Parallel Index-Based Search 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] TCG Centres for Research and Education in Science and Technology, Kolkata, India
redha.taguelmimt@gmail.com, samir.aknine@univ-lyon1.fr, djamila.boukredera@univ-bejaia.dz, narayan.changder@tcgcrest.org

Abstract

In this paper, we propose a novel algorithm to address the Coalition Structure Generation (CSG) problem. Specifically, we use a representation of the search space that enables it to be explored in a new way. We introduce an index-based exact algorithm. Our algorithm is anytime, produces optimal solutions, and can be run on large-scale problems with hundreds of agents. Our experimental evaluation on a benchmark with several value distributions shows that the representation of the search space that we combined with the proposed algorithm provides high-quality results for the CSG problem and outperforms existing state-of-the-art algorithms.

Problem Formulation and Preliminaries

Coalition Formation aims at finding the optimal coalition structure that maximizes social welfare. However, the search space of coalition structures is often too large to be fully explored. This paper presents a new algorithm for CSG that outperforms the fastest state-of-the-art algorithms for both small-scale and large-scale problems. A CSG problem defined on a set of n agents $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ is a problem of size n. A coalition $\mathcal{C}$ is any non-empty subset of $\mathcal{A}$. The size of $\mathcal{C}$ is $|\mathcal{C}|$. In CSG, a characteristic function v assigns a real value to each coalition. This value determines the efficiency of the coalition. A coalition structure $\mathcal{CS}$ is a partition of the set of agents $\mathcal{A}$ into disjoint coalitions. Formally, given a set of non-empty coalitions $\{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_k\}$, $\mathcal{CS} = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_k\}$, where $k = |\mathcal{CS}|$, which satisfies the following constraints: $\bigcup_{j=1}^{k} \mathcal{C}_i = \mathcal{A}$ and for all $i, j \in \{1, 2, ..., k\}$ where $i \neq j$, $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$. The value of $\mathcal{CS}$ is assessed as: $V(\mathcal{CS}) = \sum_{\mathcal{C} \in \mathcal{CS}} v(\mathcal{C})$. The goal in CSG is to find the optimal solution $\mathcal{CS}^* = \arg\max_{\mathcal{CS} \in \Pi(\mathcal{A})} V(\mathcal{CS})$.

Index-based Representation

The index-based representation comes with the idea of representing the space of coalition structures using integers. We rely on the representation presented in [Taguelmimt et al. 2021]. It presents a higher layer above an integer partition (IP) graph defined initially in [Rahwan et al. 2007], which divides the space of coalition structures into subspaces that are each represented by an integer partition of n. For instance, for $n = 4$ agents, the integer partitions are: $[4], [1, 3], [2, 2], [1, 1, 2], [1, 1, 1, 1]$. Each level $l \in \{1, 2, .., n\}$ in the IP graph contains nodes representing integer partitions of n containing l parts. For instance, the integer partitions of n of level 2 have two parts. Each integer partition $\mathcal{P}$ represents a set of coalition structures in which the sizes of the coalitions match the parts of $\mathcal{P}$. For example, the node [1,1,2] consists of all coalition structures that contain two coalitions of size 1 and one coalition of size 2. With the index-based representation, each node of the IP graph represents a set of coalition structures meeting the criteria of the node. For example, the node $[1, 1, 2]$ in the IP graph (see Figure 1) represents all coalition structures containing three disjoint coalitions $\mathcal{C}_0$, $\mathcal{C}_1$ and $\mathcal{C}_2$ where $|\mathcal{C}_0| = 1$, $|\mathcal{C}_1| = 1$ and $|\mathcal{C}_2| = 2$. For each such node in the IP graph, we represent each coalition with its index. For instance, C_0 is represented with index 0, C_1 with index 1, etc. Here, we precisely define this mapping by the function $\mathcal{F}$, which matches each coalition $\mathcal{C}_j$ with $\mathcal{F}(\mathcal{C}_j) = j$.

