A Prune-Based Algorithm for Computing Optimal Coalition Structures in Linear Production Domains

Chattrakul Sombattheera
Decision Systems Lab
School of IT and Computer Sience
University of Wollongong, NSW 2500
Australia
cs50@uow.edu.au

Aditya Ghose
Decision Systems Lab
School of IT and Computer Sience
University of Wollongong, NSW 2500
Australia
aditya@uow.edu.au

ABSTRACT

Computing optimal coalition structures is an important research problem in multi-agent systems. It has rich application in real world problems, including logistics and supply chains. We study computing optimal coalition structures in linear production domains. The common goal of the agents is to maximize the system's profit. Agents perform two steps: *i*) deliberate profitable coalitions, and *ii*) exchange computed coalitions and compute optimal coalition structures. In our previous studies, agents keep growing their coalitions from the singleton ones in the deliberation step. This work takes opposite approach that agents keep pruning unlikely profitable coalitions from the grand coalition. It also relaxes the strict condition of coalition center, which yields the minimal cost to the coalition, that agents merely keep generating profitable coalitions. Furthermore, we discuss relevant concept, i.e., integer partition, that draw into concept in our algorithm and provide an example of how it can work. Lastly, we show that our algorithm outperforms exhaustive search in generating optimal coalition structures in terms of elapses time and number of coalition structures generated.

1. INTRODUCTION

Coalition formation is an important area of research in multi-agent systems. It studies the process and criteria that lead to cooperation among agents. The process involves two main inter-related activities: *i*) negotiation in order to exchange information among agents, and *ii*) deliberation in order to decide with which agents should they cooperate. Coalition formation research has its roots in the theory of cooperative game [3, 4] in which a characteristic function assigns each coalition a *coalition value*. A coalition value is often economical value, such as money, that is assumingly created jointly by the coalition. Such a value will be distributed as *payoffs* among coalition members. The focus of the study is on *i*) what the agents' payoff should be, that leads to *ii*) what coalitions would form. Agents in such a setting are self-interested: they try to form coalition when they foresee an opportunity to increase their payoffs. Most of the studies in the theory of cooperative game operate in superadditive environment in which merging of two coalitions yields a new coalition value of at least as equal to the sum of the two coalition values.

However, the assumption of characteristic function is somewhat impragmatic that it leads to the ignorance of the process of forming coalitions. Also, the assumption of superadditive environment is not always true in various real world settings, taking into account multiple factors, including the c ost of coalition. Coalition formation research in multi-agent systems [6, 7, 9, 10, 8] leaves such assumptions but takes into account reality. This usually involves various factors and a large number of agents. Thus coalition formation becomes a very complex process. There is a large number of messages to be sent across while negotiating and there is a large number of coalitions to be considered while deliberating. A strategy to reduce such complication in negotiation is that agents focus on deliberation: generate a list of potential coalitions, yet to be agreed upon by agents, that are likely to be formed [8]. Most of coalition formation studies in multi-agent systems involves self-interested agents and are highly successful [6, 7, 9, 10, 8].

Coalition formation among fully-cooperative agents is also an important, yet to receive more attention, area of research in multi-agent systems. The common goal of agents is to maximize the system's utility–agents are to form coalitions such that the sum of the coalition values is maximal. This problem is known as *finding optimal coalition structure* (see section 2.2). It has rich application in real world settings, including logistics and supply chains, grid computing systems, and composite web services. These settings usually involve a large number of agents that makes the problem intractable for even a small number of agents (see section 2.2). A handful of previous studies assume the existence of characteristic function [6, 1]. Although they have achieved high performance algorithms, they still rely heavily on the existence of characteristic function. This makes the algorithms impragmatic as mentioned above. For a system of m agents, generating all coalition values (due to the non-existence of characteristic function) of m agents is exponentially complex, i.e., 2^m, and can also be intractable for even a reasonably small number of m—let alone the problem of finding optimal coalition structures.

