A Kernelization approach for Anytime Coalition Structure Generation using Knuth Algorithm X

Narayan Changder[1], Animesh Dutta[1], and Aditya K. Ghose[2]

[1] National Institute of Technology Durgapur, West Bengal, India
narayan.changder@gmail.com, animeshnit@gmail.com
[2] University of Wollongong, Wollongong NSW 2522 Australia
aditya@uow.edu.au

Abstract. *Determining the optimal coalition structure is an interesting problem in multi-agent system and remains difficult to solve. It is proved that this problem is* **NP**- *complete and even finding a suboptimal solution needs exponential time. The main problem is, after a group of agents comes together, how the agents can be partitioned such that social payoff is maximized? This work presents the application of Algorithm X in coalition structure generation problem. Before the algorithm starts, it uses some kernelization techniques to make the original input* $(2^n - 1)$ *size into some smaller input instances.*

Keywords: Multi agent system, Optimization, Coalition formation, Kernelization, **NP**-Complete

1 Introduction

The optimal coalition structure generation is an interesting research problem in Multi-Agent Systems (MAS). This problem is interesting to MAS community due to its important applications and its computational challenges. The problem is challenging because of exponential growth of coalition structures when number of agents grows linearly. It is proved that optimal coalition structure generation problem is **NP**- complete [10]. This problem directly matched with the famous mathematical problem: Complete set partitioning problem.

Coalition formation is a process by which several agents agree together to perform some task. Coalition structure formation is a mechanism by which agents are grouped into different disjoint coalitions. Each coalition is assigned a value by a characteristic function. Such a setting is known as characteristic function games (CSG). In CSG the value of a coalition in any coalition structure is not dependent on other coalition in coalition structure. Formally, coalition structure generation is defined as follows:

Definition 1 *Given a set of agents* $A = \{a_1, a_2, \ldots, a_n\}$*, a* Coalition Structure (CS) *over* A *is a partitioning of the agents into different coalitions* $\{C_1, C_2, \ldots, C_k\}$*, where* k *is called size of coalition structure i.e* $k = |CS|$*. Such that it satisfies the following constraints:*

1. $C_j \neq \emptyset$, $j = \{1, 2, \ldots, k\}$

2. $C_i \cap C_j = \emptyset$ *for all* $i \neq j$ *and*
3. $\bigcup_{i=1}^{k} C_i = A$

For example, in a multi-agent system consisting of three agents $A = \{a_1, a_2, a_3\}$, we have total seven possible coalitions:

$$\{\{a_1\}, \{a_2\}, \{a_3\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}\}$$

The set of all coalitions structures over A is denoted as Π^A

$$\Pi^A = \{\{a_1\}, \{a_2, a_3\}\}, \{\{a_3\}, \{a_1, a_2\}\}, \{\{a_2\}, \{a_1, a_3\}\}\{\{a_1\}, \{a_2\}, \{a_3\}\}, \{\{a_1, a_2, a_3\}\}$$

Now it is observed that the optimal coalition structure and complete partition of a set of agents are same. We are now ready to state our optimization problem formally.

Definition 2 *The value of any coalition structure* CS *is defined by*

$$V(CS) = \sum_{C_i \in CS} (v(C_i))$$

Generally, the goal of the coalition structure formation problem is to find the coalition structure which maximizes social welfare by finding an optimal coalition structure $CS^* \in \Pi^A$.

$$CS^* = arg\ max_{CS \in \Pi^A} V(CS)$$

Motivated by the observations in [8], we developed our algorithm .

2 Kernelization algorithm

Kernelization (preprocessing or data reduction) is used to solve NP-hard problem. The practitioner use kernelization method to improve the efficiency of algorithm to solve hard problems. Here, improvement is obtained by using preprocessing stage on input data and after processing the original inputs are replaced by smaller inputs. The basic idea of kernelization method is to reduce the given problem instance into an equivalent smaller sized problem instance. The reduction from original problem instance into the reduced problem instance must be done in time polynomial in the input size.

2.1 Preprocessing stage

In preprocessing step we filter all coalition that can not be a candidate for optimal coalition structure. For example, the coalition value for agent $v(a_1) = 4$, $v(a_2) = 6$ and $v(a_1, a_2) = 9$. The coalition $\{a_1, a_2\}$ can not be a candidate in any optimal coalition structure because

$$v(a_1) + v(a_2) \geq v(a_1, a_2)$$

a_1 and a_2 would not work together because if they works seperately they can earn more profit. We can state the above result by using the following lemma.

Lemma 1 *Given n agent and all nonempty coalitions of n agents with a value assigned to each coalition. A coalition $C = \{a_1, \ldots, a_k\}$, where $|C| = k$ can not be part of optimal coalition structure CS^* if $v(a_1) + v(a_2) + , \ldots, + v(a_k) \geq v(C)$*

In simplest term, if the agents can earn more profit by working alone then they will not create a coalition.

