An Imperfect Algorithm for Coalition Structure Generation

Narayan Changder, Samir Aknine, Animesh Dutta
Department of Computer Science and Engineering, NIT Durgapur, India, narayan.changder@gmail.com
LIRIS laboratory Claude Bernard University, Lyon 1, France, samir.aknine@univ-lyon1.fr
Department of Computer Science and Engineering, NIT Durgapur, India,animeshnit@gmail.com

Abstract

Optimal Coalition Structure Generation (CSG) is a significant research problem that remains difficult to solve. Given n agents, the ODP-IP algorithm (Michalak et al. 2016) achieves the current lowest worst-case time complexity of $O(3^n)$. We devise an Imperfect Dynamic Programming (ImDP) algorithm for CSG with runtime $O(n2^n)$. *Imperfect algorithm* means that there are some contrived inputs for which the algorithm fails to give the optimal result. Experimental results confirmed that ImDP algorithm performance is better for several data distribution, and for some it improves dramatically ODP-IP. For example, given 27 agents, with ImDP for agent-based uniform distribution time gain is 91% (i.e. 49 minutes).

The optimal CSG problem formulation

Coalition formation can be applied to many real-world problems such as task allocation, airport slot allocation, and social network analysis. Approaches to solve the CSG problem range from mixed-integer programming to branch and bound techniques (Michalak et al. 2016) through dynamic programming (Yun Yeh 1986). ODP-IP (Michalak et al. 2016) algorithm is the fastest exact algorithm for the CSG to date in practice. In this paper, we propose a new imperfect algorithm called ImDP. Imperfect algorithms can be useful if they do not fail too often. Given a set of n agents $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, a coalition C_i is a non-empty subset of $\mathcal{A}$. A coalition structure ($\mathcal{CS}$) over $\mathcal{A}$ is a partitioning of $\mathcal{A}$ into a set of disjoint coalitions $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$, where $k \in \{1, \ldots, n\}$ is called size of the coalition structure i.e. $k = |\mathcal{CS}|$. In other words, $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k\}$ satisfies the following constraints:1) $\mathcal{C}_i$, $\mathcal{C}_j \neq \emptyset$, $i, j \in \{1, 2, \ldots, k\}$. 2) $\mathcal{C}_i \cap \mathcal{C}_j = \emptyset$, for all $i \neq j$. and 3) $\bigcup_{i=1}^{k} \mathcal{C}_i = \mathcal{A}$. The value of any coalition structure $\mathcal{CS}$ is defined by $v(\mathcal{CS}) = \sum_{\mathcal{C}_i \in \mathcal{CS}} v(\mathcal{C}_i)$. The optimal solution of CSG is an optimal coalition structure $\mathcal{CS}^* \in \Pi^{\mathcal{A}}$. The set of all coalition structures over A is denoted as $\Pi^{\mathcal{A}}$. Thus, $\mathcal{CS}^* = \arg\max_{\mathcal{CS} \in \Pi^{\mathcal{A}}} v(\mathcal{CS})$.

Imperfect algorithm

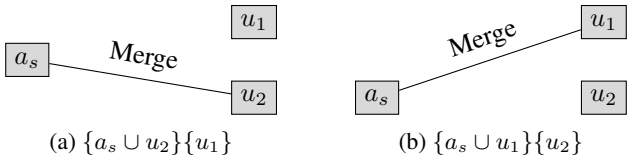
The ImDP algorithm we propose produces two tables, the partition table P_t and the optimal value table V_t. $P_t(\mathcal{C})$ stores one optimal partition of each coalition $\mathcal{C}$. There can be more than one optimal partition of a coalition $\mathcal{C}$, $P_t(\mathcal{C})$ stores any one of them. $V_t(\mathcal{C})$ stores the optimal value of a coalition $\mathcal{C}$. Let $\mathcal{C}'' = \left\{\mathcal{C}' | \mathcal{C}' \subset \mathcal{C} \text{ and } 0 \leq |\mathcal{C}'| \leq \frac{|\mathcal{C}|}{2}\right\}$, table V_t for each coalition $\mathcal{C}$ is constructed as follows:

$$V_t(\mathcal{C}) = \begin{cases} v(\mathcal{C}) & \text{if } |\mathcal{C}| = 1 \\ \arg\max_{\mathcal{C}' \in \mathcal{C}''} \{V_t(\mathcal{C}') + V_t(\mathcal{C} \setminus \mathcal{C}')\} & \text{otherwise} \end{cases}$$