To our knowledge, our previous work [11, 12] is the only attempt to tackle the problem of finding optimal coalition structure with realistic assumption, i.e., the non-existence of characteristic function and non-superadditive environment. They propose a deliberation algorithm that helps reduce the number of coalitions generated. Each agent generates profitable coalitions. From it's singleton coalition, it keeps adding profitable members based on existing resources of the coalition. The coalition grows until it cannot produce profit anymore. This work is different in various ways. Firstly, it takes the opposite approach: each agent keeps pruning least profitable agents from it's grand coalition. Secondly, it relaxes the strict condition of *coalition center* [11] that agents merely keep generating profitable coalitions. Thirdly, we introduce pattern into coalition structure generation. Lastly, we propose a concrete algorithm in for generating coalition structures. We also provide an example of how it can work.

The outline of this paper is as follows. We restate the problem domains and discuss about related issues in optimal coalition structure. We then discuss the deliberation, coalition structure generation and example. Then we discuss about the experiment, show empirical results. Lastly, we discuss related work which followed by conclusion and future work.

2. COALITION FRAMEWORK

2.1 Coalition in Linear Production Domains

We remodel Owen's work [5] as in our previous work. For the sake of completeness, we restate our model below. Let $A = \{a_1, a_2, \dots, a_m\}$ be a set of agents, whose goals are to maximize the system's profit. Let $R = \{r_1, r_2 \dots, r_n\}$ be the set of resources and $G = \{g_1, g_2, \dots, g_o\}$ be a set of goods. Resources themselves are not valuable but they can be used to produce goods. The *linear technology matrix* $L = [\alpha_{ij}]_{n\times o}$, where $\alpha_{ij} \in \mathbb{Z}^+$, specifies the units of each resource $r_i \in R$ required to produce a unit of the good $g_j \in G$. The goods can be sold to generate revenue for the system. The price of each unit of goods produced is specified by the vector $P = [p_j]_{1\times o}$. Each agent $a_k \in A$ is given a resource bundle $b^k = [b^k_i]_{n\times 1}$. In this setting, agents try to cooperate, i.e. form coalitions, in order to pool their resources, thus increase revenue for the system. A coalition $S \subseteq A$ has a total of $b^S_i = \sum_{k\in S} b^k_i$ of the i^{th} resource. Each coalition S can use their resources to produce any vector $x = \langle x_1, x_2, \dots, x_o\rangle$ of goods that satisfies the constraint:

$$\sum \alpha_{ij} x_i \leq b^S_i$$

and

$$x_j \geq 0$$

The cooperation cost among agents is specified by the matrix $C = [c_{kl}]_{m\times m}$, which assigns a cooperation cost between each pair (a_k, a_l) of agents such that $c_{kl} \in \mathbb{Z}^+$if $k \neq l, \in \{0\}$ otherwise. Agents in the coalition S have to find a vector x to maximize the revenue accruing to a coalition. Let

$$P_S = \sum_{l=1}^{o} p_l x_l.$$

be the maximal revenue the coalition can generate. Here, we introduce *virtual coalition center*. Each agent $a_k \in S$ can assume itself a coalition center and computes the virtual coalition cost.

$$C^k_S = \sum_{l\in S} c_{kl}.$$

The virtual coalition value υ^k_S computed by a_k is

$$\upsilon^k_S = P_S - C_S.$$

Each agent then can exchange the virtual coalition value. The maximal virtual coalition value is, of course, the coalition value, υ_S. Any agent a_k who yields the maximal virtual coalition value can be a coalition center.

2.2 Optimal Coalition Structures

Once all agents agree to form coalitions, they can be viewed as a set has been partitioned into mutually disjoint.proper subsets. Each subset is a coalition, $S \in A$. Since a coalition is merely a subset, we shall use the term *cardinality* to refer to the size of the coalition. The set of all agents itself is called the *grand coalition*. Each instance such a partition is known as a *coalition structure*, CS. In our setting, the coalition value is independent of the actions of other agents outside the coalition. The *value* of each coalition structure

$$V(CS) = \sum_{S\in CS} \upsilon_S$$

indicates the system' utility yielded by that partitioning. Let L be the set of all coalition structures. The goal of cooperative agents in coalition formation [6, 2] is to maximize the system's utility. That is agents are to find at least a coalition structure CS^* such that

$$CS^* = argmax_{CS\in L} V(CS)$$

In the literature, coalition structures are laid down into m layers. Each layer L_κ, where $1 \leq \kappa \leq m$, is composed if coalition structures, whose number of coalition are equal to κ. We shall call the of number of coalitions within each coalition structure the *size* of the coalition structure. The number of coalition structures within each layer L_κ is known as the *Stirling number of the Second Kind*:

$$S(m,\kappa) = \frac{1}{\kappa!}\sum_{\iota}^{\kappa-1}(-1)^\iota \binom{\kappa}{\iota}(\kappa-\iota)^m$$

