Multi-agent Based Execution Environment for Task Allocation via Coalition Formation

Merve Özbey and Nadia Erdoğan

Istanbul Technical University, Faculty of Computer and Informatics, Computer Engineering Department 34469, Maslak Istanbul, Turkey
merve.ozbey@live.com,
nerdogan@itu.edu.tr

Abstract. This paper focuses on a solution to the problem of task allocation through coalition formation and presents the design and implementation of a framework for this purpose. The framework serves as a multi-agent execution environment where worker agents follow the Shehory-Kraus's coalition formation algorithm to negotiate and form coalitions. Worker agents prefer to form an optimal coalition to maximize the joint utility of the system as a whole, rather than to maximize their own utilities. There is no central authority for distribution of tasks among the agents. The framework provides the infrastructure, which, allows the formation of coalitions, assigns a task to each coalition and then monitors task execution, taking the necessary steps for rescheduling of tasks in case of noncompletion due to agent errors, generating and recording of task execution reports, handling cyclic execution of coalition formation process. In addition, an efficient and fast messaging infrastructure has been developed for effective agent communication.

Keywords: Multi-agent system, negotiation, coalition formation.

1 Introduction

Multi-agent systems are systems of intelligent autonomous agents which communicate and collaborate with each other. Agents cooperate when the capability and knowledge of each agent alone is not sufficient to solve a problem. Coalition formation is an important method for cooperation in a multi-agent environment. It is a process where agents form coalitions and work together to solve a joint problem via cooperating or coordinating their actions within each coalition.

In this paper, we present a framework which serves as a multi-agent execution environment for coalition formation. Agents (employee) are autonomous. They follow the Shehory-Kraus's algorithm [1] to negotiate and decide on coalitions, with no user interaction. The reason behind the choice of this algorithm is that, instead of having a single agent calculate all possible coalitions, the algorithm distributes the coalitional value calculations among agents, thus both improving execution time and preventing the existence of a single point of failure. They prefer to form an optimal coalition to maximize the joint utility of the system as a whole, rather than to maximize their own

G. Jezic et al. (eds.), *Agent and Multi-Agent Systems: Technologies and Applications*, Advances in Intelligent Systems and Computing 296,
DOI: 10.1007/978-3-319-07650-8_17,

utilities. There is no central authority for distribution of tasks among the agents. Each agent has a predefined task execution capability and agents negotiate with each other concurrently until they form a grand coalition to perform a predefined task. Agents cannot be members of multiple coalitions, as the algorithm enforces disjoint coalitions. After a task is completed, the members of the coalition can join new coalitions. Furthermore, agents cannot join an already formed coalition due to extra agent addition cost. It is assumed that tasks have no precedence order and inter dependency.

The framework we present provides an infrastructure that supports multiple agents with different roles that cooperate to present the following functionalities:

- definition of employee agent capabilities and task requirements
- formation of coalitions
- assignment of tasks to each coalitions
- monitoring of task execution
- rescheduling of tasks in case of error and noncompletion
- generation and recording of task execution reports
- dynamic inclusion of new tasks
- cyclic execution of coalition formation process

The framework builds on various types of agents which are defined and implemented to handle different issues. In the design phase, we have determined the actors of the system, specifying their tasks and responsibilities. After associating each actor with an agent type, e.g. employee agents, a controller agent, coalition manager agents and a database agent, protocols that define in detail the interaction and coordination between agents were developed. In addition, an efficient and fast messaging infrastructure has been developed for effective agent communication. The framework will be used for further research on coalition formation.

2 Related Work

In existing multi-agent literature there are several works about negotiation and coalition formation. Shehory and Kraus's work [1] proposes several methods for task allocation through coalition formation. These methods are disjoint coalitions, overlapping coalitions and coalitions for tasks that have precedence orders. In this work, we focus on disjoint coalitions for tasks with no precedence orders, leaving the addition of precedence to tasks as future work.

