

# Communication-Efficient Dynamic Task Scheduling for Heterogeneous Multi-Robot Systems

Kashyap Shah and Yan Meng, *Member, IEEE*

**Abstract**— In this paper, a communication-efficient dynamic task scheduling algorithm for a heterogeneous multi-robot system is proposed. To make this task scheduling algorithm to be scalable for various robot teams, a distributed communication with shared global unit mechanism is applied to reduce the storage cost as well as communication overhead. Each robot makes its own decision through communicating with others as well as checking a global unit. This algorithm improves its efficiency by broadcasting specific information only to those who are capable and have interest in the current tasks. To improve the system robustness, an auction-based fitness function is applied to dynamically allocate the tasks among the robots if the pre-assigned robot can not handle the task or a robot fails under dynamic environment. The proposed approach is robust against communication failures and robot failures. Simulation results demonstrate the efficiency and robustness of the proposed approach.

## I. INTRODUCTION

Multi robot systems have been used for various real-world applications, such as urban search and rescue, surveillance, hazardous materials detection, and reconnaissance. An efficient task allocation and coordination among the team members is required to achieve real-time performance of such complex real-time systems. It is stated in [1] that team performance can be drastically increased if the team coordinates well and the information is being shared by all teammates in a multi-robot environment. However, dynamic task allocation for multi robot systems under dynamic environment is a challenging problem, which aims to efficiently finish all of tasks as fast as possible with the minimum cost. This is a NP-hard problem, where there is no deterministic method to find an optimized solution. Most available task scheduling methods have extensive broadcast communication overhead to share information with all of team members for both centralized methods and distributed methods. Some of available algorithms are only good for a homogeneous robot team with one global task like mapping or exploration of an area, and some of them are very vulnerable with communication failure or robot malfunctions.

In this paper, we aim at investigating a communication-efficient task scheduling algorithm for a heterogeneous

multi-robot system under dynamic environment. Instead of using either pure centralized approach, which is usually efficient with small number of agents. However, the centralized system is difficult to scale to large number of agents without losing its performance significantly. On the other hand, the decentralized approach can be scalable since each robot makes its own decisions for a particular set of tasks. No central unit is needed. Some initial decomposition of the global scheduling may be imposed and robots can negotiate with others to make the best of coordination and solve conflicts dynamically. Furthermore, error handling and system recovery are critical issues for system robustness.

Based on the above observations, we propose a communication-efficient multi-robot (CEMR) task scheduling algorithm, where there is a shared global unit for all of the robots to access while each robot makes its own decision through communicating with others. This CEMR task scheduling algorithm avoids unnecessary communication by broadcasting global information to everybody. By sending specific task information only to those which have the capability, the communication overhead can be reduced significantly. However, without the global information regarding the task status, the robots may conduct some tasks which is being carried out or has been carried out by other robots. To reduce this conflict and redundancy, a global unit is necessary where each robot can access this memory to update and check the current task status. Another advantage of this global shared global unit mechanism is that when the robot team becomes bigger, the corresponding storage requirement will be increased only in this shared global unit, instead of growing exponentially with the size of the team if each robot has to keep a global task status in its own memory. Therefore, the proposed algorithm takes advantage of centralized approaches to improve the overall efficiency and distributed approaches to reduce the communication overhead.

Another major advantage of the proposed CEMR task scheduling method lies in its robustness to adapt to dynamic environment. Two dynamic cases are considered here. First, some robots may not be able to handle the assigned task anymore since the task environment has been changed. Second, one or more robots may fail under dynamic environment.

In the first case, instead of making each robot to adapt to new tasks or dynamic environment, which sometimes would be difficult or even impossible in a heterogeneous system, the deficient robot checks the global unit and sends help

Kashyap Shah is a graduate student in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA. (e-mail: kshah@stevens.edu).

Yan Meng is with the Department of Electrical and Computer Engineering Department, Stevens Institute of Technology, Hoboken, NJ 07030, USA. (Corresponding author: phone: 201-216-5496; fax: 201-216-8246, e-mail: yan.meng@stevens.edu).

signals to those who can handle the task. In the second case, by tracking the communication signal, the failed robot will be detected, and the global robot status would be updated with this failure so that no more tasks will be allocated to the failed robots. This feature makes the system robust against robot failures.

The paper is organized as follows. Section II introduces the related work in the field of task allocation algorithms for multi-robot systems. Section III describes the problem statement. Section IV proposes a dynamic task allocation algorithm for heterogeneous multi-robot systems. Extensive simulation results are discussed in Section V. The paper is concluded by Section VI.

