cup \{C^k\}$. In so doing, we have replaced C, which induces a *disconnected* subgraph, by a number of coalitions that each induce a *connected* subgraph. □

Voice et al. [91] analyzed the coalition structure generation problem of a CFG game (A, v) where v satisfies IDM on a graph G. They proved the following:

- The problem is solvable in $O(n)$ steps if the maximum *treewidth* (see Definition 4) of G is bounded by some constant.
- The problem is solvable in $O(n^{\gamma\sqrt{n}+O(1)})$ steps, for $\gamma = \sqrt{2}/(1-\sqrt{2/3})$, if G is a *planar graph* (i.e., it can be drawn on the plane in such a way that no edges cross each other).
- The problem is NP-complete if G is a weighted planar graph, and v evaluates each coalition by summing up the weighted of the edges therein.

The proofs are somewhat lengthy, and so are omitted due to space constraints.

9. Coalition Structure Generation in Partition Function Games

Until now, we assumed that coalitions are not affected by the way non-members are partitioned. However, in many realistic settings, the performance of a coalition may be influenced by the cooperative arrangements of other agents (as discussed in the following subsection). To model such situations, Lucas and Thrall [43] proposed *Partition Function Games* (*PFG*s). In this formalism, the value of a coalition C not only depends on the identities of its members, but also depends on the coalition structure in which C is embedded. More formally, as mentioned earlier in Section 2, a partition function game is a pair, (A, w), where A is the set of agents and w is the *partition function*, which maps each embedded coalition, $(C, CS) \in EC$, to its value, $w(C, CS)$.

This section focuses on the coalition structure generation problem in PFGs. In more detail, Section 9.1 introduces *externalities*—a central notion in PFGs. After that, Section 9.2 shows how to establish a bound on solution quality by only searching part of the solution space.

ACCEPTED MANUSCRIPT

9.1. Introducing Externalities

A central notion in PFGs is that of an **externality**, which is the change in a coalition's value as a result of merging two other co-existing coalitions.[23] For instance, given $CS = \{C_1, C_2, C_3\}$ and $CS' = \{C_1, C_2 \cup C_3\}$, the value of C_1 may be different in CS than in CS'. In this case, we say that the formation of $(C_2 \cup C_3, CS')$—which is the result of merging (C_2, CS') with (C_3, CS)—imposes an *externality* on (C_1, CS'), computed as:

$$\epsilon(C, CS, CS') = w(C, CS) - w(C, CS').$$

We remark that the coalition structure graph (see Figure 1) provides a convenient overview of all externalities in a partition function game. Specifically, since every edge represents a merger of two coalitions, it can be associated with the externalities imposed on other coalitions as a result of this merger.

Interestingly, although characteristic function games are a special case of partition function games where the externalities are always zero, the space of possible solutions to the coalition structure generation problem is the same in both cases. The main difference, however, lies in the size of the input, which is $O(2^n)$ for games with no externalities, but $O(n^n)$ when externalities are present.

Generally speaking, for a game with arbitrary externalities, it is not possible to solve the coalition structure generation problem without examining every single coalition structure. This is because even if all but one have been examined, that remaining one could have a value that is arbitrarily greater than that of all other coalition structures (as a result of the arbitrary externalities therein). However, as we will show in the following subsections, the situation is not as bleak when considering the following two classes of PFGs, which were first introduced in the coalition structure generation literature by Rahwan et al. [66]:

- PFG^-—*games with weakly* ***negative*** *externalities.* Here, the merger of any two coalitions is never beneficial to other co-existing coalitions. Formally, $\epsilon(C, CS, CS') \leq 0$.
- PFG^+—*Games with weakly* ***positive*** *externalities.* Here, merging any two coalitions is never detrimental to other coalitions in the system. Formally, $\epsilon(C, CS, CS') \geq 0$.

In fact, many applications of games with externalities naturally belong to one of the above two classes:

- Examples of a PFG^- setting include collusion in oligopolies, where cooperating companies seek to undermine the competitive position of other firms in the market [94]. Another example is multi-agent systems with shared resources [23, 78]. Here, if the formation of a coalition leads its members to consume more resources, then this leads to fewer resources being available to the other coalitions. This is the case, for instance, in congestion games [55].
- Examples of a PFG^+ setting include the decision by one group of countries to reduce pollution, which has a positive impact on other countries or regions [28, 94]. Another example is multi-agent systems with overlapping or partially overlapping goals. In such systems, while satisfying their own goals, some coalitions may also satisfy (some of) the goals of other coalitions [78].

In the following subsection, we propose some of the theoretical results developed for PFG^+ and PFG^- settings.

[23] For a discussion of alternative notions of externalities in multi-agents systems see [49], and for a comprehensive economic study of externalities and related issues see [18].

ACCEPTED MANUSCRIPT

9.2. Establishing a Bound with Partial Search

Recall how in Section 5.1 we presented algorithms that (i) divide the space into subspaces, and (ii) specify a sequence in which these subspaces must be searched, so that the worst-case guarantee on solution quality improves after each subspace. All those algorithms were designed for characteristic function games (CFGs). In this section, we present two algorithms that follow the same design paradigm, but can be applied in certain classes of PFGs, where the value of a coalition may differ from one coalition structure to another.

The first such algorithm was proposed by Rahwan et al. [67] for PFG^+ and PFG^- settings. This algorithm is based on a number of theorems that we will present next. Here, for any $C \subseteq A$, the coalition consisting of the agents in $A \setminus C$ will be denoted as $\overline{C} = \{\overline{a}_1, ..., \overline{a}_k\}$.