Size	Coalition (C)	$v(C)$	Splitting	Optimal partition P_t	Optimal value V_t
1	{1}	24	$V_t(\{1\}) = 24$	{1}	24
	{2}	35	$V_t(\{2\}) = 35$	{2}	35
	{3}	20	$V_t(\{3\}) = 20$	{3}	20
2	{1,2}	47	$v(\{1,2\}) = 47, V_t(\{1\}) + V_t(\{2\}) = 59$	{1}{2}	59
	{1,3}	43	$(\{1,3\}) = 43, V_t(\{1\}) + V_t(\{3\}) = 44$	{1}{3}	44
	{2,3}	52	$v(\{2,3\}) = 52, V_t(\{2\}) + V_t(\{3\}) = 55$	{2}{3}	55
	{1,2,3}	78	$v(\{1,2,3\}) = 78, V_t(\{1\}) + V_t(\{2,3\}) = 79$ $V_t(\{2\}) + V_t(\{1,3\}) = 79, V_t(\{3\}) + V_t(\{1,2\}) = 79$	{1}{2,3}	79

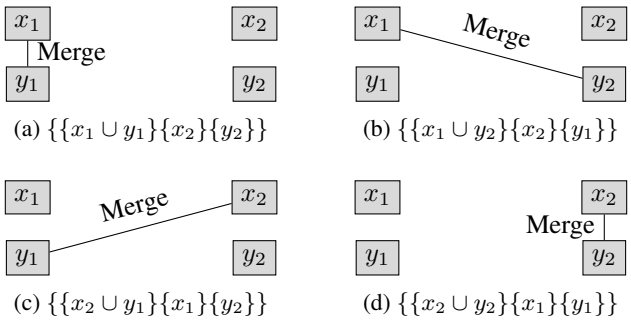
Figure 1: Working principle of ImDP algorithm computing P_t and V_t for three agents $A = \{1, 2, 3\}$.

To evaluate the coalitions, ImDP starts by evaluating all possible splits of every possible coalition of size 2, and then ImDP gradually increases the size by unit 1 till the size becomes $\lceil \frac{n}{2} \rceil$ and completes tables P_t and V_t for each evaluated coalition $\mathcal{C}$. Our main aim is to reduce the workload to solve a coalition of size x using the partial enumeration. To evaluate a coalition of size x using partial enumeration, ImDP algorithm uses two new merge functions: $Merge_1$ and $Merge_2$. To show how merge functions work, we focus on the evaluation of a single coalition. The following notation will be used to represent a coalition and its partitions. For instance, the coalition $\{1, 2, 3\}$ will be evaluated as the partition $\{\{1\}\{2, 3\}\}$ or $\{\{2\}\{1, 3\}\}$ or $\{\{3\}\{1, 2\}\}$ or $\{1, 2, 3\}$.Then ImDP stores in the P_t table the partition with the maximum value and in the V_t table this maximum value. More formally, any coalition can be stored in the partition table with any of its different possible partitions (into two halves or as the coalition itself). We will call each half a component. For example in $\{\{1\}\{2, 3\}\}$, we denote $\{1\}$ and $\{2, 3\}$ as two different components of the coalition $\{1, 2, 3\}$. In the following, we will detail each of ImDP merge func-



(a) $\{a_s \cup u_2\}\{u_1\}$ (b) $\{a_s \cup u_1\}\{u_2\}$

Figure 2: $Merge_1$ function operates on the coalition $\{a_s \cup \mathcal{C} \setminus a_s\}$. Here $Merge_1$ is applied between the coalitions a_s and $\{\mathcal{C} \setminus a_s\}$. It is assumed that the coalition $\{\mathcal{C} \setminus a_s\}$ is stored in the partition table P_t as $\{\{u_1\}\{u_2\}\}$. In the left part, $Merge_1$ is applied between the coalitions $\{a_s\}$ and $\{u_2\}$, it results with a new partition $\{\{a_s \cup u_2\}\{u_1\}\}$ of coalition $\{a_s \cup \mathcal{C} \setminus a_s\}$. In the right part, $Merge_1$ is applied between the coalitions $\{a_s\}$ and $\{u_1\}$, it results with another new partition $\{\{a_s \cup u_1\}\{u_2\}\}$ of coalition $\{a_s \cup \mathcal{C} \setminus a_s\}$.



(a) $\{\{x_1 \cup y_1\}\{x_2\}\{y_2\}\}$ (b) $\{\{x_1 \cup y_2\}\{x_2\}\{y_1\}\}$

(c) $\{\{x_2 \cup y_1\}\{x_1\}\{y_2\}\}$ (d) $\{\{x_2 \cup y_2\}\{x_1\}\{y_1\}\}$

Figure 3: $Merge_2$ function is applied between two coalitions $\mathcal{X}$ and $\mathcal{Y}$ of size $\lceil \frac{n}{2} \rceil$ and $n - \lceil \frac{n}{2} \rceil$. Each figure shows how a new coalition structure is formed by using $Merge_2$ function. In figure a) component $\{x_1\}$ of the coalition $\mathcal{X}$ is merged with component $\{y_1\}$ of the coalition $\mathcal{Y}$, resulting a coalition structure $\{\{x_1 \cup y_1\}\{x_2\}\{y_2\}\}$, and so on.

