Efficient Task Allocation with Communication Delay Based on Reciprocal Teams

Ryoya Funato
Computer Science and Communications Engineering
Waseda University, Tokyo 1698555, Japan
r.funato@isl.cs.waseda.ac.jp

Toshiharu Sugawara
Computer Science and Communications Engineering
Waseda University, Tokyo 1698555, Japan
sugawara@waseda.jp

***Abstract*—This paper proposes a method to efficiently allocate tasks to appropriate agents by forming teams based on the reciprocity in distributed environments where communication delay is not ignorable. Recent applications on a variety of devices such as PCs, tablets, and smartphones run in different locations to provide location-oriented and time-constrained services. These services are usually realized by agents on these devices communicating with single or multiple service agents operating on servers that are also deployed at multiple points. Because timely response is a key factor for quality of services, communication delay is significant in these applications. Thus, we propose a method in which agents allocate tasks in such a widely distributed environment to reduce the delay of response to the requested tasks by extending our previous work. Then, we experimentally show that our method could improve the overall performance by identifying which agents have high-throughput of task execution from the local viewpoint.**

***Index Terms*—distributed task allocation, cooperative agents, reciprocity, reinforcement learning,**

I. Introduction

Recent applications operate in cooperation with a variety of mobile devices such as PCs, tablets, and smartphones with network connectivity at different geographical locations and provide location-oriented and time-constrained services [1], [2]. These services are usually realized by a combination of multiple specialized tasks done by a collaboration of agents, which are control programs running on these devices and on servers in the Internet. Because timely response of services is always a key factor for quality of service (QoS), these tasks should be allocated to agents that can do the specialized tasks in a timely manner.

However, it is not easy to allocate tasks to appropriate agents because the number of agents is quite large and they are designed by different developers and run on different servers in widely distributed places. For example, in fog computing [2], [3], many network devices for mobile devices are deployed near the user side and have limited resources and processing power. Thus, although the number of agents is restricted due to the limited processing resources, some agents reside on and use the resources of these devices to provide a for timely response by reducing the communication delay for location-oriented and real-time services. Therefore, how to allocate tasks to geographically distributed agents is an important issue for this type of network application.

In the multi-agent system literature, this kind of task allocation problem is often formalized using a general framework called the *team formation* in which a team (or task-oriented coalition) is a set of *cooperative agents* that have their own capabilities and that achieve the shared goal by doing the assigned jobs [4], [5]. A service in our domain is modeled by the execution of the corresponding task, which consists of a number of specialized subtasks that require certain capabilities; thus, they are allocated to the appropriate members in the team to be executed.

Along this line, Hayano et al. [6] proposed the efficient team formation methods for a large number of agents operating in a busy distributed environment. It proposed a method where agents autonomously identify which other agents they should work with on the basis of reciprocity that reflects the past interaction experience, and they form an implicit alliance structure in which agents preferentially form teams. Although this method can achieve an effective team formation process, its model does not include geographically distributed environments where delay of communications between agents is not ignorable.

Thus, we extend the previous model to include communication delay so that our extended model reflects how to select appropriate member agents to form a team in the distributed environment. Because the delay in communications between agents affects the performance, the selection of an agent that is highly capable but distant or one that is close but has relatively low processing power is difficult. We also analyze the experimental results to see whether our method can achieve efficient allocation and investigate whether reciprocal behavior can improve it.

II. Related Work

Because team formation is a fundamental problem in multi-agent systems, it is used to model many types of applications that require collaboration by forming teams. For example, Bakker and Klenke [9] proposed a distributed task allocation algorithm in multi-agent contexts for collaboration for the set of unmanned aircraft systems. Pujol-Gonzalez et al. [8] discussed the inter-team coordination method using the max-sum algorithm and applied it to form teams of agents for urban rescue operations in the RoboCup Rescue Simulation

challenge. There are also many applications using team formation in multi-agent systems in which agents are human (such as crowdsourcing [10], education [11], and social network analysis [12])) or are networked pieces of equipment (such as telecommunication network facilities [13] and sensor devices [14]).

