Task Allocation via Multi-Agent Coalition Formation: Taxonomy, Algorithms and Complexity

Hoong Chuin LAU Lei ZHANG
School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543 Tel: +65-68744589
lauhc@comp.nus.edu.sg tlizl@nus.edu.sg

Abstract

Coalition formation has become a key topic in multi-agent research. In this paper, we propose a preliminary classification for the coalition formation problem based on three driving factors (demands, resources and profit objectives). We divide our analysis into 5 cases. For each case, we will present algorithms and complexity results. We anticipate that with future research, this classification can be extended in similar fashion to the comprehensive classification for the Job Scheduling Problem.

1. Introduction

With the Internet, it is now possible for operations to be optimized dynamically across company boundaries. In virtual marketplaces where buyers and sellers could come together to conduct business transactions for instance, service providers along the supply chain can form dynamic alliances so as to pool their resource capacities to serve complex customer requests, to exploit economies of scale and to achieve better utilization of spare capacities. In this paper, we are concerned with the problem of task allocation via dynamic coalition formation of multiple agents. We consider the situation where each task comprises several sub-tasks that may not be satisfied by a single agent alone but by a group of agents offering different resource capabilities. Given a set of agents and a set of tasks, the problem is in deciding how best to form coalitions so as to maximize the total profit. We term this the *Task Allocation Coalition Formation* (TACF) problem.

In this paper, we provide a classification for TACF (see diagram on last page). This classification is designed to emulate the classification for the classical Job Scheduling Problem (see, for example [5]), where the three key parameters are job characteristics, resource constraints and objective function.

The first factor for TACF is the *task demand*. A real-world example of a task is a product or a service. In this paper, we deal with two types of tasks: tasks with *high* demand and *unit* demand. When the demand for a task is high, it means that the agents can perform the task repeatedly as much as they like with the available resources. An example of this situation is seen in the demand for certain popular toys before the Christmas season, which is high such that it is profitable to produce as much as possible with the available resources. On the contrary, for the unit demand case, agents will only have the choice of either performing the task exactly once or not at all.

The second factor is the *resource constraints*, i.e. whether the total resources available are *limited* or *unlimited*. For the former, we mean the total resources available over all agents are below the total resource requirements for all the tasks while for the latter, we mean the total resources available over all agents exceed the total resource requirements for all tasks. When resources are limited, we cannot fulfill all the tasks and thus encounter the problem of choosing the best subset of tasks to maximize the objective value. When resources are unlimited, we can fulfill all the tasks and the problem lies with how coalitions should be formed, since agents have different asking prices for performing the task.

The last factor is the *objective function*. In this problem, we are concerned with the profit derived from serving a task. We term the profit of a task *variable* if the profit derived from this task depends on which set of agents perform the task (profit is equal to revenue minus the total asking price of the agents in the coalition). On the contrary, we call the profit *fixed* if agents have no asking prices and hence profit is task-dependent. The fixed profit case is a special case of the variable profit by setting all agent asking prices to be equal.

Based on our proposed classification, we identify five distinct cases to be studied in this paper.

The first case is with unit demand and unlimited resources, and the profit can be either fixed or variable. This case is easy, and we propose an optimal algorithm with a worst-case time complexity of $O(n \log n + mn)$, where n is the number of agents and m is the number of tasks in the system. The second case is with unit demand, limited resources, and fixed profit, which is NP-hard. For this case, we prove that the greedy algorithm based on profit alone as the criteria to form coalition cannot establish any approximation bound on the quality of the solution. We propose a modified greedy algorithm that considers the resource usage and the effects of look-ahead. Experimental results to illustrate the strength of this algorithm are provided. The third case is with unit demand, limited resources, and variable profit. For this case, we propose a modified greedy algorithm and again verify with experimental results. The fourth case is with high demand, limited resources, and fixed profit. For this problem, we

propose a polynomial-time approximation scheme, which establishes a constant bound on the quality of the solution. The last case is with high demand, limited resources, and variable profit. For this problem, we use a similar approach as the fourth case except that we use the average asking price of the agents to approximate the task profit. Note that the case of high demand and unlimited resources is meaningless, since the profit becomes infinite.