## II. RELATED WORK

Increasing amounts of research have been conducted in the area of dynamic task scheduling for multi-robot systems. There are many auction based methods available for handling dynamic task allocation and MURDOCH [2][3] is a popular one among them, which uses contract net protocol as its communication protocol. Generally, distributed systems rely on fitness based actions and negotiation protocols. MURDOCH uses publish/subscribe messaging for distributed control of multi-robot systems. Instead of assigning a task to an individual robot, the system allows the user to pose tasks to a group of team members. This method is also called subject based addressing because robots never communicate with each other directly but they communicate through the subject which contains the resource requirements for a publisher robot. Once publisher robot publishes the subject, it will be up to consumers whether to respond or not. The consumer robot will respond to the published message if it is subscribed for any services which can help publisher robot.

ALLIANCE [4], which was proposed by Parker, is a fully distributed, behavior-based software architecture which gives all robots the capability to determine their own actions based upon their current situation. ALLIANCE utilizes a simple form of broadcast communication to allow robots to inform other team members of their current activities, rather than relying totally on sensing through the world. Thus, at some pre-specified rate, each robot broadcasts a statement of its current action, where other robots may listen to or ignore as they wish. L-ALLIANCE [5] is an extension to ALLIANCE. L-ALLIANCE has all the advantages like fault-tolerance and adaptive characteristics of behavior-based systems as well as robust task allocation. This method is very complex and only effective in small and medium size teams of heterogeneous robots. It requires extensive communication, a great deal of time for constant monitoring of teammates and taking accurate decisions.

One of the easiest approaches to work in dynamic task assignment is trial and error method, where all robots will try the same task one by one until the perfect match is found. This method is very inefficient and time consuming. James McLurkin [6] proposed three different methods of task

assignment in robot swarms. First method is random-choice algorithm, which selects tasks randomly, but runs in constant time. This method is extremely communication extensive and inefficient. Second method is card-dealer's algorithm, which assigns tasks to individual robots sequentially, using minimal communications but a great deal of time. Third method is tree-recolor algorithm, which is a compromise between extreme-communication and card-dealer's, balancing communications use and running time. This algorithm is robust and only efficient for homogeneous multi-robot systems.

Stroupe et al. [7] proposed a behavior-based planning algorithm for multi-robot systems, where each robot predicts the behavior of its companion and proceeds for further steps. The main idea of this method is that the robot should not try to adapt to the situation but instead should directly transfer that task to an associated robot who can handle that situation. Brumitt and Stentz [8] proposed a dynamic mission planning for multi-robot systems in a dynamic environment, where the planning system dynamically reassigns robots to goals in order to continually minimize the time to complete the mission. According to robots activity and their behavior against different tasks, the mission planner calculates optimum solution for global goals and reassigns robots to different tasks. Trade-offs between robot's traveling cost and running cost of mission planner has to be balanced.

Davis and Smith [9] proposed a contract net protocol, where the collection of nodes (robot) can be represented as a contract net. Each node in the net may take the role of a manager or a contractor. A manager is responsible for monitoring the execution of a task and processing the results of its execution, while a contractor is responsible for the actual execution of the task. Whenever any node encounters a task which it cannot perform, it will send task announcement signal for help. Based on the responses, the node who sends announcement signal will grant a node for that task. This contract net protocol method did not provide robustness against message lost.

A task-assignment architecture was proposed in [10] for cooperative transport by multiple mobile robots in an unknown static environment, which consists of two real-time planners: a priority-based task-assignment planner and motion planners based on short-time estimate. This method is also compared with Stillwell's algorithm in [11], where homogenous robots are ant like objects who try to move a big piece of food from one place to their nest. Each of them tries to contribute in the most efficient way.

The scheduler proposed in [12] is one example of greedy decentralized schedulers. Generally, these kinds of cooperative search approaches are efficient and robust in applications like military scouting and automatic trash collection. A novel emotion-based recruitment approach was proposed in [13] for a multi-robot task allocation problem. This affective recruitment is tolerant of unreliable communication channels, and can find better solutions than

simple greedy schedulers.

