FACS: Fast Code-based Algorithm for Coalition Structure Generation (Student Abstract)

Redha Taguelmimt[1], Samir Aknine[1], Djamila Boukredera[2], Narayan Changder[3]

[1] LIRIS, Lyon 1 University, France
[2] Laboratory of Applied Mathematics, Faculty of Exact Sciences, University of Bejaia, 06000 Bejaia, Algeria
[3] National Institute of Technology Durgapur, India
redha.taguelmimt@gmail.com, samir.aknine@univ-lyon1.fr, boukredera@hotmail.com, narayan.changder@gmail.com

Abstract

In this paper, we propose a new algorithm for the Coalition Structure Generation (CSG) problem that can be run with more than 28 agents while using a complete set of coalitions as input. The current state-of-the-art limit for exact algorithms to solve the CSG problem within a reasonable time is 27 agents. Our algorithm uses a novel representation of the search space and a new code-based search technique. We propose an effective heuristic search method to efficiently explore the space of coalition structures using our code-based technique and show that our method outperforms existing state-of-the-art algorithms by multiple orders of magnitude.

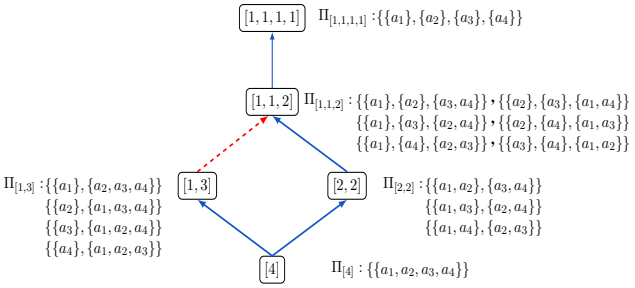

Figure 1: A four-agent example of the IP graph.

CSG Problem Formulation and Preliminaries

In this paper, we propose a new algorithm for the CSG problem that can be run with more than 28 agents while using a complete set of coalitions as input. ODP-IP (Michalak et al. 2016) and ODSS (Changder et al. 2020) algorithms are very efficient for solving many problem instances. However, in case time is limited, an approach that gives good enough quality solutions, within a reasonable time, is more valuable (Wu and Ramchurn 2020). In the CSG problem, given a set $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ of n agents, any subset of $\mathcal{A}$ is called a coalition. A coalition structure $\mathcal{CS} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_k\}$ is a partition of $\mathcal{A}$, where $k = |\mathcal{CS}|$, $\forall \mathcal{C}_i \in \mathcal{CS}$, $\mathcal{C}_i \neq \emptyset$, $\bigcup_{i=1}^{k} \mathcal{C}_i = \mathcal{A}$ and for all $i, j \in \{1, 2, ..., k\}$ where $i \neq j$, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$. The set of all coalition structures is $\Pi(\mathcal{A})$. The value $V(\mathcal{CS})$ of a coalition structure $\mathcal{CS}$ is assessed as the sum of the values $v(\mathcal{C})$ of its composing coalitions: $V(\mathcal{CS}) = \sum_{\mathcal{C} \in \mathcal{CS}} v(\mathcal{C})$. The optimal solution of the CSG problem is the highest valued coalition structure $\mathcal{CS}^* \in \Pi(\mathcal{A})$, where $\mathcal{CS}^* = \arg\max_{\mathcal{CS} \in \Pi(\mathcal{A})} V(\mathcal{CS})$.