Of course, team formation is applied to allocating tasks in distributed computer systems. Liemhetcharat and Veloso [7] introduced the concept of the synergy graph into multi-agent teams and proposed a team formation method using the learned synergy graph. Jiang et al. [15] proposed a notable task allocation model based on a negotiation reputation mechanism on the given agent networks that may contain deceptive agents. Although we assume that there are cooperative (so trustworthy) agents in our model, agents often encounter situations where they have to select actions that are unwanted for some collaborators due to unavoidable conflicts, but they must still keep the cooperative relationships. Hayano et al. [16] proposed a team formation in a distributed environment with a large number of agents in which multiple agents simultaneously attempt to form teams over time. Then, they extended their model to improve the success rate of team formation by switching between the rational and reciprocal strategies [6]. However, they ignored the communication delay that negatively affected the efficiency of team formation.

III. Problem Description

A. Agent, Task, and Communication Delay

Let $A = \{1, \ldots, n\}$ be the set of agents. Agent $\forall i \in A$ has the associated capabilities, $C_i = (c_i^1, \ldots, c_i^p)$, where $c_i^k \geq 0$ is a real number and p is the number of capability types. $c_i^k = 0$ indicates that i does not have the k-th capability. We define $p(i) = \#\{k \mid c_i^k > 0\}$. We also introduce the distance between agents $dist(i, j)$ and the required time (we assume a discrete time) for message sending. This is defined by $L(i, j) = \lceil dist(i, j)/D \rceil$ (seconds, the time unit used in this paper), where D is the positive constant called the *delay factor*. Note that $L(i, j) > 0$ if $i \neq j$.

For *workload* $\lambda > 0$, λ tasks on average are added to the tail of the system queue, $\mathcal{Q}$, every second. A *task*, $T = (S_T, U_T)$, consists of a set of subtasks, $S_T = \{s_1, \ldots, s_{l_T}\}$, and the utility $U_T \geq 0$. Task T is completed when all subtasks in S_T are completed. The subtask, s_j, has the required capabilities, $(r_{s_j}^1, \ldots, r_{s_j}^p)$, where $r_{s_j}^k \geq 0$ is the k-th capability required to perform s_j. If $\exists k$ s.t. $r_{s_j}^k > 0$ and $c_i^k = 0$, i cannot perform s_j; otherwise, i can perform s_j, and its execution time is

$$E^i(s_j) = \max_{1 \leq k \leq p} \lceil r_{s_j}^k / c_i^k \rceil, \tag{1}$$

where if $r_{s_j}^k = 0$ and $c_i^k = 0$, $r_{s_j}^k / c_i^k$ is defined as zero. For simplicity, we assume $1 \leq \exists! k_o \leq p, r_{s_j}^{k_0} > 0$, and $r_{s_j}^k = 0$ for $k \neq k_0$. We assume that an agent can do only one subtask at one time. Thus, a task should be done by a *team*, which is a temporarily formed set of agents with the required capabilities. When T is completed, the associated utility, U_T, is given. It is assumed to be proportional to the amount of the required capabilities, $U_T = \sum_{s \in S_T} u_s$, where $u_s = \sum_{k=1}^p r_s^k$.

B. Task Allocation in Agent Team

The team for task T is (G_T, σ_T), where σ_T is the one-to-one map from S_T onto G_T, and thereby $\sigma_T^{-1} : G_T \longrightarrow S_T$ can canonically be defined. $G_T (\subset A)$ is the set of agents that execute the subtasks assigned by σ_T. There are two types of agents, *leaders* (or *managers*) and *members*. A leader is an *initiator* of a team that attempts to collect the members and allocate subtasks to them. A member is an executor of the allocated subtask.