Shehory and Kraus's another study with Taase [2] proposes a protocol that enables agents to negotiate and form coalitions with time constraints and incomplete information. Test results show that their solution is close to optimal solutions. Continuation of this study [3] also shows that their solution gives good results with incomplete information under time constraints.

Ferber, Gutknecht and Michel's study [4] proposes an organizational view of multi-agent systems. They define roles for agents and assign a group manager to each agent groups. Group managers are responsible from agents in their group.

[5] presents a MAS based infrastructure for the specification of a negotiation framework in multi-agent systems. They present role definitions for agents, and a negotiation environment with design details. Another study [6] also proposes an application of coalition formation in multi-agent systems particularly for electricity markets. The framework presented is generic, appropriate for any type of application.

A current study [7] presents coalition formation process in detail. They present definition of equilibrium process in coalition formation and describe basic concepts of classical game theory which has coalitional constraints. They show how to cover some of the coalitional stability's standard notions using equilibrium process concept.

3 Shehory-Kraus Coalition Formation Algorithm

Employee agents in the execution environment follow the Shehory-Kraus's algorithm [1] to negotiate and decide on coalitions. There exist several variations of the algorithm; our implementation focuses on disjoint coalitions. We first state the underlying assumptions and definitions, and then describe the algorithm.

3.1 Assumptions

- Agents can communicate, negotiate and make agreements.
- Coalition enlargement is not considered, due to the cost of addition of agents to an already formed coalition.
- Agents cannot participate multiple coalitions.
- There exists no explicit dependency between tasks.
- Agent population does not change during coalition formation.
- Complete information is not required. However, all of the agents know about all the tasks and all other agents, and coalition members know all the details necessary for fulfilling their task.
- In this work, it is assumed that agents are group rational. Agents are not concerned with their own profits, but aim to maximize the total outcome of the system.

3.2 Definitions

- Agent set consists of *n* agents, $N = \{A_1, A_2, \dots, A_n\}$. For each agent A_i, a capability vector $B_i = \langle b_1^i, b_2^i, \dots, b_r^i \rangle$ has been defined. Each capability is a property of an agent which quantifies its ability to perform a specific action.
- Task set consists of *m* independent tasks, $T = \{t_1, t_2, \dots, t_m\}$. For the fulfillment of each task t_l, a vector of capabilities $B_l = \langle b_1^l, b_2^l, \dots, b_r^l \rangle$ is necessary. The utility gained from performing the task is a linear function of resource amount.
- Expected outcome after executing a task is selected as a linear function for decreasing complexity of calculations.
- A coalition *C* has a vector of capabilities B_c which is the sum of capability vectors of the coalition member agents. A coalition *C* can perform a task *t* only if all capabilities in the *t*'s capability vector B_t are satisfied by $b_i^t \leq b_i^C$ for all $0 \leq i \leq r$.

- The joint utility of the coalition members by contributing to a coalition is defined as the coalitional value, *V*.
- Coalitional cost, *c,* is calculated as the reciprocal of the coalitional value.

- ***Coalition formation problem:***

Given a set of m independent tasks, $T = \{t_1, t_2, \ldots, t_m\}$ and a set of n agents, $N = \{A_1, A_2, \ldots, A_n\}$ with their capabilities, the coalition formation problem is assigning tasks $t_j \in T$ to coalitions $C_i \subseteq N$ such that ΣiVi (the total outcome) is maximal.

3.3 Shehory-Kraus Coalition Formation Algorithm

The algorithm proceeds in three stages. The agents that negotiate execute the steps in all three stages concurrently, contributing to the process in distributed manner.

- **Stage 1:** All possible coalitions are calculated.
- **Stage 2:** Coalitional values of possible coalitions are calculated.
- **Stage 3:** Agents decide upon preferred coalitions and form them through iterations.

First Stage of the Algorithm
The first stage entails the calculation of the coalitional values in a distributed manner. The maximum coalition size, k, is initialized. Each agent communicates with others and eliminates duplicate coalitions from its potential coalition list L.