Our algorithms are centralized, i.e. we assume there is a central authority that distributes the tasks among the agents. Although all the computations are done through the central authority, it is not a big problem with real-time computation since our proposed algorithms are of low time complexities. This differs from the work of [8] where there is a need to distribute computations among agents due to high computational complexity.

The paper is organized as follows: section 2 gives a brief literature review. In section 3, we define notations and assumptions made. In section 4, we present results, both theoretical and experimental, to solve the different problem cases. Due to space constraints, all proofs and experimental results are omitted in this version of the paper. Section 5 concludes with future works.

2. Related Works

In this section, we review key recent works by the Distributed Artificial Intelligence community under the topic of agent coalition formation. [8] presents an iterative distributed greedy algorithm for coalition formation under Case 3 of our taxonomy. In each iteration, all possible coalition values of coalitions of a certain size need to be re-computed. The nice thing about their approach is that the overall computation process can be distributed among all the agents such that each agent will only compute some coalition values. There is no central authority that distributes the tasks among the agents: agents reach an efficient task allocation by themselves. The greedy algorithm proposed for forming coalitions considers profit alone and the time complexity of the algorithm is quite high since it grows exponential with the size of the coalitions. .

[9] presents two coalition formation approaches for *self-interested* agents in a non-super-additive environment. The computation-oriented approach searches an exponential solution space that guarantees an optimality bound. The stability of the coalition formation is also ensured in this approach, but individual agent's selfish behavior is not reflected. On the other hand, the negotiation-oriented approach is stable and efficient, but the quality of the solution cannot be guaranteed. In our paper, we adopt the computation-oriented approach. However, instead of searching an exponential solution space, we are concerned with polynomial-time algorithms.

[10] presents a stable and efficient buyer coalition formation scheme called `GroupBuyAuction` for E-marketplaces. In this scheme, buyers form a group based on a category of items and benefit form coalitions in E-marketplaces by taking advantage of volume discounts. It provides buyers with means to declare and match their preferences and compute the division of the surplus in a stable fashion.

[7] shows that none of the coalition structure generation algorithms previously studied can establish any bound because they search fewer nodes than a threshold that is necessary for establishing a bound. They present an algorithm that establishes a tight bound within this minimal amount of search, and show that any other algorithm would have to search strictly more.

In this paper, we draw a close relationship with the multi-dimensional knapsack problem. In [1] and [2], the authors propose approximation algorithms for the integer multi-dimensional knapsack problem. In the former, the authors present the algorithm that for every $\varepsilon > 0$ one can construct a polynomial-time algorithm such that the ratio of the value of the objective function by this algorithm and the optimal value is bounded below by $1 - \varepsilon$. Case 4 of our taxonomy can actually be transformed to the integer multi-dimensional knapsack problem and the approximation algorithm introduced can be used to construct a polynomial-time algorithm for Case 4.

3. Preliminaries

In this section, we present definitions, notations and assumptions used throughout this paper.

3.1 Tasks

There are m tasks in the system. Each task i consists of a fixed k number of sub-tasks. Each sub-task j requires a unique resource of its type. Let a_{ij} denote the quantity of resources required for performing task i sub-task j. For the case when the profit is fixed, each task i is associated with a profit value p_i. Otherwise, it is associated with a revenue value r_i and its profit is simply r_i minus the sum of the asking prices of the agents in the coalition serving that task.

3.2 Agents

There are n agents in the system denoted as $A_1, A_2, \ldots, A_n$. Each agent A_i is associated with a vector of non-negative resource capacities $C_i = [c_{i1}, c_{i2}, \ldots, c_{ik}]$, where c_{ij} is agent A_i's quantity of resource capable of performing sub-task j. (Obviously, if the agent cannot perform a certain sub-task, the corresponding value is set to 0.) A sub-task may be performed by one or more agents. For the case when the profit is a variable, each agent has an additional price vector $U_i = [u_{i1}, u_{i2}, u_{i3}, \ldots, u_{ik}]$, where u_{ij} is the unit asking price for agent A_i to perform sub-task j.