### III. PROBLEM STATEMENT

In order to develop a dynamic task scheduling algorithm for a heterogeneous multi-robot system, especially under dynamic environment, first we need to setup some working environments as well as the assumptions within the environments. Some simplified task environments are shown in Fig. 1, which is divided into 3 different subareas. Different types of robots are defined based on their capability to work under these subareas. For example, *type-I robot*, which is only capable of working in subarea 1, but not the other two subareas. Similar definitions are applied to *type-II robot* and *type-III robot*. Some robots may have capabilities suitable for multiple sub-areas, such as *type-I-II* or *typeII-III*. In order to implement the assigned missions in a more efficient manner, task preemption is not allowed among the robots in the proposed task allocation approach. In other words, if a robot is implementing the task, we don't need another robot to do it even if the other one has higher fitness value.

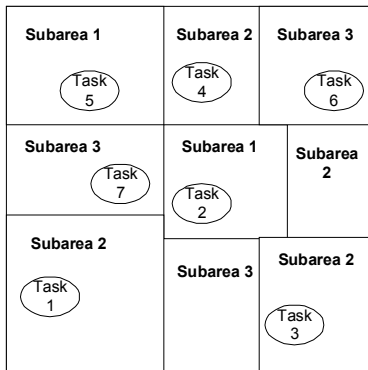


Fig. 1. Heterogeneous multi-robot task environments

Consider that we have  $N$  heterogeneous robots and  $M$  different tasks randomly distributed in different subareas. Here task is a conceptual terminology, which can be defined as various physical jobs, such as trash can collection, demining, transportation, construction, or assembling. It is assumed that robots are able to localize themselves within the environment, can avoid obstacles and plan path to a destination. It is also assumed that each robot has special onboard sensors to detect tasks and the associated environment. The robots are expected to move to the position where the tasks are located and complete the task. It is assumed that robots have no idea in which kind of environment they are going to work. Initially, some predefined tasks are stored in each robot. The requirement of these tasks may be changed due to dynamic environment. Therefore, dynamically reallocating the tasks is necessary. The objective of this project is to develop an efficient task scheduling algorithm for a heterogeneous multi-robot system under dynamic environment so that all of the tasks can be completed as soon as possible meanwhile cost can be reduced as low as possible.

## IV. THE APPROACH

### A. The Architecture of the Approach

As stated in last section, different tasks need different types of robots with associated capability to implement, and the challenge is that how to schedule these tasks in an efficient manner that only the right robot goes to the associated task and finish all tasks as soon as possible while keeping the cost as low as possible. To tackle this problem, a communication efficient multi-robot (CEMR) task scheduler is proposed, where each robot makes its own decision and communicates with others, and meanwhile all of robots can access a global unit to check robot/task status to improve the coordination efficiency. The architecture of this CEMR task scheduler is shown in Fig. 2.

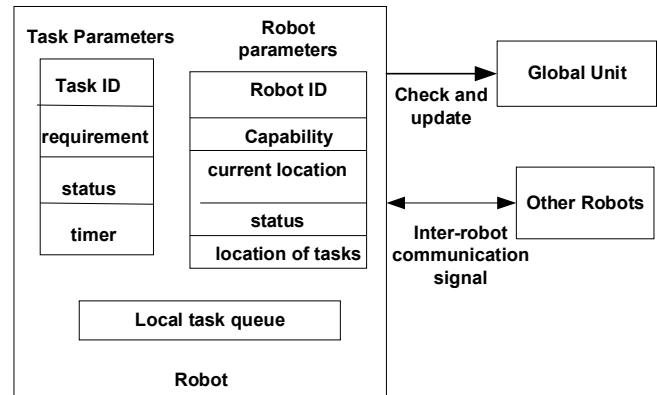


Fig. 2. The architecture of the CEMR task scheduler

Each robot has a local database to keep all the information it required to make decisions. This database structure, as shown in Fig. 2, includes three major parts: robot parameters, task parameters, and local task queues. Robot parameters consist of robot ID, capability vectors of all robots, current locations of all robots relative to a reference coordinate system, status of all robots, locations of all tasks. Task parameters consist of task ID, task requirement, task status, and task timer. Local task queue stores the tasks that the robot needs to complete sequentially. Global memory stores the current task status as well as the robot status so that each robot can check it before making its own decision.

### B. The System Parameters

1) *Task ID*. When a task is assigned to a robot, the robot needs to know which task it will conduct. To easily identify the task in the environment, here, we use the physical locations of the tasks relative to a 2D global coordinate system.

2) *Robot ID*. Robot ID is a unique identification number for each robot and capability is robot's ability to perform a task, which is a combination of different subareas represented by a capability vector, as shown in Fig. 3.