To reduce the number of inputs we can estimate which size coalitions may be a part of optimal coalition structure. By using the integer partition approach [8] it calculates the set of all promising size subsets. Here algorithm do not search any subspaces. It just calculates upper bound and lower bound of each subspace and tries to prune not-promising subspaces by checking the upper bound of each subspace with the highest lower bound. For example, with 5 agents, suppose, we get all promising subspaces as follows

$$\{1, 2, 2\}, \{1, 4\}, \{1, 1, 1, 1, 1\}$$

Which means that optimal coalition structure can be anywhere in above subspaces and it is sure that only the coalition with size 1,2 and 4 will play a role in optimal coalition structure generation. Now, we pruned all the coalition with size 3 and 5 from our input. To check all individual coalition with size 1,2 and 4, take each of the coalition of size 1,2 and 4 and calculates whether the sum of individual agents value is greater than the coalition value. For example a coalition $\{a_1, a_2, a_3, a_5\}$. If $v(a_1) + v(a_2) + v(a_3) + v(a_5) \geq v(a_1, a_2, a_3, a_5)$, then (a_1, a_2, a_3, a_5) can not be a feasible coalition in optimal coalition structure.

3 Algorithm X

The problem of finding a subfamily of disjoint sets with the same union as the given whole set is solved by Donald Knuth by using a backtracking approach. Knuth gives this algorithm name Algorithm X [3]. The Algorithm X is used to solve the exact cover problem. Formally exact cover problem is defined as follows

Definition 3 *Given a ground set U and collection of subset $F \subseteq U$. What is the possible way to arrange the disjoint subsets such that it covers the ground set U.*

Algorithm 1 shows the different steps. The algorithm works on a matrix H where each row represent the subset and each column represents the subsets in which it is present. For example, given the following subsets

1. $A = \{1, 4, 7\}$;
2. $B = \{1, 4\}$;
3. $C = \{4, 5, 7\}$;
4. $D = \{3, 5, 6\}$;
5. $E = \{2, 3, 6, 7\}$
6. $F = \{2, 7\}$.

Where ground set is $U = \{1, 2, 3, 4, 5, 6, 7\}$. The matrix H for this problem is

$$H = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

Algorithm X finds all solution to exact cover problem by recursively searching the matrix H. Algorithm X [3] works as follows

Algorithm 1 Algorithm X

Input: Set of all feasible coalitions that can be part of optimal coalition structure
Output: Good coalition structure $CS^*(n)$

if the matrix H is empty **then**
    the problem is solved; terminate successfully.
else choose a column c (deterministically)
    Choose a row r such that $H[r, c] = 1$ (nondeterministically).
    Include row r in the partial solution.
    for each column j such that $H[r, j] = 1$ **do**
        delete column j from matrix H;
        for each row i such that $H[i, j] = 1$ **do**
            delete row i from matrix H.
        end for
    end for
end if
Repeat this algorithm recursively on the reduced matrix H.
for each coalition structure (exact cover) returned by Algorithm X **do**
    calculate its value and compare with previous one.
end for
return best coalition structure found so far

Theorem 1 *If k is the number of coalition in the input, in the worst case Knuth Algorithm X will takes time $O(4^{\frac{k}{5}}) \approx O(1.31951^k)$*

This algorithm's performance is heavily depends on kernelization algorithm. If it can prune more coalition from the original input, Algorithm X will be more faster. For each of the coalition structure (exact cover) generated by Algorithm X , we calculate the total value by adding the individual coalition value in coalition structure.

4 Related Work

The current research work on coalition structure formation can be classified into three categories [4].

1. **Anytime algorithms—** It permits premature termination (i.e., before the optimal solution has been found) but at the same time it provides guarantees on the quality of the solution. One of the disadvantage of anytime algorithm in coalition formation mechanism is that they all requires, in the worst case to check all coalition structures. Hence, time required is $O(n^n)$.
2. **Design-to-time algorithms—** This type of algorithm guarantees to return an optimal solution but to do so, it must run on completion. That is they can not produce intermediate result like anytime algorithms. Currently the only design-to-time algorithm to solve coalition structure formation problem is dynamic programming with time complexity $O(3^n)$.
3. **Heuristics algorithms—**This type of algorithm sacrifices quality guarantees of solution for speed. The main drawback of this type algorithm is that it is impossible to verify the quality of generated solution.

The optimal coalition structure can be generated by using dynamic programming algorithm [12] in $O(3^n)$ time, however it is impractical for moderate size of inputs.