We start with the following theorem, which is similar to Theorem 5 except that it deals with PFG settings. Recall that in Theorem 5 we use $\delta(\Pi')$ which is defined as:

$$\delta(\Pi') = \bigcup_{CS' \in \Pi'} \{P \subseteq CS'\}.$$

In the following theorem, instead of $\delta(\Pi')$, we use $\delta^{\max}(\Pi')$ which is defined as:

$$\delta^{\max}(\Pi') = \bigcup_{CS' \in \Pi'} \{P \subseteq CS' : W(P, CS') = \max_{CS \in \Pi^A} W(P, CS)\}.$$

In other words, just like $\delta(\Pi')$, the set $\delta^{\max}(\Pi')$ consists of every subset of every $CS' \in \Pi$; the only difference is that every such subset does not belong to $\delta^{\max}(\Pi')$ unless it appears *with its maximum value* in CS', i.e., its value in CS' is greater than, or equal to, its value in any other coalition structure.

Theorem 14. *Given a PFG setting and a subspace, $\Pi' \subseteq \Pi^A$, a bound can be established on $\frac{\max_{CS \in \Pi^A} W(CS)}{\max_{CS \in \Pi'} W(CS)}$ if and only if:*

$$\forall C \subseteq A,\ \exists CS' \in \Pi' : W(C, CS') = \max_{CS \in \Pi^A} W(C, CS). \tag{24}$$

Furthermore, if the above condition holds, then we can establish the following bound:

$$\frac{\max_{CS \in \Pi^A} W(CS)}{\max_{CS \in \Pi'} W(CS)} \leq \max_{CS \in \Pi^A} \left(\min_{P \in \Pi^{CS} : P \subseteq \delta^{\max}(\Pi')} |P| \right). \tag{25}$$

Sketch of Proof. If condition (24) is not satisfied, then there exists a coalition whose value in some coalition structure Π^A is greater than its value in any of the coalition structures in Π'. Since the difference in value could be arbitrarily large, it is not possible to establish a bound on $\frac{\max_{CS \in \Pi^A} W(CS)}{\max_{CS \in \Pi'} W(CS)}$.

Conversely, if condition (24) is satisfied, then Theorem 14 states that (25) holds. The intuition behind this is very similar to the intuition behind the proof of Theorem 5. Roughly speaking, if we can identify a set consisting of groups of disjoint coalitions, such that:

- the best solution in Π^A, i.e., $\arg\max_{CS \in \Pi^A} W(CS)$, **contains at most x groups**,
- and every one of those groups appears *with its maximum value* in some $CS \in \Pi'$,
- then $\arg\max_{CS \in \Pi^A} W(CS)$ **is at most x times better** than $\arg\max_{CS \in \Pi'} W(CS)$.

Equation (25) captures exactly the above intuition. The formal proof can be found in [69]; it follows the same reasoning as in the proof of Theorem 5. □

The above theorem is general for any PFG, and states that a bound can only be established by examining coalition structures in which coalitions (or certain groups of coalitions) appear *with their maximum value*. The following theorem identifies such coalition structures given a PFG^- or PFG^+ setting.

ACCEPTED MANUSCRIPT

Theorem 15. *Consider a PFG^- (PFG^+) setting. For every coalition $C \subseteq A$, and every partition of C, i.e., $P \in \Pi^C$, and every $CS : P \subseteq CS$, the following holds:*

$$W(P, \{\overline{C}\} \cup P) \ \leq \ (\geq) \ W(P, CS) \ \leq \ (\geq) \ W(P, \{\{\overline{a}_1\}, \ldots, \{\overline{a}_k\}\} \cup P).$$

Proof. To simplify notation, let $CS' = \{\overline{C}\} \cup P$ and $CS'' = \{\{\overline{a}_1\}, ..., \{\overline{a}_k\}\} \cup P$. Also, without loss of generality, assume that $CS \neq CS'$ and $CS \neq CS''$. Then, given a PFG^- (PFG^+) setting, we first need to prove that:

$$W(P, CS') \quad \leq (\geq) \quad W(P, CS). \tag{26}$$

Starting from CS, it is always possible to reach CS' by performing multiple steps, each involving the merging of two coalitions into one. In each step, the coalitions in P remain unchanged, and due to negative (positive) externalities, their values can only decrease (increase). As a result, the inequality in (26) holds. Similarly, starting from CS'', it can be shown that:

$$W(P, CS) \leq (\geq) \ W(P, CS'').$$

□

Theorem 16. *To establish a bound β on the value of a coalition structure given a PFG^+ setting, every subspace $\Pi^A_I : I \in \mathcal{I}^n : |I| \leq 2$ must be searched. With this search, the number of coalition structures searched is 2^{n-1}, and the bound is $\beta = n$. On the other hand, given a PFG^- setting, every subspace $\Pi^A_I : I \in \{\{n\}, \{n-1, 1\}, \{n-2, 1, 1\}, \cdots, \{1, 1, \cdots, 1\}\}$ must be searched. With this search, the number of coalition structures searched is $2^n - n + 1$, and the bound is $\beta = \lceil \frac{n}{2} \rceil$.*

Proof. To establish a bound, the *maximum possible value* of each coalition C has to be observed (in some coalition structure). Given a PFG^+ setting, the only coalition structure in which C is guaranteed to have *its maximum value* is $\{C, A \backslash C\}$ (see Theorem 15). Based on this, the subspaces $\Pi^A_I : I \in \mathcal{I}^n : |I| \leq 2$ must be searched, and these collectively contain 2^{n-1} coalition structures.

To prove that $\beta = n$, we use Theorem 14. Here, the set of coalition structures that have been searched is: $\Pi' = \bigcup_{I \in \mathcal{I}^n : |I| \leq 2} \Pi^A_I$. Furthermore, $\delta^{\max}(\Pi') = 2^A$ (because every coalition has been observed in Π' with its maximum value). Thus, based on (25):

$$\beta = \max_{CS \in \Pi^A} \left(\min_{P \in \Pi^{CS} : P \subseteq \delta^{\max}(\Pi')} |P| \right) = \max_{CS \in \Pi^A} |CS| = n.$$

Moving on to the PFG^- case, the only coalition structure in which C is guaranteed to have its maximum value is: $\{C, \{\overline{a}_1\}, \ldots, \{\overline{a}_k\}\}$ (see Theorem 15). Based on this, the following subspaces have to be searched: $\Pi^A_I : I \in \{\{n\}, \{n-1, 1\}, \{n-2, 1, 1\}, \ldots, \{1, 1, \ldots, 1\}\}$. With this search, the number of searched coalition structures is $2^n - n + 1$, because every possible coalition appears in a unique coalition structure, except for singletons which all appear in a single coalition structure.