	R(S1,1)	R(S1,2)	R(S2,1)	R(S2,2)	R(S3,1)	R(S3,2)
S1	1	1	0	0	0	0
S2	0	0	1	1	0	0
S3	0	0	0	0	1	1

capability vector for type-I robot R(S1,1)

Fig.3: Robot IDs and their capability vectors

3) *Robot status*. Robot status consists of free (its local task queue is empty), busy (some tasks are in its local task queue), or failed.

4) *Task status*. Task status shows whether the task is in progress, completed, in-trouble, or time-out.

5) *Task timer*. Task timer is used to track how long the task has been processed. To prevent the system to be hanged by one task forever, if the processing time is greater than a predefined threshold, time-out status would be labeled on the task.

6) *Local task queue*. Local task queue keeps a list of tasks a robot will perform sequentially. These tasks may be some predefined tasks before the system starts, or tasks detected or reassigned on the fly. Once the robot finishes its first task in its task queue, it would remove the finished task and go to next one until the last task in the queue. Once this queue becomes empty, robot will start moving randomly to search for a new task.

7) *Global unit*. In the global memory, task status and the robot status are stored. The main purpose of this global memory is to reduce the unnecessary redundancy among robots to process the same task so that the system efficiency can be improved. When a robot performs work which affects overall team work, it would broadcast the information on the global unit instead of checking their own database which may not fully cover information of other robots in distributed approaches. Another advantage of using this global unit is that the required memory will not be increased exponentially with a large scale of robot team. Only one copy of the status data need to be stored in the global unit.

### C. Task Allocation under Dynamic Environment

To adapt to dynamic environment, multi-robot system has to be robust enough to handle different emergent situations. Here, two cases need to be considered. First, some robots may not be able to handle the assigned task anymore since the task environment has been changed. Second, one or more robots may fail under dynamic environment.

In the first case, instead of making each robot to adapt to new tasks or dynamic environment, which sometimes would be difficult or even impossible in a heterogeneous system, the deficient robot checks the global unit and sends help signals to those who can handle the task. For example, the environment of task 1 has been changed from type-I to type-II, the type-I robots can not handle task 1 anymore. If a

robot finds out that it is difficult for it to process a task in its local queue during execution, it will send help request to those robots whose capabilities match the task requirement. If help responses are received, the robot will assign the task to the responded robot, and delete it from its local queue. Meanwhile the responded robot would add that task in its local queue.

In the second case, by tracking the communication signal, the failed robot will be detected, and the global robot status would be updated with this failure. There are two situations for robot failure. First, a robot fails when it is in idle status (i.e. not processing any task). In this case, the robot status will be updated as failed and no more tasks will be allocated to this robot in the future. Second, a robot fails during its execution of a task. In this case, the task status may be time-out and another robot has to be assigned to the unfinished task.

If multiple robots respond to the help requesting signal, an auction-based method is applied to select the best candidate from those responded. The fitness function of this auction-based method is defined as follows:

$$F = k \frac{f(c_i | t_j) f(n_i) f(a_i)}{d_i}, i = 1, 2, \dots, N, j = 1, 2, \dots, M. \quad (1)$$

Where  $c_i, d_i, a_i$ , and  $n_i$  represent the capability, distance from the current task location, robot availability, and number of tasks in local queue of robot  $i$ , respectively.  $t_j$  represents the task types, and  $k$  is a scale factor.  $f(c_i | t_j)$  is the matching function of the capability of robot  $i$  related to the type of task  $j$ , which is defined as follows:

$$f(c_i | t_j) = \begin{cases} 1, & \text{if the robot capability matches the task type} \\ 0, & \text{otherwise} \end{cases}$$

$f(n_i)$  and is defined as follows:

$$f(n_i) = \begin{cases} 1, & \text{if } n = 0 \\ 1/n, & \text{otherwise} \end{cases}, \text{ and}$$

In other words, if all of the responding robots are busy, the numbers of tasks in their local task queues are compared. The smaller the task number in queue, the higher the possibility of the corresponding robot would be selected as the helper. And  $f(a_i)$  is defined as

$$f(a_i) = \begin{cases} 1, & \text{if robot is working normally} \\ 0, & \text{if robot fails} \end{cases}$$

### D. Inter-Robot Communication Protocols

Since robots need share task information with others, a specific communicate protocol is required for this application. Basically, four types of signal frames are defined in the communication protocol, (1) help request

frame; (2) help respond frame; (3) help accept frame; and (4) global unit update frame. The detailed frame definitions are shown in Fig. 4. When a help seeking robot receives help responding signal, it would send help accepting signal back to the selected robot. If responder robot is busy at that time, it would add that task in its local task queue and continue working on its current task. Global task update frame is used to update the global unit.