3.3 Coalition

A coalition is defined as a group of agents to achieve a common task. Each coalition is associated with a *coalition value*, which is intuitively defined as the joint utility that the members of a coalition derive by cooperating to satisfy a specific task. The specific definition of this value will be discussed in Section 4. In this paper, we allow overlapping coalitions, i.e. each agent can belong to more than one coalitions.

3.4 Model

In this paper, we assume that agents are group-rational, i.e. agents are co-operative and interested in maximizing the overall profits gained by the system. Hence, the objective function in this paper is the sum of profits of all agents which is equivalent to the sum of profits of all the tasks fulfilled. This differs from the game-theoretic school of thought where agents are assumed to be self-interested and hence the considerations of objective function are based on incentives and payoff distribution. Hence, the TACF problem can be modeled as follows:

$$\text{Maximize} \quad \sum_{i=1}^{m} p_i x_i$$

$$\text{subject to} \sum_{i=1}^{m} a_{ij} x_i \le C^j, \; j=1, 2 \ldots k$$

where x_i is a non-negative integer for the high demand (or boolean for unit demand) case, and C^j is the total resources available for sub-task j, which is the j^{th} column sum of the capacity matrix C defined above.

4. Algorithms and Complexity

In this section we present theoretical and experimental results for the 5 cases of our taxonomy.

4.1 Case 1: Unit Demand and Unlimited Resources

When the resources available are unlimited, all the tasks will be fulfilled eventually. Hence, if the profit for each task is *fixed*, the task allocation problem becomes trivial since the total profit gained remains a constant equal to the sum of profits for all tasks, no matter how we form the coalitions. On the other hand, if the profit is *variable*, then maximizing profits is equivalent to minimizing the total price asked by the agents in the coalition for fulfilling the task, given that resources are unlimited and hence all the tasks can be fulfilled regardless of the task allocation.

Consider the greedy algorithm G that behaves as follows.

Step 1. For each sub-task j, sort all the agents with non-zero capacity of performing this sub-task in the non-decreasing order of their unit asking prices. We call the sorted list the agent list for sub-task j.

Step 2. For each task i do the following:

for each sub-task j of task i, pick agents in order from the agent list of sub-task j until the resource requirement for sub-task j is met.

Theorem 1: G is optimal.

Corollary: TACF under unit demand and unlimited resources can be solved optimally with time complexity $O(n \log n + mn)$.

4.2 Case 2: Unit Demand, Limited Resources, Fixed Profit

Contrary to Case 1 where a greedy algorithm based on profits alone produces the optimal solution, TACF becomes NP-hard when resources are limited even for a single agent, since this problem is a generalization of the standard knapsack problem which is NP-hard. In this section, we first show that any greedy algorithm based on profits alone will fail miserably. We then propose a modified greedy algorithm that produces good results experimentally.

Theorem 2: It is impossible to establish a constant approximation bound with the above greedy approach if the greedy criterion is the profit value alone.

This negative result can be proved by construction of a counter example.

Our proposed approach to solve this problem is to iteratively construct coalitions by maximizing the utility (or coalition value). In this way, we expect the overall profit of the system to be maximized as well. There are two stages in the algorithm:

Stage 1: *Coalition value computation*. For each remaining task i in the task list, compute its coalition value according to following formula:

$$\frac{p_i}{\sqrt{a_{i1} * f_{i1} + a_{i2} * f_{i2} + \ldots + a_{ik} * f_{ik}}}$$

where $f_{it} = \sum_{j \neq i} \frac{a_{jt}}{R_t - a_{it}}$, where the sum is over all j of remaining tasks in the task list, R_t is the total remaining capacities (of all agents) for resource t before allocating task i.

We explain the rationale for computing the coalition value this way. The value represents the utility for choosing task i to be fulfilled. Obviously, the higher p_i is, the higher the utility. Since a_{ij} measures the resource usage, the smaller a_{ij} is, the higher the utility. However, this must be normalized against f_{ik}. which is a fraction between 0 to 1 that measures how scarce the resource k will become for the remaining tasks after allocating task i. Hence, the higher the f_{ik} value, the more scarce the

resource becomes for the rest of tasks and the lower the utility.

Stage 2: *Coalition formation*. Form the coalition for the task with highest coalition value among all tasks.