Finally, to prove that $\beta = \lceil \frac{n}{2} \rceil$, we use Theorem 14. Here, the set of coalition structures that have been searched is $\Pi' = \bigcup_{I \in \{\{n\}, \{n-1,1\}, \{n-2,1,1\}, \ldots, \{1,1,\ldots,1\}\}} \Pi^A_I$. Moreover $\delta^{\max}(\Pi') = 2^A \cup \{P \subseteq 2^A : \forall C \in P, |C| = 1\}$ (because every coalition, and every combination of singletons, has been observed in Π' with its maximum value). Thus, based on (25), we have:

$$\beta = \max_{CS \in \Pi^A} \left(\min_{P \in \Pi^{CS} : P \subseteq \delta^{\max}(\Pi')} |P| \right) = \left\lceil \frac{n}{2} \right\rceil.$$

This is because, out of all $CS \in \Pi^A$, the ones that need the maximum number of groups from $\delta^{\max}(\Pi')$ to partition them are those in $\mathcal{P}^A_{\{2,2,\cdots,2,1\}}$ when the number of agents is odd, and those in $\mathcal{P}^A_{\{2,2,\cdots,2\}}$ when

ACCEPTED MANUSCRIPT

the number of agents is even. In either case, the number of groups from $\delta^{\max}(\Pi')$ that partition any such coalition structure is: $\lceil \frac{n}{2} \rceil$. □

Interestingly, in CFGs, it is **sufficient** to search the first and second levels of the coalition structure graph in order to establish bound β (see Theorem 4).[24] On the other hand, given a PFG^- setting, it is **necessary** to search $\Pi^A_I : I \in \{\{n\}, \{n-1,1\}, \{n-2,1,1\}, \cdots, \{1,1,\cdots,1\}\}$ and, given a PFG^+ setting, it is **necessary** to search $\Pi^A_I : I \in \mathcal{I}^n : |I| \leq 2$ (see Theorem 16).

Rahwan et al. [67] developed an algorithm that can improve the above bounds with further search. They also developed a version of the IP algorithm for PFG^+ and PFG^- settings, called $IP^{+/-}$, which builds upon a general framework introduced by Michalak et al. [45]. A decentralized version of $IP^{+/-}$ was recently developed by Epstein and Bazzan [25].

Some of the above results were extended by Banerjee and Kraemer [8] to settings where agents are grouped into *"types"* (see Section 7.3 for more details on agent types). Specifically, the authors assume that if two coalitions C_1 and C_2 merge, then the externality imposed by this merge on a third coalition C_3 is weakly positive if the types of the agents in $C_1 \cup C_2$ do not overlap with those of the agents in C_3. Otherwise, the externality is weakly negative. Banerjee and Kraemer [8] argue that this class is intuitive, and maps to a number of applications.

Finally, in terms of concise representations of PFGs, we note that Ichimura et al. [33] extended the mixed integer programming formulation of Ohta et al. [56] (from Section 7.5) to solve the coalition structure generation problem under the *Embedded MC-nets* representation of Michalak et al. [46], which is an extension of standard MC-nets that captures externalities. A follow-up result was obtained by Liao et al. [41] and Liao et al. [42], who showed that, similarly to MC-Nets (see Section 7.5), it is also possible to encode Embedded MC-Nets into propositional Boolean logic and solve the coalition structure generation problem with an off-the-shelve MaxSAT solver.

10. Discussion

In this section, we discuss the scopes of the different representations and how they stand in relation to each other. Figure 11 provides an illustration, and outlines the relevant references discussed in this survey.

We start with characteristic function games, or CFGs (sections 4, 5 and 6), which represent settings in which every coalition is admissible, and every coalition value is represented by a real number that depends solely on the identities of the members. We studied two well-known representations for CFGs:

- the first is under the *DCOP* representation (Section 7.1), where the value of a coalition equals the solution to a Distributed Constraint Optimization Problem (DCOP) involving the coalition members;
- the second is under the *induced-subgraph* representation (end of Section 7.5), where the characteristic function is based on a weighted graph in which the nodes correspond to the agents in the system. Here, the value of a coalition is the sum of the weights of the edges whose ends belong to the coalition (if no such edges exists, the value is 0).

We discussed in this article four alternative representations to CFGs. These are:

- the *skill-based* representations (Section 7.2), where every agent has a set of skills, and the value of a coalition depends on the skills of its members. For any CFG, one can define a set of skills that represent that game, see Section 7.2;

[24] It is also possible to establish a bound by searching any other set of coalition structures as long as every coalition appears at least once in that set.