Agent i is in either an *inactive* or *active* state. i in the active state is involved in forming a team or executing a subtask; otherwise it is inactive. An inactive leader first picks up task T from the head of $\mathcal{Q}$ and becomes active. If i can find no task, it stays inactive for a second. It selects one subtask $s' \in S_T$ to do by itself if possible. Then, it internally selects N_d possible member agents for each $\forall s \in S'_T = S_T \setminus \{s'\}$, where $N_d > 0$ is an integer called the *solicitation redundancy*. How the member agents are selected is an important part of our proposed strategy and will be discussed in Section IV. The set of i and the selected agents with the allocated subtasks is called the *pre-team*, G_T^{pre}. Agent i then sends the agents in G_T^{pre} *solicitation messages* (SMs) with the subtasks that will be allocated to ask them to join the team to work with leader agent i, and then i waits for the responses. Note that any message arrives at the destination agent j in $L(i, j)$ seconds.

Leader i forms the team (G_T, σ_T) for T when it receives at least one acceptance for $\forall s \in S'_T$, where G_T is the set of agent i and the agents to which subtasks in S'_T are allocated. The assignment $\sigma_T : S_T \longrightarrow G_T$ is defined on the basis of the acceptances, which will be explained in Section IV. Leader i notifies the members in $G_T \setminus \{i\}$ of the successful team formation and sends regret messages to the agents that accepted the SMs but are allocated no subtasks. Agents in G_T execute the allocated subtasks. When subtask s' allocated to itself (leader i) is completed, it returns to the inactive state. While i can move on to another new task, it also receives the completion messages for the end of all subtasks. Thus, the leader manages which agents are still working. We define the execution time of T by G_T by

$$E^{G_T}(T) = \max_{s \in S_T} (E_{ldr}^i(\sigma_T(s), s))$$

where $E_{ldr}^i(j, s) = E^j(s) + 2L(i, j)$. After all subtasks have been completed, the members in G_T will receive utility according to the allocated subtasks.

On the other hand, if no agent accepts the solicitation for a certain subtask in S'_T, the team formation fails; thereby, i discards T and notifies the agents in G_T^{pre} of the failure. Then, i returns to the inactive state.

An inactive member agent j first checks the received SMs every second. If no messages arrive, it does nothing in this second. Otherwise, using the strategy for selection that is explained in Section IV, j selects one message and enters the

active state. Then, j sends an acceptance message to leader i, who is the sender of the selected SM, and sends the rejection messages to the other leaders. Then, j waits for a response from i. If j receives a regret message, it returns to the inactive state; otherwise, j joins the team and executes the allocated subtask. When j has completed the subtask, it notifies i of its completion, leaves the team, and returns to the inactive state.

Note that $N_d \geq 2$ means that the leader agent i selects pre-team members redundantly to avoid a situation where no agents accept a certain subtask. Thus, a larger N_d increases the success rate of team formation but may restrain some agents redundantly and increase communication cost.

IV. Proposed Method

A. Learning Team Agents via Past Collaboration

Our learning method is based on that proposed in Hayano et al. [6], but we found that just adding it with the model of communication delay did not work well. Thus, the proposed method is considerably revised it.

We introduce the learning parameter of *degree of dependability* (DE), where *dependability* is the reliability of another agent's decisions for cooperative behavior; thus, agents attempt to work with dependable agents again to increase the success rate of team formation. To decide the dependable agents, agent i has DE parameters $0 \leq d_{ij}$ whose initial value is 0.5 for $\forall j (\neq i) \in A$. The DE is used differently depending on the types of agents. Leader agent i, update the value of d_{ij} by

$$d_{ij} = (1 - \alpha_d) \cdot d_{ij} + \alpha_d \cdot \delta_d^L(G_T, j), \tag{2}$$

where $\delta_d^L(G_T, j) = 0$ if j rejects the SM, and $\delta_d^L(G_T, j) = u_{\sigma^{-1}(j)} / E_{ldr}^i(j, \sigma^{-1}(j))$ if the current team can be formed. $0 \leq \alpha_d \leq 1$ is the learning rate for the DE parameter.