Novel Search Space Representation

First, let us recall the principle of the integer partition (IP) graph (Rahwan et al. 2007), which divides the space of all the coalition structures into subspaces that are each represented by an integer partition of n. For instance, for $n = 4$ agents, the set of integer partitions is: $\{[4], [1, 3], [2, 2], [1, 1, 2], [1, 1, 1, 1]\}$. In the IP graph, each partition $\mathcal{P}$ of n is represented by a node, where $\mathcal{P}$ represents a set of coalition structures in which the size of the coalitions matches the parts of $\mathcal{P}$ (see Figure 1). For example, the node [2,2] represents all coalition structures that contain two coalitions of size 2. In our coalition structure representation, we codify each coalition $\mathcal{C}_i$ with a code (i.e., an index-based code). As a result, each agent of a given coalition $\mathcal{C}_i$ will be encoded with a number i, which corresponds to the index of the coalition to which it belongs. By doing so, each coalition structure is defined as a vector of size n. Each position p in the vector identifies the agent $p_{/p=1..n}$ while the code in position p of the vector identifies the coalition to which the agent p belongs. Hence, a coalition structure will be encoded by a set of $x_{i/i=1..n} \geq 0$ numbers. For example, let $\mathcal{A}$ be a set of $n = 10$ agents. Consider the coalition structure $\mathcal{CS} = \{\{a_3\}, \{a_2, a_4, a_6\}, \{a_1, a_5, a_7, a_8, a_9, a_{10}\}\}$ containing three coalitions: $\mathcal{C}_0 = \{a_3\}$, $\mathcal{C}_1 = \{a_2, a_4, a_6\}$ and $\mathcal{C}_2 = \{a_1, a_5, a_7, a_8, a_9, a_{10}\}$. $\mathcal{C}_0$, $\mathcal{C}_1$ and $\mathcal{C}_2$ are encoded with codes 0, 1 and 2, respectively. $\mathcal{CS}$ will be encoded by the vector of codes [x_1 x_2 ... x_{10}], where $x_{i/i=1..10} = j \Leftrightarrow a_i \in \mathcal{C}_j$. $\mathcal{CS}$ is encoded here with the vector of codes [2 1 0 1 2 1 2 2 2 2], where the number of different codes equals the number of coalitions forming the corresponding coalition structure and the size of the vector equals n. Any permutation of these numbers provides a different coalition structure. For example, the vector of codes [0 1 1 1 2 2 2 2 2 2] represents the coalition structure $\mathcal{CS} = \{\{a_1\}, \{a_2, a_3, a_4\}, \{a_5, a_6, a_7, a_8, a_9, a_{10}\}\}$.

Generating all the combinations of these numbers guarantees exploration of all the coalition structures represented by each node. Let us now generalize this representation for the entire IP graph. For each node in level l, we have l

coalitions, we then codify each coalition $\mathcal{C}_i$ with a code $i \in \{0, 1, .., l-1\}$. Then, to represent with a vector of codes each coalition structure $\mathcal{CS}$ of this node, each coalition $\mathcal{C}_i$ of size k is depicted by k times the number (code) i.

Code-based Coalition Structure Generation

FACS[1] partially enumerates the coalition structures of each node of the IP graph. Specifically, we start by generating all the integer partitions of n and searching each node of the IP graph. Once all the possible partitions of the integer n are generated, the FACS algorithm proceeds in two main phases.

First, FACS partially generates the permutations between the codes of the vectors. Consider the node $[1, 3, 6]$. To enumerate the coalition structures represented by this node, we use the codes $0, 1$ and 2. Rather than computing all the combinations of the 10 numbers (which is not efficient due to the high computational load), FACS calculates the combinations of the three codes, $\{0, 1, 2\}$ that represent the coalitions. We then obtain these combinations: $\vartheta = \{\{0, 1, 2\}, \{0, 2, 1\}, \{1, 0, 2\}, \{1, 2, 0\}, \{2, 0, 1\}, \{2, 1, 0\}\}$. For each combination in ϑ, FACS calculates the first coalition structure of the combination called initialization. Each code in the combination will then be repeated as many times as the size of the coalition it represents. This means that in each initialization, the agents are assigned to the coalitions by respecting the order of these coalitions in the combination. For example, the initialization of $\{0, 1, 2\}$ is [0 1 1 1 2 2 2 2 2 2] because the coalition encoded with 0 is of size 1, the coalition encoded with 1 is of size 3 and the coalition encoded with 2 is of size 6. The different initializations thus obtained will then be used in the second phase to generate the different permutations.

Second, FACS generates the permutations of the codes composing the vectors obtained at the end of phase 1. Each newly generated vector of codes after each permutation will then be associated with a different coalition structure. For each initialization vector performed in phase 1, FACS computes the permutations as follows. First, FACS starts with the first agent (first number) and permutes its code with each of the codes of the other agents. Then, FACS moves to the next agent and applies the same permutation operations to its code. This process is then iterated until FACS reaches the last code of the concerned vector.