ACCEPTED MANUSCRIPT

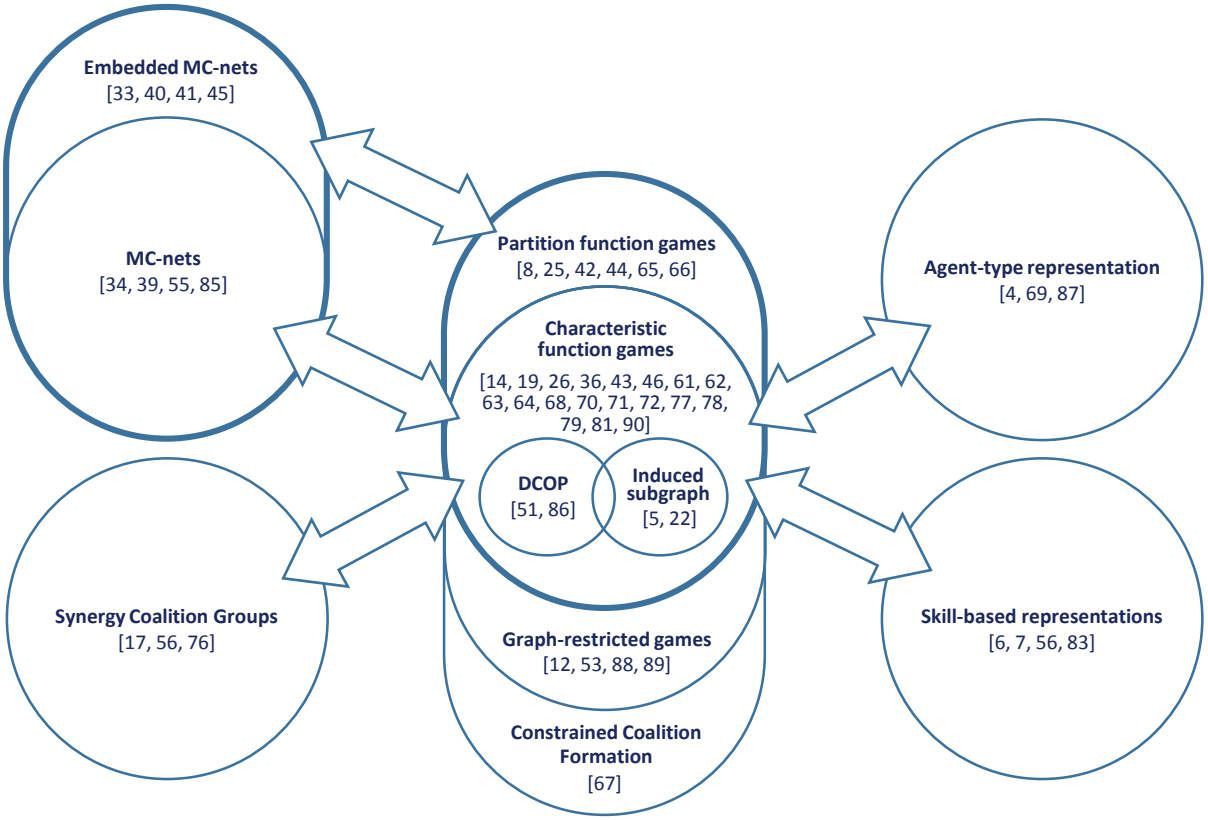


Figure 11: A map of the different representations considered in this article.

- the *agent-type* representation (Section 7.3), whereby agents are classified into *types* based on their contributions: agents of the same type make the same contribution to any coalition they belong to. This clearly can represent any CFG (if none of the agents were identical, the agent-type representation will simply define a unique type for every agent).
- the *Synergy Coalition Groups* representation, where only the coalitions with synergy are explicitly modeled. As mentioned earlier in Section 7.4, this representation is *fully expressive* under certain conditions, i.e., it can represent any CFG;
- the *Marginal Contribution net (MC-net)* representation, which is fully expressive, see Section 7.5.

Importantly, in all of the above representations, every coalition is permitted. There are cases, however, where certain coalitions are inadmissible; these cannot be represented by CFGs. Instead:

- One can use *graph-restricted games* (Section 8.2), where agents are connected via a graph, and a coalition is feasible if it induces a connected subgraph. Observe that CFGs form a proper subclass of graph-restricted games, where the graph happens to be complete; in this case every coalition is admissible.
- One can also use the *constrained coalition formation* (or CCF) model (Section 8.1), where constraints are expressed in terms of acceptable coalition sizes, and in terms of subsets of agents whose presence in a coalition is desirable (each such subset is a *positive* constraint in $\mathcal{P}$), and other subsets whose presence is prohibited (each such subset is a *negative* constraint in $\mathcal{N}$). Note that graph-restricted games form a proper subclass of CCF games. This is based on two observations:

ACCEPTED MANUSCRIPT

1. Given a graph-restricted game with an underlying graph, G, a corresponding CCF game can be constructed as follows: for every $C \subseteq A$ that induces a connected subgraph of G, add C to $\mathcal{P}$, and add $C \cup \{a_i\}$ to $\mathcal{N}$ for every $a_i \in A \setminus C$ such that $C \cup \{a_i\}$ does not induce a connected subgraph of G.
2. There exists CCF games that cannot be represented by a graph-restricted game. These CCF games include, for instance, those in which $\{\{a_1, a_2\}, \{a_2, a_3\}\} \subseteq \mathcal{P}$ and $\{a_1, a_2, a_3\} \in \mathcal{N}$.

Finally, let us consider *partition function games* (or PFGs), where the same coalition can have different values depending on the way non-members are partitioned. Clearly, CFGs form a proper subclass of PFGs where a coalition has the same value for every partition of non-members. An extension of MC-nets, called *embedded MC-nets*, was proposed to represent PFGs.

11. Conclusions

The coalition structure generation problem is a simple abstraction of a basic challenge in team formation: How can agents divide themselves into groups to optimise overall performance? This article has provided a comprehensive synthesis of the current state of the research into this problem. We did this by: (i) presenting various representations of the search space, (ii) discussing a number of dynamic programming techniques to solve this problem, (iii) describing the main theoretical results on how to establish worst-case guarantees with partial search, (iv) presenting a number of anytime exact algorithms, (v) outlining the main findings on how to solve large problem instances with metaheuristics, (vi) describing how to approach the problem under various concise representations, e.g., MC-nets, Skill Games, agent-types etc., (vii) reviewing the main methods that handle constraints, whether represented using propositional logic or in the form of a graph, (viii) describing the main findings on how to handle externalities—influences between co-existing coalitions. All of these results were presented in a self-contained manner, with an emphasis on exposing the underlying intuition whenever possible.

