Coalition Structure Generation Utilizing Graphical Representation of Partition Function Games

Kazuki Nomoto, Yuko Sakurai, Makoto Yokoo
Graduate School and Faculty of Information Science and Electrical Engineering
Kyushu University
744 Motooka Nishi-ku Fukuoka, 819-0395, Japan
{nomoto@agent., ysakurai@, yokoo@} inf.kyushu-u.ac.jp

Introduction

Forming effective coalitions is a central research challenge in AI and multi-agent systems. The coalition structure generation (CSG) problem, which is a well-known major research topics in coalitional games (Sandholm et al. 1999), is to partition a set of agents into coalitions to maximize the sum of utilities.

We study the CSG problem for partition function games (PFGs), where the value of a coalition differs depending on the formation of other coalitions generated by non-member agents. For example, consider restructuring and the consolidation of multiple companies in a market. If several competitors are merged into a single entity, the competitor of this new large company might lose sales. The market must consider the best way of restructuring the merged companies to maximize the expected sales. Such a problem is represented as a PFG. In PFGs, the input of a coalitional game is a black-box function called a partition function that maps an embedded coalition (a coalition and the coalition structure) to its value. Representing an arbitrary partition function explicitly requires $\Theta(n^n)$ numbers, which are prohibitive for large n.

Recently, several concise representation schemes for a partition function have been proposed (Skibski et al. 2015; Rahwan et al. 2015). Among them, a partition decision tree (PDTs) is a graphical representation based on multiple rules. In this paper, we propose new algorithms that can solve CSG problems by PDTs representation. More specifically, we modify PDTs representation to effectively handle negative rules and apply the depth-first branch-and-bound algorithm. We experimentally show that our algorithm can solve CSG problems reasonably well[1].

CSG Problem

Let $A = \{a_1, a_2, \ldots, a_n\}$ be the set of agents and let $C \subseteq A$ be a coalition. Coalition structure CS is a partition of A into disjoint and exhaustive coalitions. To be more precise, $CS = \{C_1, C_2, \ldots\}$ satisfies the following conditions: $\forall i, j\ (i \neq j),\ C_i \cap C_j = \emptyset, \bigcup_{C_i \in CS} C_i = A$. A PFG generates a positive/negative value $w(C, CS)$.

The value of coalition structure CS, denoted as $W(CS)$, is given by: $W(CS) = \sum_{C_i \in CS} w(C_i, CS)$. Optimal coalition structure CS^* satisfies the following condition: $\forall CS, W(CS^*) \geq W(CS)$.

[1]https://www.dropbox.com/sh/ckg88smxuglgyd1/AADie7Xy6cDE70IbLMCzq5CZa?dl=0w

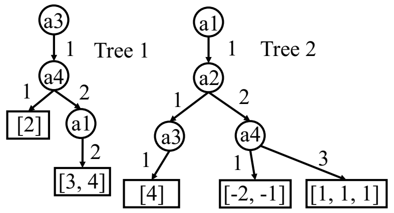

Figure 1: Partition Decision Trees

Partition Decision Trees

We utilize a graphical representation of PFGs called partition decision trees (PDTs) (Skibski et al. 2015). Intuitively, multiple rules are graphically represented by using rooted directed trees. Each rule consists of a condition for the partitioning agents and their positive/negative values. We separate a set of rules into several groups and concisely represent a group of rules using a tree. PDTs are based on rooted directed trees, where non-leaf nodes are labeled with agents' identities, leaf nodes are labeled with payoff vectors, and the edges indicate the membership of the agents in the coalitions. Each path indicates a single rule. For example, we assume that there exit 4 agents. Consider the following 2 rules. The first rule is that when agents a_3 and a_4 belong to an identical coalition, 2 is added to the value of the coalition that includes a_3 and a_4. The second rule is that when a_3 and a_4 are in different coalitions, but a_4 and a_1 are in an identical coalition, 3 is added to the value of the coalition that includes a_3 and 4 is added to the value of the coalition including a_4 and a_1. Fig. 1 shows an example of PDTs. Tree 1 represents the rules shown in this example. The value of coalition structure $W(CS)$ is calculated as the sum of the values of the compatible rules to CS. When CS is $\{\{a_1, a_2, a_3, a_4\}\}$, $W(CS)$ becomes $2 + 4 = 6$ by applying the rules indicated by the right-hand path in Tree 1 and the left-hand path in Tree 2.