1. Calculate all of the possible coalitions up to size k in which A_i is a member and form P_i, the set of the potential coalitions of agent A_i.
2. For each coalition in P_i, contact each member A_j and retrieve its capabilities B_j.
3. Commit to the calculation of the values of a subset S_{ij} of the common potential coalitions (i.e., a subset of the coalitions in P_i in which both A_i and A_j are members).
4. Subtract S_{ij} from P_i. Add S_{ij} to long-term commitment list L_i.
5. For each agent A_k that has contacted A_i, subtract from P_i the set S_{ki} of the potential coalitions A_k had committed to compute values for.
6. Repeat contacting of other agents until $Pi = \{A_i\}$.

Second Stage of the Algorithm
After the first stage, the agent has a list of coalitions for which it had committed to calculate the values. It also has all of the necessary information about the capabilities of the members of these coalitions. In the second stage each agent checks whether a coalition's capability vector is sufficient for the tasks in the task list. Coalition's capability vector is the sum of capability vectors of the agents in that coalition. For sufficient coalitions; agent calculates expected values. Then, agent selects the maximum expected value as the coalition value for each coalition, calculates coalitional cost.

1. Calculate the coalitional potential capabilities vector B_c^{pc} by summing up the unused capabilities of the members of the coalition.
2. Form a list E_c of the expected outcomes of the tasks in T when coalition C performs them. For each task $t_j \in T$, perform:

(a) Check what capabilities B_j are necessary for the satisfaction of t_j.
(b) Compare B_j to the sum of the unused capabilities of the members of the coalition B_c^{pc}, thus finding the tasks that can be satisfied by coalition C.

3. Calculate the expected outcome of the tasks that can be performed by the coalition.
4. Among all of the expected outcomes, on list Ec, choose the maximal. This will be the coalitional value V_c.
5. Calculate the coalitional cost which is $c_c = 1/V_c$.

Third Stage of the Algorithm
In this stage, each agent calculates coalitional weights w, as the cost divided by coalitional size, the number of agents in that coalition. Each agent chooses the best coalition from among its list, i.e., the coalition Ci that has the smallest Wi. Next, each agent announces the coalitional weight that it has chosen, and the lowest among these is chosen by all agents. The members of the coalition that is chosen are deleted from the list of candidates for new coalitions. In addition, any possible coalition from the lists of any agent, that includes any of the deleted agents, is deleted from its list. These steps are iteratively repeated until all agents join a coalition or all tasks are assigned to a coalition. Each agent Ai iteratively performs the following steps:

1. Locate in L_i the coalition C_j with the smallest w_j.
2. Announce the coalitional weight w_j that it has located.
3. Choose the lowest among all of the announced coalitional weights. This w_{low} will be chosen by all agents. Choose corresponding coalition C_{low} and task t_{low} as well.
4. Delete the members of the chosen coalition C_{low} from the list of candidates.
5. If A_i is a member of the chosen coalition C_{low}, join its members and form C_{low}.
6. Delete from L_i the possible coalitions that include deleted agents.
7. Delete from T the chosen task t_{low}.
8. Assign to L_i^{cr} the coalitions in L_i for which values should be re-calculated.
9. Execute second stage of the algorithm.

4 System Architecture

Four different types of agents cooperate to provide an infrastructure with the functionalities listed in Section 1. Every type of agent is specialized in its actions, following well defined protocols. The agent types and their roles are as follows:

- ***employee agents*** which execute the coalition formation algorithm to form coalitions and, after the coalitions are determined, perform the tasks assigned to them.
- ***a controller agent*** which is in charge of system's overall coordination.
- ***coalition manager agents(s)*** which reduce the controller agent's heavy work load via coordinating the interaction and the communication between employee agents and the controller agent.
- ***a database agent*** which stores and maintains information associated with employee agents and tasks in a database, and meets query requests of all types of agents.