1. Choose the highest among all coalition values computed in Stage 1. That coalition will be formed and the corresponding task for it would be assigned to the coalition.
2. For each agent that becomes a member of the coalition formed, the resource capabilities are updated.
3. The task assigned to the coalition at this iteration is removed from the task list.
4. Goto Stage 1. Stop when no more task is left or no more coalition can be formed.

Complexity Analysis:
In Stage 1, the coalition value for each task can be computed in O(mn) time. Thus Stage 1will take O(m^2n) time to compute the coalition values for all the tasks in the task list in each iteration. Stage 2 takes O(max(m,n)) to form coalition in each iteration.. These two stages will repeat for at most m times since the maximum number of coalitions that can be formed is m. Hence, the algorithm takes O(m^3n) time.

Experimental Results:
This case of the problem is in fact the 0/1 multi-dimensional knapsack problem. Thus we can use the benchmark test instances for this problem obtained from the OR-Library (available on line at http://mscmga.ms.ic.ac.uk/info.html).
Among these instances, the dimensions of the knapsack range from 2 to 30 and the number of items range from 20 to 105. For each test problem, we obtain the result using our greedy algorithm, as well as the algorithm with a local improvement phase (more specifically, tabu search [3]), and the optimal result provided with the test problem. We measure the gap between our results and the optimal result. For the greedy algorithm, 2 out of 54 tested instances produce results with the gap range within 10%; 6 test problems with the gap range of 4% to 6%; 4 test problems with gap range of 1% to 2%; the other 42 test problems obtain results with difference of less than 1%. After applying tabu search on the solution found by greedy, 50 test problems produce results with gap of less than 2%; 3 test problems obtain results with gap range of 2% to 4% and only 1 test problem with gap range of 4% to 6%.

4.3 Case 3: Unit Demand, Limited Resources, Variable Profit

For this case, the challenge is in handling the *agent-dependent* unit asking prices for each sub-task. In this section, we propose an algorithm that extends the algorithm for Case 2. This algorithm has three stages:

Stage 1: *Preprocessing.* For each sub-task, we sort all agents that have capabilities for the sub-task in the increasing order of their *unit asking prices*.

Stage 2: *Coalition value computation.* This is a greedy procedure: in each iteration, we compute the coalition value of the coalition that can best fulfill each task. For each sub-task j of task i, pick in order from agent list as many agents as necessary to fulfill j. The selected agents will form the coalition which best fulfills task i. Compute the coalition value of this coalition by using the same formula as defined in Case 2 Stage 1.

Stage 3: *Coalition formation*. Same as Case 2 Stage 2.

Complexity Analysis:
In Stage 1, we can sort agents for each sub-task within O(n log n) and we have constant number of sub-tasks, thus Stage 1will take O(n log n) time. In Stage 2, the coalition value for each task can be calculated within O(mn), thus it will take O(m^2n) to calculate all the coalition values in each ieration. Stage 3 takes O(max(m,n)) to form coalition in each iteration. Stage 2 and 3 will repeat for at most m times since the maximum number of coalitions that can be formed is m. It can be concluded that the algorithm for Case 3 will take O(m^3n) + O(n log n).

Experimental Results:
Since there is no benchmark available for this problem, we compare our approach with the same algorithm with the exception that it uses the profit p alone to compute the coalition value in Stage 2. We again use the OR-Library instances as described in Case 2 and randomly generate the agent unit asking prices in the real number range [1,2]. Our modified algorithm outperforms the standard greedy algorithm in 49 test instances out of the 54 test instances.

4.4 Case 4: High Demand, Limited Resources, Fixed Profit

In this and the next sections, we consider the TACF problem under *high demand*. For the case where resources are limited and profit is fixed, we model the problem as a multi-dimensional knapsack problem. We present a complexity result and a polynomial-time approximation scheme that follows from the work of [1].

The multi-dimensional knapsack problem is defined as:

Maximize $\sum_{i=1}^{m} p_i x_i$

subject to $\sum_{i=1}^{m} w_{ij} x_i \le W_j$, j=1, 2…k

where p_i is the value of item i; x_i is a non-negative variable; w_{ij} is the item i's weight in the jth dimension of the knapsack, and W_j is the maximum weight of the jth dimension. Given an integer P, the decision problem asks whether there exists an allocation with a value of at least P.