Empirical Evaluation

We evaluated the performance of our algorithm on several value distributions given different numbers of agents (20 to 30) and compared it with CSG-UCT (Wu and Ramchurn 2020) and ODP-IP. For CSG-UCT and ODP-IP, we used the codes provided by the authors. Figure 2 illustrates the execution time of FACS, ODP-IP and CSG-UCT on different distributions. For each point in Figure 2, we conducted an average of 50 tests. We tested more than 12 distributions, but we show the results for only some of them. As we can see, for all these value distributions, our algorithm outperforms all the considered algorithms. For example, with 25 agents for the

[1]FACS stands for **F**ast **A**gorithm for **C**oalition **S**tructure generation

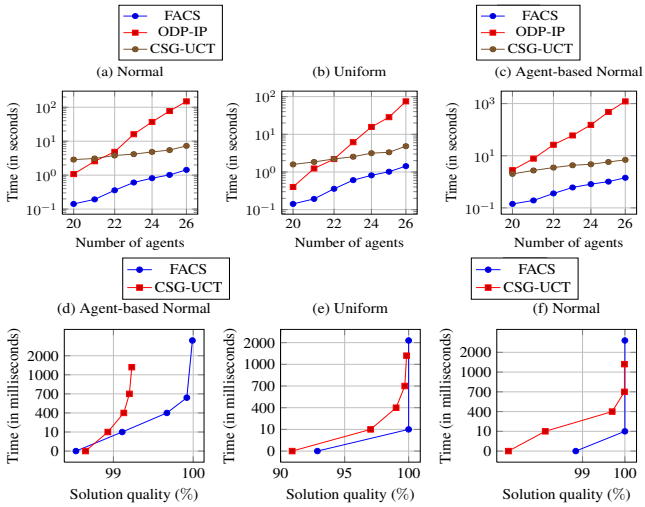

Figure 2: The first figures a, b, c depict the time performance in seconds of FACS, ODP-IP and CSG-UCT for a number of agents between 20 and 26. The figures d, e, f depict the time taken to produce a certain solution quality by FACS and CSG-UCT for 30 agents. Here, solution quality is calculated by using the formula: $\frac{v(\mathcal{CS})}{max(v(\mathcal{CS}^+_{FACS}), v(\mathcal{CS}^+_{CSG-UCT}))}$, where $v(\mathcal{CS})$ is the current best solution of the algorithm.

Normal distribution, the ratio of ODP-IP time and our algorithm time is 75.9, meaning that our algorithm takes 0.013% of the time taken by ODP-IP to provide a good quality solution (99%). FACS can be run with more than 28 agents. To demonstrate this, we conducted experiments on several value distributions and showed the solution quality of FACS and CSG-UCT given 30 agents. As no exact algorithm is run for these sets of agents, the solution quality shown in Figure 2 is calculated as follows: $\frac{v(\mathcal{CS})}{max(v(\mathcal{CS}^+_{FACS}), v(\mathcal{CS}^+_{CSG-UCT}))}$, where $v(\mathcal{CS}^+_{FACS})$ and $v(\mathcal{CS}^+_{CSG-UCT})$ are the values of the best solutions provided by FACS and CSG-UCT, respectively. As we can see, FACS yields good quality solutions when run with more than 28 agents, compared to those provided by CSG-UCT. Moreover, for several distributions, solution quality exceeds 99% after about 10 milliseconds.

References

Changder, N.; Aknine, S.; Ramchurn, S. D.; and Dutta, A. 2020. ODSS: Efficient Hybridization for Optimal Coalition Structure Generation. In *Proc. of AAAI*, 7079–7086.

Michalak, T.; Rahwan, T.; Elkind, E.; Wooldridge, M.; and Jennings, N. R. 2016. A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence* 230: 14–50.

Rahwan, T.; Ramchurn, S. D.; Dang, V. D.; and Jennings, N. R. 2007. Near-Optimal Anytime Coalition Structure Generation. In *Proc. of IJCAI*, volume 7, 2365–2371.

Wu, F.; and Ramchurn, S. D. 2020. Monte-Carlo Tree Search for Scalable Coalition Formation. In *Proc. of IJCAI*, 407–413.