Hence, the number of all coalition structure is

$$|L| = \sum_{\kappa=1}^{m} S(m,\kappa)$$

Computing the optimal coalition structures in a non-superadditive environment is non-trivial [6]. Sandholm et. al. show that it is NP-hard [6]. Due to the large search space, existing algorithms can *generate* coalition structures which are within a certain bound from optimal and will get closer as the algorithms proceed. This work assumes non-superadditive environment and non-existence of characteristic function. Each coalition value is not known a priori. Thus agents have to compute all coalition values first. Given a set of m agents, there are 2^m possible subsets, hence the complexity of computing all coalition structures is substantially worse.

2.3 Best Coalition and Coalition Structure Pattern

In previous studies [6, 2], coalition structures are generated based on the size of coalition structures and the cardinality of the coalitions. It appears that the search space is very large. Here, we try to reduce the search space. For each cardinality, each agent tries to do local search for a small number of coalitions. Firstly, we define the agent a_k's *best coalition* for the cardinality κ the coalition S_k^κ, whose members include a_k, that is found from a search within a given time and yields the maximal v_S. Within the same cardinality, the next coalition that yields the second highest coalition value is *second* best coalition, and so on.

We introduce the *pattern* of generating coalition structures. A pattern of a coalition structure describes the number of coalitions and their cardinalities in the coalition structure. It is written in the form

$$B_1 + B_2 + \ldots + B_\kappa \text{where} \sum_{\iota=1}^{\kappa} B_\iota$$

This work proposes coalition structure pattern in breaking manner as the followings. Given a set of 6 agents, the first pattern is 6 in layer L_1. There can be just one coalition, which is the grand coalition, whose cardinality is 1. In the next layer, L_2, the grand coalition will be broken into 2 coalitions by splitting a member from the grand coalition into the new coalition. Hence the pattern is 5 + 1. The next pattern is 4+2 and 3+3. The pattern in each layer cannot grow once the difference between each pair of coalition's cardinality is ≤ 1. Then the pattern grows into the next layer, i.e., 4 + 1 + 1, 3 + 2 + 1, 2 + 2 + 2. The last pattern is obviously 1 + 1 + 1 + 1 + 1 + 1. The full patterns for 6 agents are shown below:

No. of coalitions	1	2	3	4
Patterns	6	5 + 1	4 + 1 + 1	3+1+1+1
		4 + 2	3 + 2 + 1	2+2+1+1
		3 + 3	2 + 2 + 2	

No. of coalitions	5	6
Patterns	2+1+1+1+1	1+1+1+1+1+1

Agents can use best coalitions to generate coalition structures by following these patterns. By using the best coalitions alone, agents will achieve some coalition structures whose best one will be close to the optimal one. Using more coalitions, i.e., the second, third best and so on, coalition structure values can be improved.

3. ALGORITHM FOR GENERATING COALITION STRUCTURE

Each agent has to do two steps of deliberation: *i*) Pruning: deliberate over what coalitions it might form by deleting unprofitable coalition members from the grand coalition, and *ii*) Generating: exchange coalitions generated and use the breaking pattern to generate coalition structures. The sets of such coalitions are at least close to the optimal coalition structures. The main goal of the algorithm is to reduce search space for finding the optimal coalition structures. This can be achieved by reducing the number of coalitions to be considered.

3.1 Deliberating Process

We take opposite approach our previous algorithms [11, 12] for agents' deliberation. We explain the ranking trees that are used as infrastructure in the early stage of the deliberation. Each agent ranks other agents based on their suitability to be coalition members. Then we will explain the extended part where each agent tries to shrink its coalitions.