tions. We use $Merge_1$ function to evaluate each coalition of size $2, 3, \ldots, \lceil \frac{n}{2} \rceil$. For any coalition $\mathcal{C}$, $Merge_1$ picks a single agent a_s and creates all the partitions of the coalition $\mathcal{C}$ as $\{\{a_s\}\{C \setminus a_s\}\}$. For each $a_s \in \mathcal{C}$, $Merge_1$ is applied between $\{a_s\}$ and $\{C \setminus a_s\}$. The figure 2 shows the detailed operation of $Merge_1$ function. The principle of $Merge_2$ function is shown in Figure 3. $Merge_2$ function is used between two disjoint coalitions of size $\lceil \frac{n}{2} \rceil$ and $n - \lceil \frac{n}{2} \rceil$. After $Merge_2$ is applied, ImDP calculates highest valued coalition structure using the P_t table in a bottom up fashion. In our example (c.f. Figure 1), ImDP sets $\mathcal{CS}^* = P_t(\mathcal{A}) = P_t(\{1, 2, 3\})$ and finds that it is beneficial to split it into the coalitions $\{1\}$ and $\{2, 3\}$. In the same way, by looking at $P_t(\{2, 3\})$, ImDP finds that it is also beneficial to split $\{2, 3\}$ into the coalitions $\{2\}$ and $\{3\}$. Now, the optimal $\mathcal{CS}$ is $\{\{1\}\{2\}\{3\}\}$ with a value of 79.

Experimental evaluation

We empirically evaluated the ImDP algorithm and benchmarked it against ODP-IP. We compared the performances of both algorithms given different numbers of agents (5 to 27). As can be seen, ImDP is faster for the distributions shown in table 1. ImDP algorithm finds optimal coalition structure when one of the optimal partitions is actually considered by $Merge_1$ and $Merge_2$ functions. If none of these partitions is considered by ImDP, then it fails to give the optimal result. However, ImDP will still give a sub-optimal solution. As ImDP is an imperfect algorithm, we need to know if ImDP does not produce optimal $\mathcal{CS}$, then what is the difference between ODP-IP generated optimal $\mathcal{CS}$ and ImDP generated $\mathcal{CS}$. ImDP algorithm fails only for few datasets but the failure rate is very low. For example, we found that for geometric, laplace, F- distribution failure rates are respectively 1.7%, 3.34%, and 2.12%. We also found that, in the case of failure, (ImDP generated $\mathcal{CS}$ value)/(Optimal $\mathcal{CS}$ value) is always greater than .90 and most ratios are .99.

	Time measured in seconds		
Distribution	ODP-IP time (t_1)	ImDP time (t_2)	Difference $t_1 - t_2$
Agent-based uniform	3224.8075	293.0295	2931.778
Agent-based normal	2696.692	283.88	2412.812
Chi-square	1078.2885	277.5145	800.774
Geometric	441.038	238.922	202.116
NDCS	500.1605	245.4155	254.745
Raleigh	628.897	242.6665	386.2305
F-distribution	588.918	244.1565	344.7615
Laplace	396.9215	252.4695	144.452

Table 1: Time difference between ODP-IP and ImDP in seconds for 27 agents.

Computational efficiency of ImDP: ImDP evaluates all coalitions of size $1, \ldots \lceil \frac{n}{2} \rceil$, i.e. it performs the following steps. $\binom{n}{1} * 1 + \binom{n}{2} * 2, + \ldots, + \binom{n}{\lceil n/2 \rceil} * (\lceil n/2 \rceil) = \sum_{k=1}^{(\lceil \frac{n}{2} \rceil)} \binom{n}{k} * k$. Using the identity $\binom{n}{k} = \frac{n}{k}\binom{n-1}{k-1}$ (assume n is even), we get, $\sum_{k=1}^{\frac{n}{2}} \binom{n}{k} * k = \sum_{k=1}^{\frac{n}{2}} k\frac{n}{k}\binom{n-1}{k-1} = n\sum_{j=0}^{\frac{n}{2}-1} \binom{n-1}{j}$. The bound for above series is $O(n2^n)$. So, the total time complexity is $O(n2^n) + O(2^n) + O(2^n) = O(n2^n)$. As a conclusion, we can notice that the results already obtained for ImDP will be very beneficial for future work since the improvement already obtained on ODP-IP is of the order of 91% for some distributions.

Acknowledgments

The research presented in this article is funded by "Visvesvaraya PhD Scheme for Electronics & IT", grant no: PhD-MLA/4(29)/2015-16. Samir Aknine was supported by Univ. Lyon 1.

References

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.

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