Fig. 1 depicts the architectural design of the framework, reflecting agent interactions during the coalition formation phase. Employee agents can communicate with each other and the controller agent. After a coalition is formed, this interaction slightly changes, as described in Section 4.3.

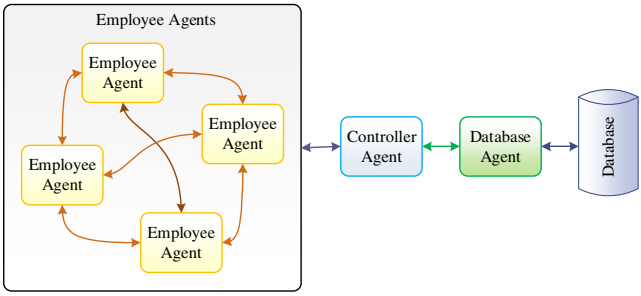

Fig. 1. System architecture displaying agent interaction during coalition formation phase

4.1 Database Agent

The database agent is the only agent in the system which has the capability of accessing the database. It accepts database requests, executes queries and delivers results. Only, the controller agent has direct communication with the database agent. If other agents need to query the database, they transmit their requests through the controller agent to database agent and receive results in the same way.

4.2 Controller Agent

The framework presents an execution environment where coalitions are formed in a distributed manner with no central control. However, the framework itself has a central management system, the controller agent being in charge of system's overall coordination. Central management usually becomes a bottleneck; therefore the control agent conveys some of its tasks to coalition manager agents as to decrease its work load. The controller agent monitors agents' status and task control period. When a predefined task control period ends, it restarts the coalition formation phase, thus eliminating the need to reset the system after new tasks have been defined.

Controller agent monitors the coalition formation phase. After coalition formation is completed, the controller agent starts a new coalition manager agent for each coalition, sends them the employee agents' identifications and task details. Next, the controller agent waits for messages from coalition manager agents. When a coalition manager reports, the controller agent prepares an execution report and sends it to database agent to be inserted into the database.

Furthermore, the controller agent is responsible for error handling. When an error message is received from a coalition manager agent, the controller agent updates the status of employee agents and also changes the task state to "failed" on the database. The failed task is scheduled on the next round of coalition formation phase. With this approach, the system becomes fault tolerant against agent failures.

4.3 Employee Agents

Employee agents execute the coalition formation algorithm to form coalitions and, after deciding on the coalitions, they perform the tasks which have been assigned to the coalition. They cannot be members of multiple coalitions at a time because of the disjoint coalition assumption. When activated, these agents firstly contact the controller agent and register themselves on the active agent list. Next, they wait for the arrival of information related to currently active agents and tasks from controller agent. After all employee agents receive the required information, they start executing the coalition formation algorithm and determine the coalitions they will join. Employee agents inform the controller about the result and wait until they are notified by the coalition manager agents assigned to their coalition. At this point in time, employee agents lose contact with the controller agent and carry on all further interactions with their manager agents. Fig 2 depicts agent interactions after coalitions are formed. Employee agents join their coalitions, execute their tasks and inform coalition manager agents that they have completed subtasks. If an error occurs during subtask execution, it is reported to the coalition manager agent.

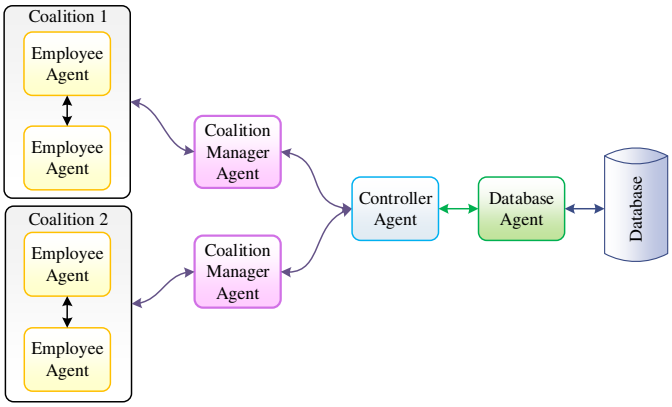

Fig. 2. Agent interaction after coalition formation

4.4 Coalition Manager Agents

For each coalition formed, a new coalition manager is created and remains in existence during the lifetime of the coalition. The coalition manager agent is representative of its coalition. It checks whether employee agents have completed subtasks; after a task is completed, the manager agent informs the controller and shuts down itself.