In the following, we will identify a coalition by the identifier of agent a_k. Thus the coalition S^k refers to a coalition being consider by agent a_k. Hence b^S represents the resource vector of S^k. Given a coalition S^k, let G^k refer to the set of goods whose resource requirements are fully or partially satisfied by b^S, the resources available in S^k (excluding goods whose resource requirement might be trivially satisfied because these are 0). For each good $g_j \in G^k$, the coalition center agent a_k ranks agents not currently in its coalition on a per good basis. For each resource r_i of good g_j, agent a_k ranks non-member agents by computing for each $a_l \notin S^k$, whose $b_i^l > 0$, the value π_i^j—its proportional contribution to the profit of the good (using its fraction of the resource requirements for that good provided by the a_l) minus the (pair-wise) collaboration cost between a_l and a_k, i.e.,

$$\pi_i^j = \frac{b_i^l}{\alpha_{ij}} p_j - c_{kl}.$$

The agent a_k uses this proportional contribution π_i^j to construct a binary tree for each g_j. The only child of the root g_j is the first resource α_{1j}, whose left child is the second resource α_{2j}, and so on. For each α_{ij}, its right child is either *i*) null if $\alpha_j^i = 0$, or *ii*) the agent $a_{1st}^{r_i}$, whose pi_i^j value is the greatest. The right child of $a_{1st}^{r_i}$ is the agent $a_{2nd}^{r_i}$, whose π_i^j value is the second greatest, and so on. Agent a_k can use these tree to eliminate surplus agents.

The agent a_k uses b^S to determine surplus resources not used to produce additional units of a good g_j. For each $g_j \in G^k$ and resource r_i,

$$\beta_i^j = b_i^S - I(\alpha_{ij}),$$

where $I \in \mathbb{Z}^+$ is the smallest integer such that $\beta_i^j > 0$, represents the surplus amount of r_i that coalition S^k does not use to produce good g_j, provided the amount is non-negative ($\beta = 0$ otherwise). The *indicative vector*, $\beta^j = [\beta_i^j]_{1\times n}$, represents surplus units of each resource r_i of good g_j.

In this paper, the agent a_k creates the grand coalition and tries to shrink it by pruning least profitable members. The agent utilizes indicative vectors β^js and the the tree T^j in order to locate the agent who is the least useful to its present coalition. For each good, the positive value of β_i^j in the indicative vector indicates surplus resource that the agent who possesses the equivalent resource should be eliminated from the present coalition. The agent a_k create a trial coalition S' for each good. The surplus agents will be eliminated from S for the next smaller quantity of the good possible. Each trial coalition will be inserted into the pruning members S^-. The sub-algorithm for selecting profitable members is shown in algorithm 1.

In the main algorithm, the agent a_k considers itself a singleton coalition at the beginning of deliberating. It create the ranking tree T^G of all agent for each good. At this point it is only root of the profitable-coalition tree, L^-, and is the base of the growing coalition. It prune the pruning agents S^- from the coalition. Each $S'_j \in S^-$ will be added as the children of the base coalition. Among all S'_js, the most prof-

Algorithm 1 Select the most profitable members

```
Require: A coalition $S$
Require: ranking trees $T^G$
  set highest profit $v^* = 0$
  set pruning members $S^- = S$
  for all $g_j \in G$ do
    if $S$ is not capable of producing $g_j$ then
      continue
    end if
    get surplus agents $S'$
    set trial coalition $S'_j = S \cup S'_j$
    compute trial coalition's profit $v_{S'_j}$
    set $S^- = S^- \leftarrow S'_j$
  end for
  return $S^-$
```

itable agents S^* are those that provide the highest additional profit v^* and are kept as the base for the further growing coalition. The coalition keeps shrinking in this fashion until there are no pruning members left in T^G. Then the next profitable sibling of the base S'_j will be the new base. This repetition goes on until it cannot find the new base. The number of coalitions each agent a_k has to maintain is also much smaller compared to that of the exhaustive search. The main algorithm is shown in algorithm 2.

