Efficient Size-based Hybrid Algorithm for Optimal Coalition Structure Generation

Extended Abstract

Redha Taguelmimt
Univ Lyon, UCBL, CNRS, INSA Lyon, Centrale Lyon, Univ Lyon 2, LIRIS, UMR5205
F-69622 Villeurbanne, France
redha.taguelmimt@gmail.com

Samir Aknine
Univ Lyon, UCBL, CNRS, INSA Lyon, Centrale Lyon, Univ Lyon 2, LIRIS, UMR5205
F-69622 Villeurbanne, France
samir.aknine@univ-lyon1.fr

Djamila Boukredera
University of Bejaia, Faculty of Exact Sciences, Laboratory of Applied Mathematics
06000 Bejaia, Algeria
djamila.boukredera@univ-bejaia.dz

Narayan Changder
TCG Centres for Research and Education in Science and Technology
Kolkata, India
narayan.changder@tcgcrest.org

Tuomas Sandholm
Carnegie Mellon University
Strategic Machine, Inc.
Strategy Robot, Inc.
Optimized Markets, Inc.
Pittsburgh, USA
sandholm@cs.cmu.edu

ABSTRACT

Coalition Structure Generation (CSG) involves dividing agents into coalitions in such a way as to coordinate them into solving problems together efficiently. In this paper, we revisit the CSG problem and propose a new search method that introduces an offline phase to speed up the search process, where the best coalition sets to search are preprocessed. These sets are calculated only once regardless of the coalition values and can be reused each time a CSG instance is to be solved. Then our search in the online phase combines dynamic programming with integer partition-based search in a novel way.

KEYWORDS

Coalition formation; Coalition structure generation; Preprocessing; Integer partition graph

ACM Reference Format:
Redha Taguelmimt, Samir Aknine, Djamila Boukredera, Narayan Changder, and Tuomas Sandholm. 2024. Efficient Size-based Hybrid Algorithm for Optimal Coalition Structure Generation: Extended Abstract. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 3 pages.

1 PROBLEM FORMULATION

The design of algorithms for CSG has been an active research topic in AI since the 1990s [1–13]. This paper presents an exact algorithm for the CSG problem that outperforms the fastest state-of-the-art algorithms. While ODP-IP [4] and BOSS [3], which combine IDP [5] and IP [7], are very efficient for many value distributions, there are instances where they fail to produce the exact results within a reasonable time. 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 C is any non-empty subset of $\mathcal{A}$. 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 $\{C_1, C_2, ..., C_k\}$, $\mathcal{CS} = \{C_1, C_2, ..., C_k\}$, where $k = |\mathcal{CS}|$ and $\mathcal{CS}$ satisfies the following constraints: $\bigcup_{j=1}^{k} C_i = \mathcal{A}$ and for all $i, j \in \{1, 2, ..., k\}$ where $i \neq j$, $C_i \cap C_j = \emptyset$. The value of $\mathcal{CS}$ is assessed as: $v(\mathcal{CS}) = \sum_{C \in \mathcal{CS}} v(C)$. The goal in CSG is to find the optimal solution $\mathcal{CS}^* = \arg\max_{\mathcal{CS} \in \Pi(\mathcal{A})} v(\mathcal{CS})$. The Integer Partition (IP) graph [7] (see Figure 1) divides the solutions into subspaces that are represented by integer partitions of n.

2 THE CESAR ALGORITHM

Our method is a hybrid solution that combines dynamic programming and integer partition based techniques. It consists of an offline preprocessing phase and an online search phase.

The offline phase computes the best set of coalition sizes that determine the coalitions to evaluate in the online phase, regardless of their values, to speed up the search in the IP graph (see Figure 1). The offline phase is performed once for each CSG problem size irrespective of the online phase. The principle of the offline phase is to choose certain coalition sizes to evaluate, which can be illustrated on the IP graph as follows. For example with ten agents, dividing a coalition of size 2 into two coalitions of size 1, when searching for the solutions (i.e. in the online phase), corresponds to an upward movement in this graph from the node $[2, 8]$ to the node $[1, 1, 8]$ or from $[1, 1, 2, 6]$ to $[1, 1, 1, 1, 6]$, etc. By choosing to split in this graph a subset of integers starting from the bottom node, a subset of nodes in the IP graph become reachable from the bottom node. From this, we observe that to search at least a certain percentage of the solution subspaces, several sets of coalition sizes could be considered to reach that percentage, with a different runtime for each set of sizes due to the different cost of the explored sizes. To do this, we propose