Coalition manager agent is also responsible for error checking, informing the controller agent about the error. The coalition manager also checks employee agents at predefined time periods to see if they are alive. If any employee agent does not respond in certain duration, the coalition manager assumes the agent has failed and informs the controller agent about the error. In both error cases, coalition manager stops execution of all subtasks and shuts down itself as the coalition is broken down.

5 Experiments and Evaluation

We have performed a comprehensive experimental analysis of the framework we have presented. The implementation is coded in Java 7 and runs on a notebook computer with 1.73 GHz Intel Core i7 processor, with 8.0 GB memory on a 64-bit Windows 7 Professional operating system. Experiments were performed on agents which were developed using JADE 4.2.0 version [8].

We tested our system approximately with 3000 runs to observe the effect of increasing values of agents, tasks and coalition sizes on coalition formation time. Test results reflect the time employee agents have consumed during the execution of the coalition formation algorithm and exclude the time taken by other activities, such as employee agent registration or task list sharing. We have applied the coalition formation algorithm to transportation problem and have carried out the test runs on the described environment over this application.

5.1 Effect of the Number of Employee Agents on Coalition Formation Time

In order to see the effect of varying numbers of employee agents on coalition formation time, we have carried out two sets of experiment, each with a different of tasks. Fig. 3 shows the results.

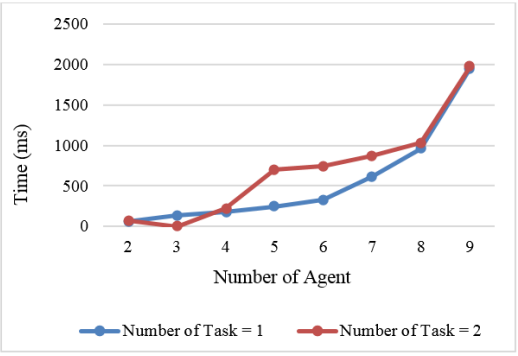

Fig. 3. Effect of number of employee agents on coalition time for different numbers of tasks

In both experiments, the maximum coalition size, k, was set to 2. The first set of experiments were executed with the number of tasks equal to 1 (blue graph), and the second with the number of tasks equal to 2 (red graph). In both cases, the number of employee agents is increased from 2 to 9 and the time taken for coalition formation is recorded. We have observed that increase in the number of employee agents directly effects coalition formation time. While the increase is acceptable up to 8 agents, further increases in the number of agents result in a steep rise. This is due to the heavy communication load between agents. We also observe that an increase in the number of tasks also effect coalition formation time, with larger values of tasks resulting in longer time periods, as seen in Figure 3.

Next, we have carried out experiments to observe how varying numbers of employee agents effect coalition formation time in systems with different maximum coalition sizes. The results are depicted in Fig. 4. In the experiments, the number of tasks was to 1. The experiments were executed for two maximum coalition size, one with *k* equal to 2 (blue graph), and another with *k* equal to 3 (green graph).

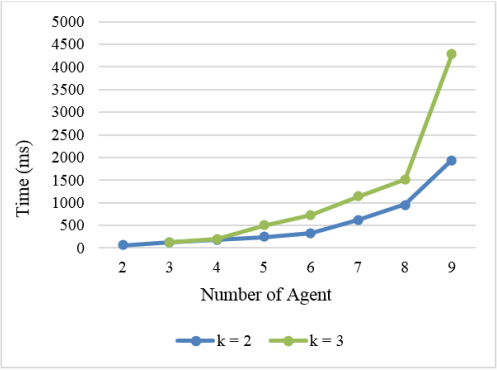

Fig. 4. Effect of number of employee agents on coalition time for different coalition sizes

We have observed that formation of larger coalitions take more time, compared to smaller coalitions. In conclusion, we can say that increasing the number of employee agents results in longer coalition formation time due to increase in the iterative calculations and inter agent communication.