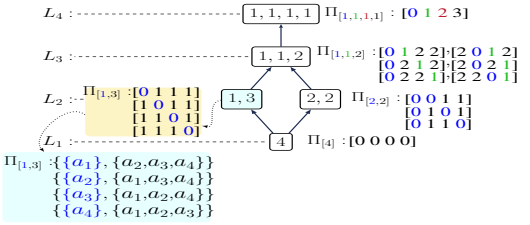

Figure 1: Example of the index-based IP graph with 4 agents. Each vector corresponds to a coalition structure. For example, the coalition structures of the blue rectangle correspond to the vectors highlighted in the yellow rectangle.

Now, we represent each coalition structure with a vector of indexes of size $n = |\mathcal{A}|$: $[x_1, x_2, .., x_p, .., x_n]$, where each index in position $p_{/p=1..n}$ represents the coalition to which a corresponding agent a_i belongs. For the remainder of this section, the index in position p of the vector represents the coalition to which the agent a_p belongs (i.e. $i = p$). For instance, let $\mathcal{A}$ be a set of $n = 6$ agents. Consider the coalition structure $\mathcal{CS}_1 = \{\{a_3\}, \{a_1, a_5\}, \{a_2, a_4, a_6\}\}$ of the node $[1, 2, 3]$ containing three coalitions: $\mathcal{C}_0 = \{a_3\}$, $\mathcal{C}_1 = \{a_1, a_5\}$ and $\mathcal{C}_2 = \{a_2, a_4, a_6\}$. $\mathcal{C}_0$, $\mathcal{C}_1$ and $\mathcal{C}_2$ are represented with indexes 0, 1 and 2, respectively. $\mathcal{CS}_1$

is encoded by the vector of indexes $[x_1\ x_2\ ..\ x_p\ ..\ x_6]$, where $x_{p/p=1..6} = j \Leftrightarrow a_p \in \mathcal{C}_j$. $\mathcal{CS}_1$ is encoded here with the vector of indexes [1 2 0 2 1 2] of size $n = 6$, where the number of different indexes equals the number of coalitions forming $\mathcal{CS}_1$. For example, the index associated with a_3 in this vector is 0 because a_3 belongs to $\mathcal{C}_0$ in $\mathcal{CS}_1$. Any permutation of these indexes provides a different coalition structure. For example, the vector of indexes [0 2 2 2 1 1] represents the coalition structure $\mathcal{CS}_2 = \{\{a_1\}, \{a_5, a_6\}, \{a_2, a_3, a_4\}\}$.

The PICS Algorithm

Given the integer partitions of n and an ordered list of agents[1], PICS (**P**arallel **I**ndex-based **C**oalition **S**tructure generation) partially generates the coalition structures of each node of the Index-based Integer Partition (IIP) graph (see Fig. 1). To explain the principle of PICS, let us consider this ordered list of agents $[a_1\ a_2\ ...\ a_n]$.

For each node in the IIP graph, PICS generates the vectors that represent the coalition structures by permuting the indexes of some selected initial vectors. PICS starts by generating these initial vectors, which we call initializations. Let us consider the previous example of the node $[1, 2, 3]$, which contains 3 coalitions represented by indexes 0, 1 and 2. To generate these initializations, PICS generates the combinations of the indexes that represent the coalitions (i.e. 0, 1 and 2). As a result, we obtain these combinations: $\vartheta = \{\{0, 1, 2\}, \{0, 2, 1\}, \{1, 0, 2\}, \{1, 2, 0\}, \{2, 0, 1\}, \{2, 1, 0\}\}$. For each such combination, PICS generates the initialization as follows. The indexes in the initialization are positioned according to the order of indexes in the combination. Each index in the combination is then repeated as many times as the size of the coalition it represents. For instance, the initialization of $\{0, 2, 1\}$ is [0 2 2 2 1 1] because the coalitions represented with 0, 1, 2 are of sizes 1, 3, and 2, respectively.

Now, given a set of initializations, PICS permutes the indexes of these vectors to generate the coalition structures. Each newly generated vector of indexes after each permutation is then associated with a different coalition structure. For each initial vector, PICS generates the permutations as follows. First, PICS starts with the first index of the initialization and permutes it with each of the other indexes. Then, PICS moves to the next index and applies the same permutation operations to it. This process is then iterated until PICS reaches the last index of the initial vector concerned.