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

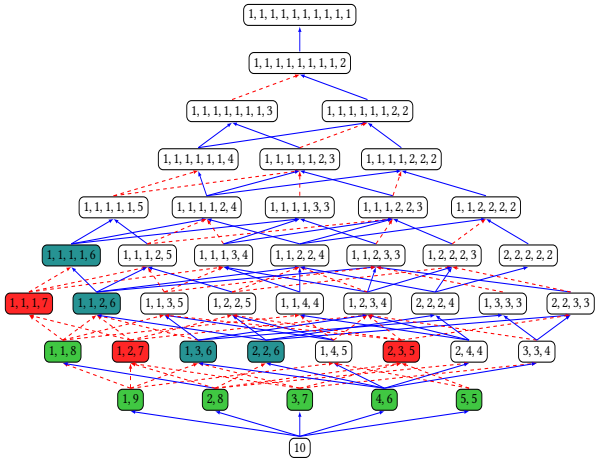

Figure 1: A ten-agent example of the integer partition graph. (a) SDP evaluates the coalitions of size $s \in \mathcal{BS}_1$={2, 4, 6, 10}. The blue edges represent the splittings of coalitions of size $s \in \mathcal{BS}_1$. The red edges represent the skipping of coalitions of size $s \in \{3, 5, 7, 8, 9\}$. After finishing the evaluation process, SDP will have fully searched, with the set $\mathcal{BS}_1$, all the subspaces, except the red ones, which are not connected to the bottom node through a series of blue edges.
(b) SIP evaluates the subspaces starting with the most promising ones and prunes out those that have no chance of improving the solution. When SDP finishes evaluating the coalitions of size 2, all the green nodes are searched, meaning that SIP does not need to search them. When SDP finishes evaluating the coalitions of size 4, all the blue nodes along with the green ones are now searched and SIP does not need to search them either. Finally, when SDP finishes evaluating the coalitions of size 6, all the nodes of the graph are searched, except the red nodes, meaning that SIP no longer needs to search the nodes already searched by SDP. The remaining red nodes are searched by either SIP or SDP when considering the set $\mathcal{BS}_2$={2, 3, 4, 5, 6, 10} in parallel.

and use the Size Extraction Algorithm (SEA). First, SEA starts by computing the time required to evaluate the coalitions of each size. Then, SEA computes the best coalition size set that searches at least 70% of the solution subspaces with minimum runtime. The 70% is a hyperparameter that we optimized. The remaining 30% are searched with a second technique that we detail in the online phase. SEA yields the best size set by evaluating all possible size sets and selecting the best one. For example, with 10 agents, SEA returns $\mathcal{BS}_1 = \{2, 4, 6, 10\}$ that searches 39 subspaces out of 42.

The online phase computes the optimal solution using the CESAR (Coalition-size Extraction and Subspace seArch Redesign) algorithm, which combines a Size-based Dynamic Programming algorithm (SDP) with a Size-based Integer-Partition algorithm (SIP).

SDP consists of two parts that run in parallel to increase performance and optimality. Each part of SDP evaluates the different ways of splitting a set of coalitions. The first part evaluates all the coalitions whose sizes belong to the best coalition size set obtained from the offline phase. The second part of SDP evaluates all the

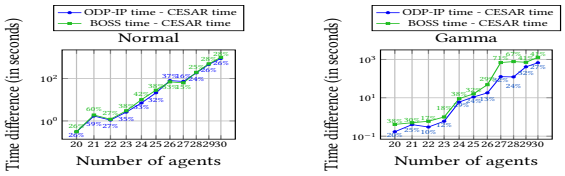

Figure 2: Time difference between CESAR and ODP-IP, and CESAR and BOSS, for a range of agents between 20 and 30. The percentages indicate the time gain achieved by CESAR.

coalitions of size $s \in \mathcal{BS}_2 = \{2, ..., \lfloor \frac{2n}{3} \rfloor, n\}$ to guarantee searching all the solution subspaces. [5] proved that evaluating the set of coalitions of size 2 to $\lfloor \frac{2n}{3} \rfloor$ for a problem of size n is sufficient to guarantee finding the exact result. SDP computes the optimal coalition structure by evaluating all possible ways of splitting each coalition C of a selected size into two coalitions and checks whether it is beneficial to split C or not. For each coalition, the splitting that generates its optimal value along with this value are stored in two tables, the partition table P_t and the value table V_t, respectively. This allows the algorithm to compute the optimal coalition structure, once it has finished evaluating all selected coalitions, in a recursive manner starting with the grand coalition. The splitting operations of SDP can be visualized in the IP graph as upward movements through edges. Figure 1.a details the SDP algorithm.