We envisage two broad future directions in this growing area of research. On one hand, there is a need to consider richer and more sophisticated models to capture the various factors that influence the coalition formation process in the real world. Examples include *overlapping* coalition formation [15], and *generalized* characteristic functions, where the order of agents in a coalition matters [50]. A potential downside here is that richer models may render the coalition structure generation problem harder to solve (as we saw in the case of partition function games). On the other hand, there is a need for further research on tractability and scalability. Here, perhaps one of the most promising directions is that of concise representations. These may not necessarily be fully expressive, but instead focus on capturing specific properties in a way that facilitates an efficient computation of the coalition structure generation problem.

Acknowledgements

Tomasz Michalak and Michael Wooldridge were supported by the European Research Council under Advanced Grant 291528 ("RACE"). Nicholas R. Jennings was supported by the ORCHID Project, funded by EPSRC (Engineering and Physical Research Council) under the grant EP/I011587/1.

References

[1] Aadithya, K. V., Michalak, T. P., and Jennings, N. R. 2011. Representation of coalitional games with algebraic decision diagrams. In *Proceedings of 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1121–1122.

[2] Andrews, G. E. and Eriksson, K. 2004. *Integer Partitions*. Cambridge University Press, Cambridge, UK.

[3] Augustine, J., Chen, N., Elkind, E., Fanelli, A., Gravin, N., and Shiryaev, D. 2011. Dynamics of profit-sharing games. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. IJCAI'11. AAAI Press, 37–42.

ACCEPTED MANUSCRIPT

[4] Aziz, H. and de Keijzer, B. 2011. Complexity of coalition structure generation. In *AAMAS '11: Tenth International Conference on Autonomous Agents and Multi-Agent Systems*. 191–198.

[5] Bachrach, Y., Kohli, P., Kolmogorov, V., and Zadimoghaddam, M. 2013. Optimal coalition structure generation in cooperative graph games. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI)*, M. desJardins and M. L. Littman, Eds. AAAI Press.

[6] Bachrach, Y., Meir, R., Jung, K., and Kohli, P. 2010. Coalitional structure generation in skill games. In *Twenty Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 703–708.

[7] Bachrach, Y. and Rosenschein, J. S. 2008. Coalitional skill games. In *AAMAS'08: Seventh International Conference on Autonomous Agents and Multi-Agent Systems*. 1023–1030.

[8] Banerjee, B. and Kraemer, L. 2010. Coalition structure generation in multi-agent systems with mixed externalities. In *AAMAS '10: Ninth International Conference on Autonomous Agents and Multi-Agent Systems*. 175–182.

[9] Bell, E. T. 1934. Exponential numbers. *American Mathematical Monthly 41*, 411–419.

[10] Berghammer, R. and Bolus, S. 2012. On the use of binary decision diagrams for solving problems on simple games. *European Journal of Operational Research 222,* 3, 529–541.

[11] Berre, D. L. and Parrain, A. 2010. The sat4j library, release 2.2. *JSAT 7,* 2-3, 59–6.

[12] Bistaffa, F., Farinelli, A., Cerquides, J., Rodríguez-Aguilar, J., and 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 '14. 13–20.

[13] Bitar, E., Baeyens, E., Khargonekar, P., Poolla, K., and Varaiya, P. 2012. Optimal sharing of quantity risk for a coalition of wind power producers facing nodal prices. In *Proceedings 31st IEEE American Control Conference*.

[14] Björklund, A., Husfeldt, T., and Koivisto, M. 2009. Set partitioning via inclusion-exclusion. *SIAM Journal on Computing 39,* 2, 546–563.

[15] Chalkiadakis, G., Elkind, E., Markakis, E., Polukarov, M., and Jennings, N. R. 2010. Cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research 39,* 1, 179–216.

[16] Chalkiadakis, G., Elkind, E., and Wooldridge, M. 2011. *Computational Aspects of Cooperative Game Theory*. Morgan-Claypool.

[17] Conitzer, V. and Sandholm, T. 2006. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence 170*, 607–619.

[18] Cornes, R. and Sandler, T. 1996. *The Theory of Externalities, Public Goods, and Club Goods*. Cambridge University Press.

[19] Dang, V. D. and Jennings, N. R. 2004. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. 564–571.

[20] de Bruijn, N. G. 1981. *Asymptotic Methods in Analysis*. Dover.

[21] Dechter, R. and Pearl, J. 1989. Tree clustering for constraint networks. *Artificial Intelligence 38,* 3, 353–366.

[22] Deng, X. and Papadimitriou, C. 1994. On the complexity of cooperative solution concepts. *Mathematics of Operations Research 19,* 2, 257–266.

[23] Dunne, P. 2005. Multiagent resource allocation in the presence of externalities. In *CEEMAS*. 408–417.

[24] Edgeworth, F. Y. 1881. *Mathematical Psychics: An essay on the mathematics to the moral sciences*. Kegan Paul.

[25] Epstein, D. and Bazzan, A. L. C. 2013. Distributed coalition structure generation with positive and negative externalities. In *EPIA*. Lecture Notes in Computer Science Series, vol. 8154. Springer, 408–419.

[26] Farinelli, A., Bicego, M., Ramchurn, S., and Zucchelli, M. 2013. C-link: A hierarchical clustering approach to large-scale near-optimal coalition formation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*. 106–112.

[27] Feo, T. A. and Resende, M. G. 1995. Greedy randomized adaptive search procedures. *Journal of Global Optimization 6*, 109–133.

[28] Finus, M. 2003. New developments in coalition theory: An application to the case of global pollution control. In *Environmental Policy in an International Perspective*. Kluwer Academic Publishers, Netherlands, 19–49.

[29] Gillies, D. B. 1959. Solutions to general non-zero-sum games. In *Contributions to the Theory of Games, volume IV*, H. Kuhn, A. W. Tucker, and L. D. Luce, Eds. Princeton University Press, 47–85.

[30] Gottlob, G., Leone, N., and Scarcello, F. 2002. Hypertree decompositions and tractable queries. *J. Comput. Syst. Sci. 64,* 3, 579–627.