Algorithm 2 Main

```
  set $L^- = N$
  create ranking trees $T^G$ for all goods
  collect pruning members $S^-$
  while $S^- \neq \emptyset$ do
    locate $S^* \in S^-$
    set $A' = A' - S^*$
    set $S = S \cup S^*$
    set $L^- = L^- \cup S$
    collect pruning members $S^+$
    if $S^+ = null$ then
      set $S^*$ = the next profitable sibling of$S^*$
    end if
  end while
```

3.2 Generating Coalition Structure

Once each agent finishes its deliberation in the first stage, it exchanges all the coalitions generated with all other agents. It then uses the pattern to generate coalition structures. Start with the best coalitions, it follows the patters layer by layer from left to right and from top to bottom in each layer. For each pattern, the agent will choose a combination of it's own best coalitions and those it received from other agents to generate coalition structures. For example, with a pattern of 4 + 3 + 2, the agent will place it's best coalition of cardinality 4 as the first coalition of that coalition structure. One of the best coalitions of cardinality 3, whose members are not in the first coalition, will be placed as the second coalition. One of the best coalitions of cardinality 2, whose member is not in the first two coalitions will be placed as the coalition structure as the last coalition. In the case the agent can not find appropriate coalitions to fit in, it places an empty set instead. The coalition structure value is the sum of those coalition values. In each round of proceeding through all patterns, agent can extend the scope of best coalitions involved one by one. It, for example, generates the coalition structure using the only best coalitions in the first round. It then use the best plus the second best coalitions for the second round, and so on. The following is the algorithm for generating coalition structures is shown in algorithm 3:

Algorithm 3 Generating Coalition Structures

```
  exchange best coalitions with all other agents
  sort coalitions for each cardinality by their coalition values
  in descending order
  generate patterns for each layer
  set bestcoal to 1
  while time is available do
    insert the bestcoal coalitions for each CScardinality
    for all layers do
      for all patterns do
        generate combinations of best coalitions in CScar-
        dinality
      end for
    end for
    increase bestcoal by 1
  end while
```

3.3 Example

This section gives an example of how this algorithm works. Let the system is composed of a set of four agents: $A = \{a_1, a_2, a_3, a_4\}$. After the first deliberation process, all the coalition values are computed and sent across. Their values are as the followings:

$$\begin{array}{lllllllll}
v_1 = 8 & v_{12} = 13 & v_{123} = 21 & v_{1234} = 22 \\
v_2 = 12 & v_{13} = 16 & v_{124} = 23 & \\
v_3 = 13 & v_{14} = 10 & v_{134} = 16 & \\
v_4 = 6 & v_{23} = 18 & v_{234} = 19 & \\
 & v_{24} = 20 & & \\
 & v_{34} = 15 & &
\end{array}$$

After exchanging the coalitions generated among each other, each agent can select for each cardinality it's best coalition. Let assume that agents only operate on the best coalitions. Agents' best coalitions are as the followings:

Cardinality	a_1	a_2	a_3	a_4
1	v_1 8	v_2 12	v_3 13	v_4 6
2	v_{13} 16	v_{24} 20	v_{23} 18	v_{24} 20
3	v_{124} 23	v_{124} 23	v_{123} 21	v_{124} 23
4	v_{1234} 22	v_{1234} 22	v_{1234} 22	v_{1234} 22

For the system of 4 agents, the breaking patterns of coalitions are as the followings:

No. of coalitions	1	2	3	4
Patterns	4	3 + 1	2 + 1 + 1	1+1+1+1
		2 + 2		

Using the algorithm in the second deliberation process, each agent's coalition structures computed are shown below. Each agent will achieve the same optimal coalition structure whose value is 41.