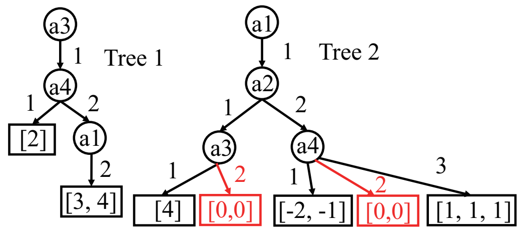

Figure 2: Modified Partition Decision Trees

CSG Algorithm

In this section, we propose new CSG algorithms utilizing PTDs. As shown in Fig. 1, some rules could have negative values in PDTs. For an optimization/maximization problem, we must carefully consider how to deal with negative rules. If there exists no negative rule, we can straightforwardly solve a CSG problem.

Naive algorithm for positive rules

When all rules are positive, we look for as many compatible rules as possible. Thus, a CSG problem is a dilemma to find a set of compatible rules that maximizes the sum of the values by selecting at most a rule from each tree. Here, the obtained result does not imply a concrete coalition structure, since some agents might not appear in the set of rules. Since adding non-member agents does not affect the coalition's values, we randomly add them to the coalitions to generate the optimal coalition structure.

Difficulty for negative rules

However, if there exist a negative rule in PDTs, finding an optimal coalition structure is not straightforward. When a PDT has a negative rule, we need to examine whether the negative rule is incompatible with the other obtained rules. As a naive method, we first find an optimal set of rules that maximize the sum of the values by ignoring the existence of negative rules. If there exists a tree with negative rules in which a positive rule is not selected in an optimal set of rules, we examine whether the obtained rules are incompatible with every negative rule in the tree. Thus, we examine whether the set of obtained rules is compatible with all possible negations of the negative rules. This operation is very complicated, since the computational complexity depends on the number of agents included in the negative rules.

Algorithm using Modified PDTs

To solve the above difficulty, we modify the description of the PDTs and explicitly represent the rules with a value of 0 when a tree has a negative rule. If the original PDT is concisely represented, the negations of the negative rules are also concisely represented. Fig. 2 shows our modified PDTs. In tree 2, there exist negative rules and thus we add two rules with the values of 0.

Figure 3 shows a search tree generated by the modified PDTs shown in Fig. 2. Since Tree 1 has only positive rules, we add an empty node that indicates a set of rules with a value of 0. In a search tree, a node indicates a rule. We sort the nodes by decreasing the node value. For this search tree, we apply a depth-first branch-and-bound algorithm. Whenever we go down to a child node, we examine the compatibility between the obtained nodes on a path and the child node. If they are compatible, we expand the grandchild node. Otherwise, we move to a sibling node next to the child node. If there exits no sibling node, we expand the sibling node next to the node. If we reach the lead node, the rules on the path imply a condition satisfied by the optimal coalition structure.

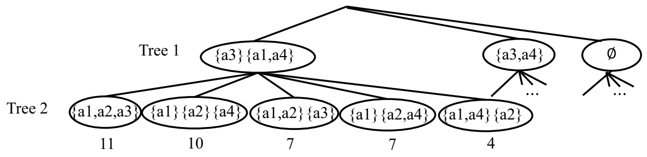

Figure 3: Search Tree

Experimental Results

We evaluate our new algorithms for 100 agents. Table 1 shows the average computational times (ms) for 100 instances. The algorithm utilizing modified PDTs is faster than the naive algorithm. We set q to the probability that a negative rule will occur.

Table 1: Average Computational Time

q	0	0.1	0.2
Naive	3.7×10^3	over 1 hour	
MPDT	2.6×10^2	1.6×10^4	6.0×10^3

Conclusion and Future Work

We developed a new CSG algorithm for partition function games utilizing a concise graphical representation. Existing works did not consider the existence of embedded coalitions with negative values. Our future work includes comprehensive computational experiments with various problem settings.

Acknowledgement

This research was partially supported by JSPS KAKENHI Grant Numbers 24220003, 15H02751, and 15K12101.

References

Rahwan, T.; Michalak, T. P.; Wooldridge, M.; and Jennings, N. R. 2015. Coalition structure generation: A survey. *Artificial Intelligence* 229:139–174.

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.

Skibski, O.; Michalak, T. P.; Sakurai, Y.; Wooldridge, M.; and Yokoo, M. 2015. A graphical representation for games in partition function form. In *The 29th AAAI Conference on Artificial Intelligence (AAAI2015)*.