[31] Han, Z. and Poor, H. V. 2009. Coalition games with cooperative transmission: a cure for the curse of boundary nodes in selfish packet-forwarding wireless networks. *IEEE Transactions on Communications 57,* 1, 203–213.

ACCEPTED MANUSCRIPT

[32] Horling, B. and Lesser, V. 2005. A survey of multi-agent organizational paradigms. *The Knowledge Engineering Review 19,* 4, 281–316.

[33] Ichimura, R., Hasegawa, T., Ueda, S., Iwasaki, A., and Yokoo, M. 2011. Extension of mc-net-based coalition structure generation: Handling negative rules and externalities. In *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*. AAMAS '11. Richland, SC, 1173–1174.

[34] Ieong, S. and Shoham, Y. 2005. Marginal Contribution Nets: a Compact Representation Scheme for Coalitional Games. In *ACM EC '05: 6th ACM Conference on Electronic Commerce*. 193–202.

[35] Immorlica, N., Markakis, E., and Piliouras, G. 2010. Coalition formation and price of anarchy in cournot oligopolies. In *WINE*, A. Saberi, Ed. Lecture Notes in Computer Science Series, vol. 6484. 270–281.

[36] Keinänen, H. 2009. Simulated annealing for multi-agent coalition formation. In *Proceedings of the Third KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*. KES-AMSTA '09. Springer, Berlin, Heidelberg, 30–39.

[37] Ketchpel, S. P. 1993. Coalition formation among autonomous agents. In *Proceedings of the fifth European Workshop on Modelling Autonomous Agents (MAAMAW)*. 73–88.

[38] Khan, Z., Lehtomaki, J., and DaSilva, M. L.-A. L. A. 2010. On selfish and altruistic coalition formation in cognitive radio networks. In *Fifth International Conference on Cognitive Radio Oriented Wireless Networks and Communications (CROWNCOM-10)*. Cannes, France.

[39] Li, C., Sycara, K., and Scheller-Wolf, A. 2010. Combinatorial coalition formation for multi-item group-buying with heterogeneous customers. *Decis. Support Syst. 49,* 1, 1–13.

[40] Liao, X., Koshimura, M., Fujita, H., and Hasegawa, R. 2012. Solving the coalition structure generation problem with maxsat. In *Tools with Artificial Intelligence (ICTAI), 2012 IEEE 24th International Conference*. Vol. 1. 910–915.

[41] Liao, X., Koshimura, M., Fujita, H., and Hasegawa, R. 2014a. Extending maxsat to solve the coalition structure generation problem with externalities based on agent relations. *IEICE TRANSACTIONS on Information and Systems E97-D,* 7, 1812–1821.

[42] Liao, X., Koshimura, M., Fujita, H., and Hasegawa, R. 2014b. Maxsat encoding for mc-net-based coalition structure generation problem with externalities. *IEICE TRANSACTIONS on Information and Systems E97-D,* 7, 1781–1789.

[43] Lucas, W. and Thrall, R. 1963. *n*-person games in partition function form. *Naval Res. Logist. Quart. X*, 281– 298.

[44] Mauro, N. D., Basile, T. M. A., Ferilli, S., and Esposito, F. 2010. Coalition structure generation with grasp. In *Proceedings of the 14th international conference on Artificial intelligence: methodology, systems, and applications*. AIMSA'10. Springer, Berlin, Heidelberg, 111–120.

[45] Michalak, T., Dowell, A., McBurney, P., and Wooldridge, M. 2008. Optimal coalition structure generation in partition function games. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI)*. 388–392.

[46] Michalak, T., Marciniak, D., Samotulski, M., Rahwan, T., McBurney, P., Wooldridge, M., and Jennings, N. R. 2010. A Logic-based Representation for Coalitional Games with Externalities. In *AAMAS '10: Proceedings of the Ninth International Conference on Autonomous Agents and Multi-Agent Systems*. 125–132.

[47] Michalak, T., Sroka, J., Rahwan, T., Wooldridge, M., McBurney, P., and Jennings, N. R. 2010. A Distributed Algorithm for Anytime Coalition Structure Generation. In *AAMAS '10: Proceedings of the Ninth International Conference on Autonomous Agents and Multi-Agent Systems*. 1007–1014.

[48] Michalak, T. P., Aadithya, K. V., Szczepanski, P. L., Ravindran, B., and Jennings, N. R. 2013. Efficient computation of the shapley value for game-theoretic network centrality. *J. Artif. Intell. Res. (JAIR) 46*, 607–650.

[49] Michalak, T. P., Rahwan, T., Sroka, J., Dowell, A., Wooldridge, M. J., McBurney, P. J., and Jennings, N. R. 2009. On representing coalitional games with externalities. In *ACM Conference on Electronic Commerce (ACM-EC)*, J. Chuang, L. Fortnow, and P. Pu, Eds. 11–20.

[50] Michalak, T. P., Szczepański, P. L., Rahwan, T., Chrobak, A., Brânzei, S., Wooldridge, M., and Jennings, N. R. 2014. Implementation and computation of a value for generalized characteristic function games. *ACM Trans. Econ. Comput. 2,* 4, 16:1–16:35.

[51] Minato, S. 1993. Zero-suppressed bdds for set manipulation in combinatorial problems. In *Proceedings of the Thirtyth Design Automation Conference (DAC-93)*. 272–277.

[52] Modi, P. J. 2003. Distributed constraint optimization for multiagent systems. Ph.D. thesis, Los Angeles, CA, USA.

[53] Moerkotte, G. 2006. Analysis of two existing and one new dynamic programming algorithm for the generation of optimal bushy join trees without cross products. In *In Proc. 32nd International Conference on Very Large Data Bases*. 930–941.

[54] Myerson, R. 1977. Graphs and cooperation in games. *Mathematics of Operations Research 2,* 3, 225–229.