The state-of-the-art *Improved Dynamic Programming* (IDP) [5] algorithm is improved version of [12] it is improved by Rahwan and Jennings [5] to enhance the usability of dynamic programming. Their approach is not to evaluate some unnecessary splitting. The author shows that their approach is empirically faster and uses less memory, but still worst case runtime is $O(3^n)$. The main drawback of this approach is it does not have anytime properties.

Due to the high complexity of dynamic programming many researchers are developing anytime algorithm which allow quality suboptimal solution very faster. Many anytime coalition structure formation algorithm operates on the space of all coalition structures, between $\omega(n^n)$ to $O(n^n)$ [10]. Sandholm et al.[10] proposes first anytime algorithm for coalition structure formation problem. They proved that to provide a bound on the quality of the solution algorithm needs to check at-least 2^{n-1} coalition structures. Their approach uses coalition structure graph, where nodes in the graph are coalition structures and edges between nodes are splitting(or mergers) of coalition structure.

On the other hand the algorithm proposed by Dang and Jenning [2] also used the coalition structure graph representation as Sandholm et al.[10], they define and search a particular graph subsets and shown that their technique empirically generate tighter quality guarantee's than Sandholm et al.[10]. Moreover, several researchers developed anytime algorithms that searches the space of coalition structure graph in different ways [4,9,2].

Rahwan and Jennings [9] have not used coalition structure graph. They use a totally new representation of coalition structures space based on integer partitions and shown that their approach empirically gives high quality solutions than any other previous anytime algorithms. Their approach is called anytime IP algorithm.

The algorithm in [4] , called IDP-IP, it combines postive sides of IDP and anytime IP algorithm and avoids their weakness.

Service and Adams [11] proposed an anytime dynamic programming (ADP) to solve coalition structure formation problem. Their approach is to first create a greedy solution in $O(2^n)$ time by greedily adding the coalition of largest value, out of unassigned agents. Next, dynamic programming is used to solve the problem with sizes

$1, 2, \ldots k$. If run to completion, it gives optimal result. Otherwise, it will produce better of greedy solution and current iterative solution.

In this work , we raised the question of how to achieve high quality solutions to the coalition structure formation problem (see e.g. [7] for a survey of the problem), especially when the search execution times to find solutions are very limited.

5 Future work

Our algorithm is anytime (after preprocessing stage complete). In future we will compare our algorithm with [8,6] by using different data distributions . To the best of our knowledge, we believe that [8,6] the fastest exact anytime algorithm for coalition structure generation in the literature. The Algorithm X can unnecessary search the non feasible subspaces and takes more time. In our previous example we found three feasible subspaces but Algorithm X will search all the extra subspaces that can be formed by 1,2 and 4. There are three feasible subspace $\{1, 2, 2\}, \{1, 4\}, \{1, 1, 1, 1, 1\}$ but it will search the extra subspace $\{2, 1, 1, 1\}$. We have to modify the Algorithm X such that it will only search the promising subspaces.

6 Acknowledgments

This research work is funded by Visvesvaraya PhD scheme of DeitY (Department of Electronics & Information Technology), Govt. of India.

References

1. M. S. Boddy. Anytime problem solving using dynamic programming. In *AAAI*, pages 738–743, 1991.
2. V. D. Dang and N. R. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 564–571. IEEE Computer Society, 2004.
3. D. E. Knuth. Dancing links. *arXiv preprint cs/0011047*, 2000.
4. T. Rahwan and N. R. Jennings. Coalition structure generation: Dynamic programming meets anytime optimization. In *AAAI*, volume 8, pages 156–161, 2008.
5. T. Rahwan and N. R. Jennings. An improved dynamic programming algorithm for coalition structure generation. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1417–1420. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
6. T. Rahwan, T. Michalak, and N. R. Jennings. A hybrid algorithm for coalition structure generation. 2012.
7. T. Rahwan, T. P. Michalak, M. Wooldridge, and N. R. Jennings. Coalition structure generation: A survey. *Artificial Intelligence*, 229:139–174, 2015.
8. T. Rahwan, S. D. Ramchurn, V. D. Dang, A. Giovannucci, and N. R. Jennings. Anytime optimal coalition structure generation. In *AAAI*, volume 7, pages 1184–1190, 2007.
9. T. Rahwan, S. D. Ramchurn, N. R. Jennings, and A. Giovannucci. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, pages 521–567, 2009.

10. T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1):209–238, 1999.
11. T. C. Service and J. A. Adams. Anytime dynamic programming for coalition structure generation. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1411–1412. International Foundation for Autonomous Agents and Multiagent Systems, 2010.
12. D. Yun Yeh. A dynamic programming approach to the complete set partitioning problem. *BIT Numerical Mathematics*, 26(4):467–474, 1986.