a_1
$CS_{1234} = 22$
$CS_{124,3} = 23 + 13 = 36$
$CS_{1,234} = 8 + 0 = 8$
$CS_{13,24} = 16 + 20 = 36$
$CS_{13,2,4} = 16 + 12 + 6 = 34$
$CS_{1,23,4} = 8 + 18 + 6 = 32$
$CS_{1,2,34} = 8 + 12 + 0 = 20$
$CS^*_{1,3,24} = 8 + 13 + 20 = 41$
$CS_{1,2,3,4} = 8 + 12 + 13 + 6 = 39$

a_2
$CS_{1234} = 22$
$CS_{124,3} = 23 + 13 = 36$
$CS_{2,134} = 12 + 0 = 12$
$CS_{24,13} = 20 + 16 = 36$
$CS^*_{24,1,3} = 20 + 8 + 13 = 41$
$CS_{2,13,4} = 12 + 16 + 6 = 34$
$CS_{2,3,14} = 12 + 13 + 0 = 23$
$CS_{2,1,34} = 12 + 8 + 0 = 20$
$CS_{1,2,3,4} = 8 + 12 + 13 + 6 = 39$

a_3
$CS_{1234} = 22$
$CS_{123,4} = 21 + 6 = 27$
$CS_{3,124} = 13 + 23 = 26$
$CS_{23,14} = 18 + 0 = 18$
$CS_{23,1,4} = 18 + 8 + 6 = 32$
$CS^*_{3,1,24} = 13 + 8 + 20 = 41$
$CS_{3,2,14} = 13 + 12 + 0 = 25$
$CS_{1,2,3,4} = 8 + 12 + 13 + 6 = 39$

a_4
$CS_{1234} = 22$
$CS_{124,3} = 23 + 13 = 39$
$CS_{4,123} = 6 + 21 = 27$
$CS_{24,13} = 20 + 16 = 36$
$CS^*_{24,1,3} = 20 + 8 + 13 = 41$
$CS_{4,13,2} = 6 + 16 + 12 = 32$
$CS_{4,23,1} = 6 + 18 + 8 = 32$
$CS_{1,2,3,4} = 8 + 12 + 13 + 6 = 39$

4. EXPERIMENT

We conduct experiment by simulating agents executing our algorithm against exhaustive search within the range of $20 - 50$ agents due the the limitation to run exhaustive search. We compare the performance of both algorithms in terms of number of partitions generated and elapsed time of generating best coalition structures. In each round, the agents number increases by 5. The number of goods and resources are equal, begins from 3 and increase by 1 in every 2 rounds. The technology matrix, agents' resources and cooperation costs among agents are randomly generated with uniform distribution. The number of each resource α_{ij} in the technology matrix is in the range $0 - 10$. The prices of the goods are in the range of $10 - 20$ while the cooperation costs are in the range of 0 and the number of agents in that round, e.g., $10, 15, \ldots$. We test our algorithm with the 5th best coalitions only. As our algorithm deals with non-superadditive environments, this setting tends to increase the cooperation cost of a coalition as its size grows. Hence it forces agents to work harder to form profitable coalitions and to achieve optimal coalition structures. Both algorithms uses the Simplex algorithm to find the optimal solution for each coalitions. Figure 1 compares the performance of our algorithm against that of exhaustive search. The left x-axis is the number of coalition structures generated while the right x-axis is the elapsed time spent for generating optimal coalition structures in milliseconds. Since the data used is randomly generated, we present average values of various which constantly show signficant difference between results of the two algorithms. The empirical results show that our algorithm performs significantly better than exhaustive search. We experienced that exhaustive algorithm hardly make progress after the number of agents is larger than 40. As shown in the figure, the number of coalition structures generated by exhaustive algorithm is much larger than that of our algorithm. Furthermore, the elapsed time for generating optimal coalition structures of exhaustive search is also much larger than that of our algorithm.

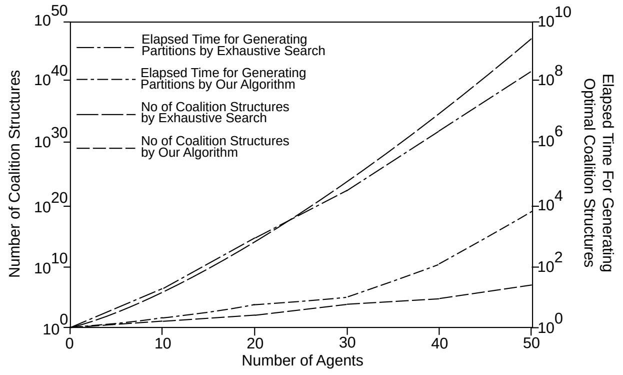

Figure 1: This graph shows the number of coalition structures generated and elapsed time for generating the coalition structures of our algorithm against those of exhaustive search.