[55] Oh, J. 2009. *Multiagent Social Learning in Large Repeated Games (Ph.D. thesis)*. School of Computer Science, Carnegie Mellon University.

[56] Ohta, N., Conitzer, V., Ichimura, R., Sakurai, Y., Iwasaki, A., and Yokoo, M. 2009. Coalition structure generation utilizing compact characteristic function representations. In *CP'09: 15th International Conference on Principles and Practice of Constraint Programming*. 623–638.

ACCEPTED MANUSCRIPT

[57] Ohta, N., Iwasaki, A., Yokoo, M., Maruono, K., Conitzer, V., and Sandholm, T. 2006. A compact representation scheme for coalitional games in open anonymous environments. In *AAAI'06: 21st National Conference on Artificial Intelligence*. 697–702.

[58] Peleg, B. and Sudholter, P. 2002. *Introduction to the Theory of Cooperative Games (second edition)*. Springer-Verlag: Berlin, Germany.

[59] Petcu, A. and Faltings, B. 2005. A scalable method for multiagent constraint optimization. In *Proceedings of the nineteenth international joint conference on Artificial intelligence (IJCAI)*. San Francisco, CA, USA, 266–271.

[60] Peters, H. 2003. Ntu games. In *Optimization and Operations Research*, U. Derigs, Ed. Encyclopedia of Life Support Systems (EOLSS). Eolss, Oxford, http://www.eolss.net.

[61] Rahwan, T. and Jennings, N. R. 2007. An algorithm for distributing coalitional value calculations among cooperative agents. *Artificial Intelligence 171,* 8–9, 535–567.

[62] Rahwan, T. and Jennings, N. R. 2008a. Coalition structure generation: Dynamic programming meets anytime optimisation. In *AAAI'08: Twenty Third AAAI Conference on Artificial Intelligence*. 156–161.

[63] Rahwan, T. and Jennings, N. R. 2008b. An improved dynamic programming algorithm for coalition structure generation. In *AAMAS'08: Seventh International Conference on Autonomous Agents and Multi-Agent Systems*. 1417–1420.

[64] Rahwan, T., Michalak, T., Elkind, E., Wooldridge, M., and Jennings, N. R. 2014. An exact algorithm for coalition structure generation and complete set partitioning. *http://www.cs.ox.ac.uk/publications/publication6962-abstract.html*.

[65] Rahwan, T., Michalak, T., and Jennings, N. R. 2012. A hybrid algorithm for coalition structure generation. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*.

[66] Rahwan, T., Michalak, T., Jennings, N. R., Wooldridge, M., and McBurney, P. 2009. Coalition structure generation in multi-agent systems with positive and negative externalities. In *Proceedings of the Twenty First International Conference on Artificial Intelligence (IJCAI)*. Pasadena, USA.

[67] Rahwan, T., Michalak, T., Wooldridge, M., and Jennings, N. R. 2012. Anytime coalition structure generation in multi-agent systems with positive or negative externalities. *Artificial Intelligence 186,* 0, 95 – 122.

[68] Rahwan, T., Michalak, T. P., Elkind, E., Faliszewski, P., Sroka, J., Wooldridge, M., and Jennings, N. R. 2011. Constrained coalition formation. In *Twenty Fifth AAAI Conference on Artificial Intelligence (AAAI)*. 719–725.

[69] Rahwan, T., Michalak, T. P., and Jennings, N. R. 2011. Minimum search to establish worst-case guarantees in coalition structure generation. In *IJCAI'11: Twenty Second International Joint Conference on Artificial Intelligence*. 338–343.

[70] Rahwan, T., Nguyen, T.-D., Michalak, T., Polukarov, M., Croitoru, M., and Jennings, N. R. 2013. Coalitional games via network flows. In *Proc. 23rd International Joint Conference on AI (IJCAI)*. 324–331.

[71] Rahwan, T., Ramchurn, S. D., Dang, V. D., and Jennings, N. R. 2007. Near-optimal anytime coalition structure generation. In *IJCAI'07: Twentieth International Joint Conference on Artificial Intelligence*. 2365–2371.

[72] Rahwan, T., Ramchurn, S. D., Giovannucci, A., Dang, V. D., and Jennings, N. R. 2007. Anytime optimal coalition structure generation. In *AAAI'07: Twenty Second Conference on Artificial Intelligence*. 1184–1190.

[73] Rahwan, T., Ramchurn, S. D., Giovannucci, A., and Jennings, N. R. 2009. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research (JAIR) 34*, 521–567.

[74] Robinson, N., Gretton, C., Pham, D., and Sattar, A. 2010. Partial weighted maxsat for optimal planning. In *PRICAI 2010: Trends in Artificial Intelligence*, B.-T. Zhang and M. A. Orgun, Eds. Lecture Notes in Computer Science Series, vol. 6230. Springer Berlin Heidelberg, 231–243.

[75] Russell, S. J. and Norvig, P. 2003. *Artificial Intelligence: a Modern Approach* 2nd edition Ed. Prentice Hall, Upper Saddle River, N.J.

[76] Sakurai, Y., Ueda, S., Iwasaki, A., Minato, S.-I., and Yokoo, M. 2011. A compact representation scheme of coalitional games based on multi-terminal zero-suppressed binary decision diagrams. In *Agents in Principle, Agents in Practice*, D. Kinny, J.-j. Hsu, G. Governatori, and A. Ghose, Eds. Lecture Notes in Computer Science Series, vol. 7047. Springer Berlin Heidelberg, 4–18.

[77] Sandholm, T. 2002. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence 135*, 1–54.

[78] Sandholm, T., Larson, K., Andersson, M., Shehory, O., and Tohmé, F. 1999. Coalition structure generation with worst case guarantees. *Artificial Intelligence 111,* 1–2, 209–238.

[79] Sen, S. and Dutta, P. 2000. Searching for optimal coalition structures. In *ICMAS'00: Sixth International Conference on Multi-Agent Systems*. 286–292.