5.2 Effect of Number of Tasks on Coalition Formation Time

We have also carried out experiments to observe how varying the number of tasks effects coalition formation time in systems with different numbers of employee agents. The results are depicted in Fig. 5.

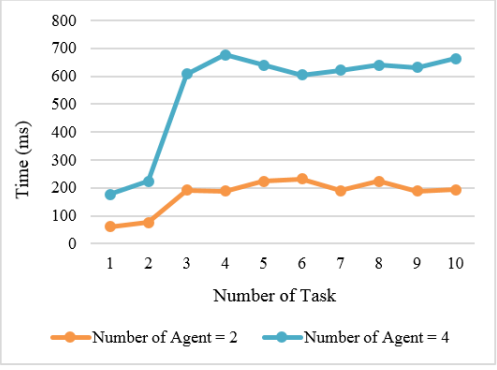

Fig. 5. Effect of number of tasks on coalition formation time

In these experiments, we have set the maximum coalition size, k, to 2. Two sets of experiments were executed, one with 2 (orange graph) and another with 4 (blue graph) employee agents. The results conform with the experiments; coalition formation time is directly affected by the number of employee agents and increasing the number of tasks does not create a significant difference in the time required.

6 Conclusions and Future Work

In this paper, we focus on a particular solution to the problem of task allocation through coalition formation. First, we shortly describe the Shehory-Kraus coalition formation algorithm, and next present a multi-agent based execution environment where employee agents can negotiate to form coalitions and execute tasks in accordance with the Shehory-Kraus algorithm. We describe the system architecture of the framework and discuss its various components in detail. For an assessment of the framework, we present experiments and report the effects of several factors, such as numbers of employee agents, tasks, and coalition sizes on coalition formation time.

Experimental results show that the framework fully provides the requirements for a negotiation environment for agents to form coalitions and execute tasks. The framework can be used as a test bed for further research as well. We plan to enhance the coalition formation algorithm by allowing agents to join multiple coalitions and adding precedence constraints to tasks as future work.

References

1. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence 101(1-2), 165–200 (1998)
2. Kraus, S., Shehory, O., Taase, G.: Coalition formation with uncertain heterogeneous information. In: Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 2003, Melbourne, Australia, July 14-18 (2003)
3. Kraus, S., Shehory, O., Taase, G.: The Advantages of Compromising in Coalition Formation with Incomplete Information. In: Proc. AAMAS, New York, July 19-23, pp. 588–595 (2004)
4. Ferber, J., Gutknecht, O., Michel, F.: From Agents to Organizations: an Organizational View of Multi-Agent Systems. In: Giorgini, P., Müller, J.P., Odell, J.J. (eds.) AOSE 2003. LNCS, vol. 2935, pp. 214–230. Springer, Heidelberg (2004)
5. Alfonso, B., Botti, V., Garrido, A., Giret, A.: A MAS-based Infrastructure for Negotiation and its Application to a Water-Right Market. In: Infrastructures and Tools for Multiagent Systems, Valencia, Spain (2012)
6. Pinto, T., Morais, H., Oliveira, P., Vale, Z., Praça, I., Ramos, C.: A new approach for multi-agent coalition formation and management in the scope of electricity markets. Energy 36(8), 5004–5015 (2011)
7. Ray, D., Vohra, R.: Coalition Formation. In: Young, P., Zamir, S. (eds.) Preliminary draft, prepared as a chapter for Handbook of Game Theory, vol. 4. North-Holland, The MIT Press, Cambridge, Massachusetts (2013)
8. JADE Home Page, `http://jade.tilab.com`