It is known that even though the decision version of multi-dimensional knapsack problem is NP-complete in the weak sense, [4] shows that no fully polynomial time approximation scheme exists, unless P =NP. Hence, the following negative result holds:

Theorem 3: The limited resource fixed-profit TACF does not have a fully polynomial time approximation scheme, unless P=NP, even for a single agent.

However, we can obtain the following approximation result:

Theorem 4: For every constant ε>0, there exists an algorithm TACF $_\varepsilon$ that takes time O($m^{\lceil k/\varepsilon \rceil}$) (for some constant k) such that the approximation bound is at least 1-ε.

4.5 Case 5: High Demand, Limited Resources, Variable Profit

For this problem, we first study how the profit factor affects the complexity of the problem compared to Case 4. Then, we propose a heuristic method to convert this problem to the Case 4 problem.

Recall that the Case 4 problem can be modeled as follows:

Maximize $\sum_{i=1}^{m} p_i x_i$

subject to $\sum_{i=1}^{m} a_{ij} x_i \le C^j$, j=1, 2…k

where $p_i = r_i - s_i$, r_i is the revenue gained for doing this task, and s_i is the total amount paid to agents participating in this task

Note that the value of s_i will depend on which set of agents serve the task, and is thus a *variable*. Hence, p_i also becomes a variable. Consequently, this problem is no longer a linear programming problem since both p_i and x_i are variables. Thus we cannot apply the well-known result in linear programming as we did for Case 4. Instead, we propose the following heuristic method that makes the profit "constant". For task i denoted as [a_{i1}, a_{i2},…, a_{ik}] with revenue r_i, we estimate its profit as:

$$p_i = r_i - (AP_1 * a_{i1} + AP_2 * a_{i2} \ldots + AP_k * a_{ik})$$

where AP_j is the average unit price for sub-task j and it is defined as $AP_j = \sum_{i=1}^{n} (u_{ij} * c_{ij}) \Big/ \sum_{i=1}^{n} c_{ij}$.

Now the profit for each task becomes a "constant" and we can then apply the result presented for Case 4 to this problem. Obtaining computational results for this approach is an ongoing work.

5 Conclusion

In this paper, we proposed a taxonomy that categorizes the TACF problem according to 3 driving factors we believe to be important in E-market places. As this is the first such taxonomy, we anticipate much future research work to refine the results presented here. For example, we can potentially extend this taxonomy to consider other factors such as agent self-interest, profit distribution and other constraints relevant in a multi-agent environment setting.

References

[1] M. Ashok, K. Chandea, D. S. Hirschberg, and C. K. Wong. Approximation Algorithm for Some Generalized Knapsack Problems. *Theo Comp. Sci.* 3, 293-304, 1976.

[2] A. M. Frieze and M. R. B. Clarke. Approximation algorithms for the m-dimensional 0-1 knapsack problem: worst-case and probabilistic analyses. *European J. Operational Research*, 15(1), 100-109, 1984.

[3] F. Glover and M. Laguna. Tabu Search. Readings. Kluwer Academic Publishers, 1997.

[4] B. Korte and R. Schnader. On the Existence of Fast Approximation Schemes. In Mangasarian et al. (eds), *Nonlinear Programming* 4, 415-437, Academic Press, 1981.

[5] E. Lawler *et al.,* Sequencing and Scheduling: Algorithms and Complexity. In *Handbooks in Operations Research / Management Science,* v. 4*,* North Holland, 445-522, 1993.

[6] C. L. Liu. Introduction to Combinational Mathematics, Readings, McGraw-Hill, N.Y., 1968.

[7] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalition Structure Generation with Worst Case Guarantees. *Artificial Intelligence*, 111(1-2), 209-238, 1999.

[8] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101, 165-200, 1998.

[9] O. Shehory and S. Kraus. Feasible formation of coalitions among autonomous agents in non-super-additive environment. *Computational Intelligence*, 15(3), 218-251, 1999.

[10] J. Yamamoto and K. Sycara. A Stable and Efficient Buyer Coalition Formation Scheme for E-Marketplaces. In Proc. 5th Int'l Conf. On Multi-Agent Systems (ICMAS), Montreal, 2001.