SIP is based on the IP graph and considers each node as a subspace of solutions. This makes it possible to compute the upper bounds on the quality of the best solution in each subspace. By comparing the bounds of the subspaces, SIP explores the most promising ones. SIP progressively improves the solutions by exploring the solutions of the subspaces and eliminating the subspaces that do not have a better upper bound than the last best solution found.

The search in SIP is combined with SDP. When SDP finishes evaluating all the coalitions of size x, a certain number of subspaces become reachable from the bottom node. Hence, they are fully searched by SDP, which finds the optimal coalition structure among them. This means that SIP does not need to search them anymore, thereby pruning them out. Figure 1.b details the CESAR algorithm.

3 EMPIRICAL EVALUATION

We evaluated the performance of our algorithm on several value distributions given different numbers of agents (20 to 27). Figure 2 shows the performance of CESAR compared to ODP-IP and BOSS. All the codes are written in Java. We see that for all distributions, CESAR performs better than ODP-IP and BOSS. For example, for the Normal distribution with 30 agents, there is a difference of 15 minutes between CESAR and ODP-IP and a difference of 17 minutes with BOSS. The percentage of improvement of CESAR compared to ODP-IP and BOSS is 26% and 28%, respectively. Moreover, we tested the algorithms on a total of 15 value distributions and observed that CESAR is always faster than ODP-IP.

ACKNOWLEDGMENTS

Tuomas Sandholm is supported by the Vannevar Bush Faculty Fellowship ONR N00014-23-1-2876, National Science Foundation grants RI-2312342 and RI-1901403, and ARO award W911NF2210266.

REFERENCES

[1] Narayan Changder, Samir Aknine, and Animesh Dutta. 2019. An Imperfect Algorithm for Coalition Structure Generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 9923–9924.

[2] Narayan Changder, Samir Aknine, Sarvapali D Ramchurn, and Animesh Dutta. 2020. ODSS: Efficient Hybridization for Optimal Coalition Structure Generation.. In *Proc. of AAAI*. 7079–7086.

[3] Narayan Changder, Samir Aknine, Sarvapali D. Ramchurn, and Animesh Dutta. 2021. BOSS: A Bi-directional Search Technique for Optimal Coalition Structure Generation with Minimal Overlapping (Student Abstract). In *Proc. of AAAI*, Vol. 35. 15765–15766. https://ojs.aaai.org/index.php/AAAI/article/view/17879

[4] Tomasz Michalak, Talal Rahwan, Edith Elkind, Michael Wooldridge, and Nicholas R Jennings. 2016. A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence* 230 (2016), 14–50.

[5] Talal Rahwan and Nicholas R Jennings. 2008. An improved dynamic programming algorithm for coalition structure generation. In *Proc. of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*. International Foundation for Autonomous Agents and Multiagent Systems, 1417–1420.

[6] Talal Rahwan, Tomasz P Michalak, Michael Wooldridge, and Nicholas R Jennings. 2015. Coalition structure generation: A survey. *Artificial Intelligence* 229 (2015), 139–174.

[7] Talal Rahwan, Sarvapali D Ramchurn, Nicholas R Jennings, and Andrea Giovannucci. 2009. An anytime algorithm for optimal coalition structure generation. *Journal of artificial intelligence research* 34 (2009), 521–567.

[8] Tuomas Sandholm, Kate Larson, Martin Andersson, Onn Shehory, and Fernando Tohmé. 1999. Coalition structure generation with worst case guarantees. *Artificial Intelligence* 111, 1 (1999), 209–238.

[9] Redha Taguelmimt, Samir Aknine, Djamila Boukredera, and Narayan Changder. 2022. Subspace-Focused Search Method for Optimal Coalition Structure Generation. In *2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI)*. 1435–1440.

[10] Redha Taguelmimt, Samir Aknine, Djamila Boukredera, and Narayan Changder. 2023. Anytime Index-Based Search Method for Large-Scale Simultaneous Coalition Structure Generation and Assignment. In *Proceedings of the 26th European Conference on Artificial Intelligence, ECAI-23*. 2282–2289. Main Track.

[11] Redha Taguelmimt, Samir Aknine, Djamila Boukredera, Narayan Changder, and Tuomas Sandholm. 2023. Optimal Anytime Coalition Structure Generation Utilizing Compact Solution Space Representation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*. 309–316. Main Track.

[12] Feng Wu and Sarvapali D Ramchurn. 2020. Monte-Carlo Tree Search for Scalable Coalition Formation. In *Proc. of IJCAI*. 407–413.

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