[80] Service, T. C. and Adams, J. A. 2011. Constant factor approximation algorithms for coalition structure generation. *Autonomous Agents and Multi-Agent Systems 23,* 1, 1–17.

[81] Shehory, O. and Kraus, S. 1993. Coalition formation among autonomous agents: Strategies and complexity. In *Proceedings of the fifth European Workshop on Modelling Autonomous Agents (MAAMAW)*. 56–72.

[82] Shehory, O. and Kraus, S. 1995. Task allocation via coalition formation among autonomous agents. In *In Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI)*. 655–661.

ACCEPTED MANUSCRIPT

[83] Shehory, O. and Kraus, S. 1996. Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. In *Proceedings of International Conference on Multi-Agent Systems (ICMAS-96)*. 330–337.

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

[85] Shoham, Y. and Leyton-Brown, K. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press: Cambridge, England.

[86] Tran-Thanh, L., Nguyen, T.-D., Rahwan, T., Rogers, A., and Jennings, N. R. 2013. An efficient vector-based representation for coalitional games. In *Proceedings of the Twenty First International Conference on Artificial Intelligence (IJCAI)*.

[87] Tuy, H., Ghannadan, S., Migdalas, A., and Värbrand, P. 1996. A strongly polynomial algorithm for a concave production-transportation problem with a fixed number of nonlinear variables. *Math. Program. 72*, 229–258.

[88] Ueda, S., Hasegawa, T., Hashimoto, N., Ohta, N., Iwasaki, A., and Yokoo, M. 2012. Handling negative value rules in mc-net-based coalition structure generation. In *AAMAS '12: Eleventh International Conference on Autonomous Agents and Multi-Agent Systems*, W. van der Hoek, L. Padgham, V. Conitzer, and M. Winikoff, Eds. IFAAMAS, 795–804.

[89] Ueda, S., Iwasaki, A., Yokoo, M., Silaghi, M. C., Hirayama, K., and Matsui, T. 2010. Coalition structure generation based on distributed constraint optimization. In *Twenty Fourth AAAI Conference on Artificial Intelligence (AAAI)*. 197–203.

[90] Ueda, S., Kitaki, M., Iwasaki, A., and Yokoo, M. 2011. Concise characteristic function representations in coalitional games based on agent types. In *IJCAI'11: Twenty Second International Joint Conference on Artificial Intelligence*. 393–399.

[91] Voice, T., Polukarov, M., and Jennings, N. R. 2012. Coalition structure generation over graphs. *Journal of Artificial Intelligence Research (JAIR) 45*, 165–196.

[92] Voice, T., Ramchurn, S. D., and Jennings, N. R. 2012. On coalition formation with sparse synergies. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*. AAMAS '12. Richland, SC, 223–230.

[93] Yeh, D. Y. 1986. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics 26,* 4, 467–474.

[94] Yi, S.-S. 2003. Endogenous formation of economic coalitions: A survey on the partition function approach. In *The Endogenous Formation of Economic Coalitions*. Edward Elgar, London, UK, 80–127.

[95] Zilberstein, S. 1996. Using anytime algorithms in intelligent systems. *AI Magazine 17,* 3, 73–83.

[96] Zlotkin, G. and Rosenschein, J. S. 1994. Coalition, cryptography, and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the 12th AAAI Conference on Artificial Intelligence (AAAI)*. 432–437.

[97] Zuckerman, D. 2007. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing 3,* 6, 103–128.

ACCEPTED MANUSCRIPT

ACCEPTED MANUSCRIPT

Appendix A. Appendix: Summary of Main Notation

A	the set of agents (or players).
a_i	an agent (or a player) in A.
n	the number of agents in A.
2^A	the power set of A, i.e., the set of all possible coalitions.
C	a coalition.
CS	a coalition structure.
CS^*	an optimal coalition structure.
CS^{**}	the best coalition structure found at any point in time (i.e., the current best solution found so far).
β	the established bound on the quality of CS^{**}, i.e., $\frac{V(CS^*)}{V(CS^{**})} \leq \beta$.
Π^A	the set of all coalition structures over A.
Π^A_s	the set of all coalition structures over A that are of size s.
Π^A_I	the set of all coalition structures over A in which the coalition sizes match the parts in integer partition I.
Π^C	the set of all partitions of coalition C.
Π^I	the set of possible partitions of the integer partition I.
Π^{CS}	the set of possible partitions of the coalition structure CS.
P	a partition.
p	an element in partition P.
(C, CS)	a coalition C embedded in a coalition structure CS.
$w(C, CS)$	the value of C in CS given a partition function game.
$W(P, CS)$	the value of P in CS given a partition function game, i.e., $\sum_{C \in P} w(P, CS)$.
$W(CS)$	the value of CS given a partition function game, i.e., $\sum_{C \in CS} w(C, CS)$.
$V(CS)$	the value of CS given a characteristic function game, i.e., $\sum_{C \in CS} v(C)$.
$v(C)$	the value of the coalition C given a characteristic function game.
$f(C)$	the value of the optimal partition of C.
$\epsilon(C, CS, CS')$	the externality imposed on C by forming CS from CS'.
$\mathcal{I}^k$	the set of all integer partitions of some integer, k.
I	an integer partition.
Max^A_s	the maximum value of all coalitions of size s.
Avg^A_s	the average value of all coalitions of size s.
UB^*	upper bound on the value of CS^*.
LB^*	lower bound on the value of CS^*.
UB^A_I	upper bound on the value of the best coalition structure in Π^A_I.
LB^A_I	lower bound on the value of the best coalition structure in Π^A_I.
$\mathcal{M}$	the set of movements in the coalition structure graph.
M^*	the subset of $\mathcal{M}$ evaluated by the algorithm ODP.
M^{**}	the subset of $\mathcal{M}$ evaluated by the IDP—the size-based version of ODP.

ACCEPTED MANUSCRIPT