A member j also updates d_{ji} depending on leader i's reply to j's acceptance message:

$$d_{ji} = (1 - \alpha_d) \cdot d_{ji} + \alpha_d \cdot \delta_d^M(s), \tag{3}$$

where $\delta_d^M(s) = u_s / (2L(i,j) + E^j(s))$ and s is the task allocated by i. However, if G_T cannot be formed or j is not included in G_T, $\delta_d^M(s) = 0$. Note that $2L(i,j) + E^j(s)$ is the binding time as a member of the current team.

The DE of i is gradually reduced every time to adapt to the environmental changes by

$$d_{ij} = \max(d_{ij} - \nu, 0) \text{ for } \forall j (\neq i) \in A \tag{4}$$

every second, where ν is a small positive number.

Dependable agents $j \in \mathcal{D}_i$ are defined as the set of the first X_F agents by sorting according to the descending order of DE values and $j \geq T_{\mathcal{D}}^i$, where $T_{\mathcal{D}}^i$ is a threshold number and $X_F > 0$ is a positive integer. When $\mathcal{D}_i \neq \emptyset$, i enters the *reciprocal mode* and change the behavior to keep the current dependability relationship. Note that, from Formula (3), the DE indicates the distance and the extent to which the capabilities of members are effectively used. Since member agent j knows its own capability, we set $T_{\mathcal{D}}^j = T_{M,\mathcal{D}}^j \times \sum_{c \in C_j} c/p(j)$ so that it is proportional to the average of their capabilities. On the other hand, the leader's performance is mainly decided by the capabilities of its members, which is unforeseeable, so we define $T_{\mathcal{D}}^i$ as a positive constant $T_{L,\mathcal{D}}$.

TABLE I
Parameter values used in experiments.

Description	Parameter and value
Types of capabilities	$p = 3$
Workload	$\lambda = 2.5$ to 10
Queue length	$\lvert\mathcal{Q}\rvert = 500$
Number of redundant messages	$N_d = 2$
Learning rate for DE	$\alpha_d = 0.01$
Reduction value	$\nu = 2.0 \times 10^{-6}$
Threshold of dependability in leaders	$T_{L,\mathcal{D}} = 1.5$
Threshold of dependability in members	$T_{M,\mathcal{D}} = 0.5$
Greedy value in the ε-greedy strategy	$\varepsilon = 0.05$

B. Reciprocal Behavior and Strategies for Leader/Member selections

Agent i using our proposed method behaves as follows. It looks for collaborative agents in a rational manner at first, i.e., leader agent i selects member agents according to which agents will likely accept the SMs and will bring more utility per second. Member j selects the SMs according to which leader will put j into G_T with a higher probability and bring a higher utility per second. By definition, these decision are made using the DE values in both cases.

Member agent j in the reciprocal mode selects only the SM that is sent from the leader in $\mathcal{D}_i$ and ignores other messages. If j received multiple SMs from dependable agents, it selects the SM sent from leader whose DE value is the highest. This behavior seems non-rational from the viewpoint of pursuing utility because j might be able to receive some utility from non-dependable leaders. However, if j accepted it, it might not be able to accept a SM from a dependable agent arriving during the execution. Because dependability is the result of past consecutive instances of mutual cooperative behavior, by giving priority to dependable agents, we can enhance the reciprocal structure.

When leader agent i selects the members of the pre-team, it first chooses from $\mathcal{D}_i$ and then from $A \setminus \mathcal{D}_i$ based on the DE values (with the ε-greedy strategy) as follows. Initially, i sets $G_T^{pre} = \emptyset$ and sorts $A' = A \setminus \{i\}$ by descending order of DE values, and $\tilde{S}_T = S'_T$. $\tilde{S}_T$ is also sorted by descending order of its utility u_s. Agent i selects the first element s in $\tilde{S}_T$ and search member j from the top of A' that can execute the task. Then, (j, s) is added to G_T^{pre}. If $j \in \mathcal{D}$, i removes s from $\tilde{S}$. i iterates this selection for S'_T. If $N_d = 1$, the current G_T^{pre} is the pre-team.