5. RELATED WORK

Shehory et. al [8] propose an algorithm to allocate tasks to agents in distributed problem solving manner, i.e., agents try to maximize the utility of the system. They consider a domain where a task composed of multiple subtasks, each of which requires specific capacity. These tasks have to be carried out by agents who have specific capacities to carry out tasks. Each agent prepares its list of candidate coalitions and proposes to other agents. Shehory et. al. [9] study overlapping coalition formation in distributed problem solving systems in non-superadditive environments. Although agents can belong to multiple coalitions at the same time, agents execute one task at a time. The task allocation process is completed prior to the execution of the tasks. Agents are group-rational, i.e., they form coalition to increase the system's payoff. Sandholm et. al. [7] analyze coalition formation among self-interested agents who are bounded-rational. They consider deliberation cost in terms of monetary cost. The agents' payoffs are directly affected by deliberation cost. In their work, agents agree to form coalition and each of the agents can plan to achieve their goals. Soh et. al. [10] propose an integrated learning approach to form coalition in real time, given dynamic and uncertain environments. This work concentrates on finding out potential coalition members by utilizing learning approach in order to quickly form coalitions of acceptable quality (but possibly sub-optimal.) Sandholm et. al. [6] study the problem of coalition structure generation. Since the number of coalition structures can be very large for exhaustive search, they argue whether the optimal coalition structure found via a partial search can be guaranteed to be within a bound from optimum. They propose an anytime algorithm that establishes a tight bound withing a minimal amount of search.

6. CONCLUSION AND FUTURE WORK

We propose an algorithm for computing optimal coalition structure for linear programming domains among fully cooperative agents. Our algorithm tries generate best coalitions by pruning the least profitable from the grand coalition. Then the coalitions generated will be exchanged among agents. Lastly, agents use coalitions exchanged to generate coalition structure. The empirical results show that our algorithm help generate the optimal coalition structures much faster than exhaustive search. Our algorithm dramatically reduces the number of coalitions generated hence reducing the number of coalition structures. As a result, the elapsed time of generating the coalition structures is relatively small.

Although this algorithm helps reduce number of coalitions involved in generating optimal coalition structures, there is always rooms to improve. We want to further improve our algorithm for larger number of agents, for example, up to 1000 agents. Lastly, we want to study this problem in related domains, e.g., non-linear programming.

7. REFERENCES

[1] V. D. Dang and N. R. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proc. of AAMAS-2004*, 2004.

[2] V. D. Dang and N. R. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2 (AAMAS'04), pp. 564-571*, 2004.

[3] J. P. Kahan and A. Rapoport. *Theories of Coalition Formation*. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1984.

[4] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behaviour*. Princeton University Press, Princeton, New Jersey, 1953 (1963 printing).

[5] G. Owen. On the core of linear production games. *Mathematical Programming 9 (1975) 358-370*, 1975.

[6] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohm. Coalition structure generation with worst case guarantees. *Artif. Intell.*, 111(1-2):209–238, 1999.

[7] T. Sandholm and V. Lesser. Coalition Formation among Bounded Rational Agents. *14th International Joint Conference on Artificial Intelligence*, pages 662–669, January 1995.

[8] O. Shehory and S. Kraus. Task allocation via coalition formation among autonomous agents. In *Proc. of IJCAI*, pages 655–661, August 1995.

[9] O. Shehory and S. Kraus. Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. In *ICMAS-96*, pages 330–337, December 1996.

[10] L.-K. Soh and C. Tsatsoulis. Satisficing coalition formation among agents. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1062–1063. ACM Press, 2002.

[11] C. Sombattheera and A. Ghose. A distributed algorithm for coalition formation in linear production domain. In *Proceedings of ICEIS 06*, May 2006.

[12] C. Sombattheera and A. Ghose. A distributed branch-and-bound algorithm for computing optimal coalition structures. In *Proceedings of the 4th Hellenic Conference on Artificial Intelligence*, May 2006.