To speed up finding high-quality solutions, PICS explores different lists of agents. Consider the previous example of the node $[1, 2, 3]$ and the initial vector $\mathcal{I}$ = [2 2 2 1 1 0], which is the initialization of the combination $\{2, 1, 0\}$. Now, assume that the ordered list of agents used is $\mathcal{L}$ = $[a_1\ a_5\ a_6\ a_4\ a_3\ a_2]$. For this list $\mathcal{L}$, the p^{th} index in $\mathcal{I}$ represents the index of the coalition to which the agent $a_i = \mathcal{L}(p)$ belongs. For instance, the second index of $\mathcal{I}$ (i.e. $p = 2$) represents the coalition of the agent $\mathcal{L}(2) = a_5$. Thus, the coalition structure associated with

[1]For example, with 6 agents, $[a_2\ a_4\ a_1\ a_6\ a_3\ a_5]$ is an ordered list of agents.

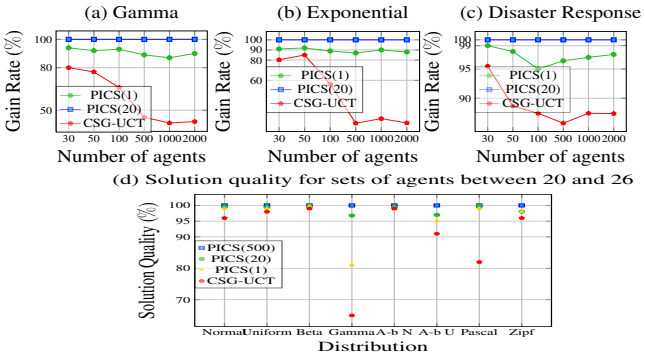

Figure 2: Gain rate and solution quality of PICS and CSG-UCT.

$\mathcal{I}$ is $\{\{\mathcal{L}(6)\}, \{\mathcal{L}(4), \mathcal{L}(5)\}, \{\mathcal{L}(1), \mathcal{L}(2), \mathcal{L}(3)\}\}$, which corresponds to $\{\{a_2\}, \{a_3, a_4\}, \{a_1, a_5, a_6\}\}$. To allow PICS to find the optimal coalition structure faster, PICS runs multiple processes in parallel. Each of these runs a different list of agents. To generate these lists, PICS shuffles the ordered list $[a_1\ a_2\ ...\ a_n]$ to obtain a new list that is forwarded to one process. The number of processes used and the result of using them are discussed in Section 4.

Empirical Evaluation

To evaluate the performance of PICS, we compare it to CSG-UCT [Wu and Ramchurn 2020]. Our algorithm was implemented in Java, and we used the code provided by the authors of CSG-UCT. We tested more than 12 distributions, but we show the results for only some of them. Figure 2.d shows the solution quality obtained by the algorithms. As we can see, PICS outperforms CSG-UCT when using one process (see the results for PICS(1) in Figure 2.d). PICS produces better solutions when using 20 processes and it produces optimal solutions when using 500 processes (see the results for PICS (20) and PICS(500) in Figure 2.d). Figures 2.a, 2.b, 2.c depict the gain rate of PICS and CSG-UCT for large-scale problems when PICS uses 20 processes. The *gain rate* is computed as $\frac{v(\mathcal{CS})}{max(v(\mathcal{CS}^+_{PICS}), v(\mathcal{CS}^+_{CSG-UCT}))}$, where $v(\mathcal{CS}^+_{PICS})$ and $v(\mathcal{CS}^+_{CSG-UCT})$ are the values of the best solutions provided by PICS and CSG-UCT, respectively. As can be seen, PICS outperforms CSG-UCT with better gain rates in all of the considered value distributions, achieving gain rates up to 50% higher than CSG-UCT for problems with more than 100 agents.

References

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.

Taguelmimt, R.; Aknine, S.; Boukredera, D.; and Changder, N. 2021. Code-based Algorithm for Coalition Structure Generation. In *ICTAI*, 1075–1082.

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