If $N_d > 1$, the same process repeats another $N_d - 1$ times. Note that since the subtasks allocated to dependable agents are removed from $\tilde{S}_T$, they are allocated to only one agent by relying on their reciprocal behavior. Note that all agents also adopt the ε-greedy strategy when selecting SMs and the members of a pre-team.

V. Experimental Evaluation and Discussion

A. Experimental Setting

We experimentally evaluated our method by investigating the numbers of successfully formed teams and comparing them with those of the model based on the traditional *contract net protocol* (CNP) [17] and those of rational agents who only pursue the values of utility per second. Leaders of the rational agents in this experiment attempt to increase the success rate of forming teams; thus, they learn which member agents are likely to accept the solicitation of joining teams. For this purpose, they have the parameters, like the DE, whose values are updated using Formula 2, but $\delta_d^L(G_T, j) = 1$ if j accepted the SM, and $\delta_d^L(G_T, j) = 0$ if j did not. On the other hand, members (contractors) try to maximize the utility per second; thus, they also use Formula (3), but it is updated by $\delta_d^M(s) = u_s/E^j(s)$. They do not have the reciprocal mode. In the CNP-based protocol, the leaders (managers) announce all subtasks to the nearest one hundred members. Then, members bid $(s, E^j(s))$, where s is the subtask whose utility per second $u_s/E^j(s)$ is the largest. The leader selects the best members.

$|A_L| = 100$ and $|A_M| = 400$, and these agents are placed on the 50×50 grid environment randomly. We use the Manhattan distance between agents. For $\forall i \in A$, its capability $c_i^k \in C_i$ is set to a non-negative integer randomly selected from $[0, 5]$, and if $p(i) = 0$, i's capabilities are redefined. Task, T, consists of three to six subtasks, $s_j \in S_T$, each of which has only one positively required capability $r_{s_j}^{k_0} > 0$ whose value is also set randomly between five and ten. Other parameters used in the proposed method are listed in Table I. The upper limit of dependable agents is $X_F = 1$ for member agents and $X_F = \infty$ for leaders. The experimental results shown below are the average of 20 independent experimental runs with different random seeds.

B. Performance Comparison

Figure 1 shows the number of completed tasks, which is identical to that of successfully formed teams, per 100 seconds, when $\lambda = 2.5$ (system is under low workload), $\lambda = 5$ (moderate workload), and $\lambda = 7.5$ (a few tasks have overflowed). We also plotted the average communication time (one-way), $L(i, j)$, and the average execution time of subtasks, $E^i(s)$, when $\lambda = 5.0$ in Fig. 2. Note that we did not indicate on the graphs when λ is other values because these values were quite similar to those of Fig. 2.

First, we cannot see the obvious difference in the number of proposed agents and rational agents when $\lambda = 2.5$ from Fig. 1 (a). However, in other cases, the agents using the proposed method outperformed the rational agents. Particularly, when the agents were rational, the number of complete tasks hit the ceiling, which was approximately four hundred per 100 seconds as shown in Fig. 1 (b) and (c). In an environment where all agents are rational, leader agents pursue improving the success rate, and members focus on the earned utility; thus, they can increased the number of completed tasks. However, because they do not care about the communication delay as shown in Fig. 2(a), they could execute only a limited number of tasks. On the other hand, the agents using the proposed method could decrease both communication time (Fig. 2(a)) and execution time (Fig. 2(b)) over time; thus, they could complete more tasks than rational agents.

The CNP agents exhibited relatively low performance in all cases, although the average values of communication delay and execution time are much better than those of other agents (see Fig. 2 (a) and (b)). We think that this is caused by conflicts in selecting messages in members and possible members in leaders. Even when the range of communication was restricted (so the communication time was relatively small), the same announcement messages were received by multiple members. If they had a similar capability structure, they selected the same member and made bids to the corresponding leaders. Any leaders can select only the best agents (so the execution time was very small) for each subtask, and other members could not join the team. This lowered the success rate of team formation.

C. Remarks

Figure 1 also indicates that a number of tasks could not be executed due to the failures of team formation, even when λ was not large. Because agents use the ε-greedy strategy, it is probable that they select members/a leader randomly with the probability of ε, and they are likely to fail when forming teams. In addition, because the environment is the grid plane, the agents near corners and fringes had fewer close leaders and member agents, so they could not fully exert their capabilities. To improve the success rate in real applications, tasks that could not be executed were returned to the system's queue. However, we did not do this in our experiment because we want to know the pure success rate of team formation.

In our experiment, we fixed the number of leaders to 100, but this number may change depending on the structure of tasks. Hayano et al. [6] proposed autonomous role selection so that they can adaptively decide their role: leaders or members. However, we could not add this decision to our current model. We also define a number of threshold parameters. Therefore, we plan to identify their values based on their contribution to task execution; these issues will be addressed in our future work.

VI. Conclusion

Recent applications cooperate with a variety of mobile devices network connectivity at a variety of locations to provide location-oriented and real-time services. These services are usually realized by a combination of different programs with specialized capabilities/functionalities for the required tasks. In networking environments, cooperation between agents is always affected by the time required for communication. This paper proposes a method to efficiently allocate tasks to appropriate agents by forming teams based on the reciprocity in the distributed environments where communication delay is not ignorable for timely services. The proposed method was experimentally evaluated and found to have good performance

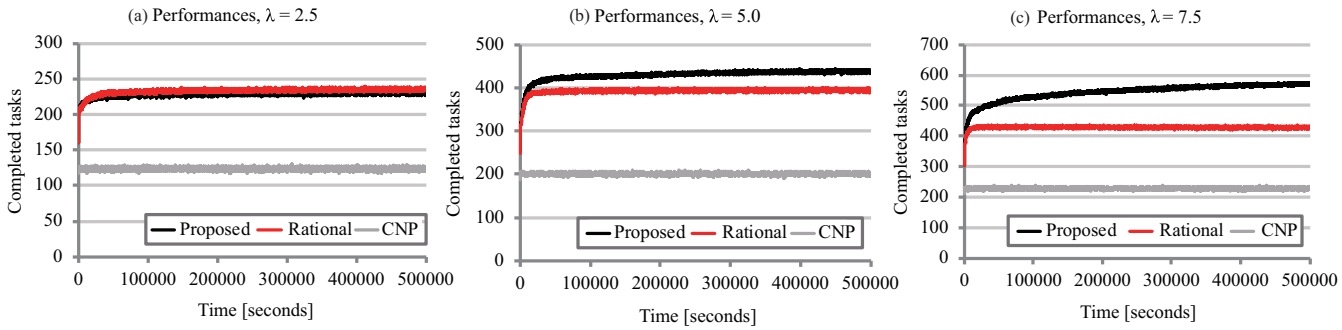

Fig. 1. Number of Executed Tasks.

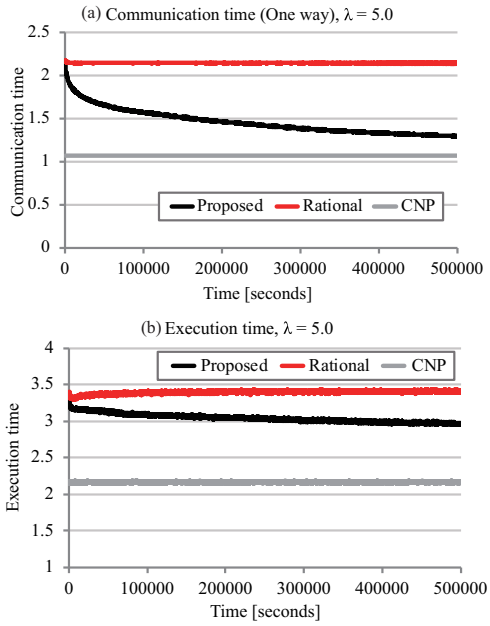

Fig. 2. Communication Time and Execution Time.

by selecting local agents and/or distant agents with high capabilities for their collaboration. We plan to learn the values of parameters/thresholds used in our model so that they can be used in a wide range of services in networked and distributed environments.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[2] S. Chen, T. Zhang, and W. Shi, "Fog Computing," *IEEE Internet Computing*, vol. 21, no. 2, pp. 4–6, Mar 2017.

[3] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *2015 Third IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb)*, Nov 2015, pp. 73–78.

[4] B. Horling and V. Lesser, "A Survey of Multi-agent Organizational Paradigms," *Knowledge Engineering Review*, vol. 19, no. 4, pp. 281–316, Dec. 2004.

[5] B. M. Dunin-Keplicz and R. Verbrugge, *Teamwork in Multi-Agent Systems: A Formal Approach*, 1st ed. Wiley Publishing, 2010.

[6] M. Hayano, N. Iijima, and T. Sugawara, "Asynchronous Agent Teams for Collaborative Tasks Based on Bottom-Up Alliance Formation and Adaptive Behavioral Strategies," in *IEEE 15th Int. Conf. on Dependable, Autonomic and Secure Computing*, Nov 2017, pp. 589–596.

[7] S. Liemhetcharat and M. Veloso, "Modeling and learning synergy for team formation with heterogeneous agents," in *Proc. of the 11th Int. Conf. on Autonomous Agents and Multiagent Systems - Volume 1*, Richland, SC: IFAAMAS, 2012, pp. 365–374.

[8] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, and J. A. Rodriguez-Aguilar, "Efficient inter-team task allocation in robocup rescue," in *Proc. of the 2015 Int. Conf. on Autonomous Agents and Multiagent Systems*, Richland, SC: IFAAMAS, 2015, pp. 413–421.

[9] T. Bakker and R. H. Klenke, *Dynamic Multi-Task Allocation for Collaborative Unmanned Aircraft Systems*. American Institute of Aeronautics and Astronautics, 2014.

[10] W. Wang, J. Jiang, B. An, Y. Jiang, and B. Chen, "Toward Efficient Team Formation for Crowdsourcing in Noncooperative Social Networks," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4208–4222, Dec 2017.

[11] E. del Val, J. M. Alberola, V. Sanchez-Anguix, A. Palomares, and M. D. Teruel, " Team Formation Tool for Educational Environments," in *Trends in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, Springer Int. Publishing, 2014, pp. 173–181.

[12] A. Farasat and A. G. Nikolaev, "Social structure optimization in team formation," *Computers & Operations Research*, vol. 74, pp. 127 – 142, 2016.

[13] A. Terauchi, O. Akashi, M. Maruyama, K. Fukuda, T. Sugawara, T. Hirotsu, and S. Kurihara, "ARTISTE: Agent Organization Management System for Multi-Agent Systems," in *Multi-Agent Systems for Society* (PRIMA 2005), LNCS Vol. 4078, Springer Berlin Heidelberg, 2009, pp. 207–221.

[14] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, "Distributed Search and Rescue with Robot and Sensor Teams," in *Field and Service Robotics: Recent Advances in Reserch and Applications*, Springer Berlin Heidelberg, 2006, pp. 529–538.

[15] Y. Jiang, Y. Zhou, and W. Wang, "Task Allocation for Undependable Multiagent Systems in Social Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 8, pp. 1671–1681, Aug 2013.

[16] M. Hayano, D. Hamada, and T. Sugawara, "Role and member selection in team formation using resource estimation for large-scale multi-agent systems," *Neurocomputing*, vol. 146, no. 0, pp. 164 – 172, December 2014.

[17] R. G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, pp. 1104–1113, 1980.