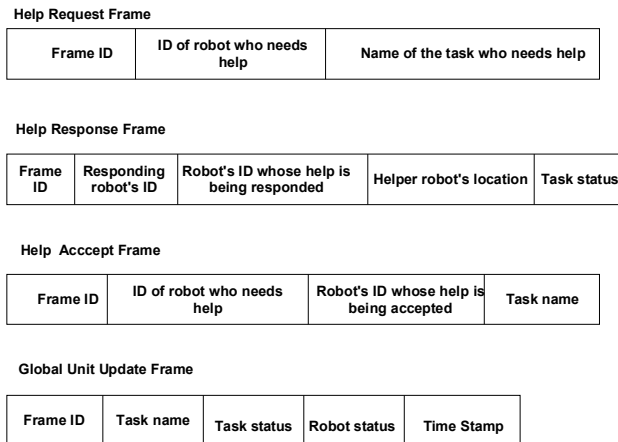


Fig.4. Communication protocols

### E. Approach Summary

Initially, all robots move around randomly in the environment searching for tasks. When a robot detects a task, it checks the requirements of the task first. If the requirements of the task match with its own capability, then the robot would perform the task and inform others about its initialization of task. Each robot updates the status of its work whenever it changes from one state to another. The states include beginning, completion, in-trouble, and time-out. When a robot completes a task, it would update the task status in the global unit of its completion of task. Whenever it needs help, it first checks the global unit to see who are capable and available for the task. Then the robot sends help request signals to those robots that would actually be able to provide the service that task requested instead of broadcasting to everyone. Here a trade-off between memory capacity of robots and communication overhead among robots has to be made. The proposed mechanism reduces the memory requirement while keeping the communication overhead as low as possible. The system stops when the global task status queue is filled up with all tasks with status of completion.

## V. SIMULATION RESULTS

To evaluate the proposed algorithm, a simulator is developed using C/C++ language under Windows environment. Some proof-of-concept simulations have been conducted. Six robots are employed in the simulation including 2 type-I (represented as R(S1,1) and R(S1,2)), 2 type-II (R(S2,1) and R(S2,2)), and 2 Type-III (R(S3,1) and R(S3,2)). Eight tasks are generated, which are represented

by the location coordinate within a reference frame. T1(X1,Y1), T2(X2,Y2) and T3(X3,Y3) are in subarea 1, T4(X4,Y4), T5(X5,Y5) and T6(X6,Y6) in subarea 2, and T7(X7,Y7) and T8(X8,Y8) in subarea 3. Each robot is capable of localizing itself within the environment and can avoid obstacles and plan path to a task destination.

Most task allocation problems among robots have applied broadcast communication to share the information and negotiate with team members. This kind of broadcast communication usually requires more communication overhead, power, time, and cost, especially for a heterogeneous team where only some specific members can conduct some types of tasks, not all of them. In our algorithm, instead of broadcast to everyone, the help information is only broadcasted to those which have the capability for the task. The communication cost comparison results are shown in Table 1. Communication overhead is directly proportional to task processing time and power consumption. In other words, our approach would be more power efficient and spend less time to finish the tasks than the broadcasting method.

TABLE I  
SIMULATION RESULTS

	Communication Cost (messages)	Processing Time (iteration time)
Proposed Approach	31	689
Broadcast approach	58	803

To evaluate the performance of the proposed method, a simple auction-based method using broadcasting is applied for comparison, where robots randomly search for tasks and broadcast the task information to all team members. If one robot needs help, and it receives multiple responses, the robot which is closest to the task will be selected. Four cases of different task distributions are designed in the simulation, where 8 task locations are re-distributed in the environment at different cases. The time required to finish all tasks under different task configuration cases are recorded and shown in Fig. 5. The proposed algorithm obviously outperforms the random searching one. There are two reasons for this advantage. First, in our approach, we select the helper robot not only depends on its current distance to the task, but also its current status.

To evaluate the robustness of the proposed algorithm under the failure situations, such as communication failure and robot failure, another set of simulation results with the same four task distributions as in previous experiments are shown in Fig. 6. It is assumed that the communication failure happens once for a while due to the environment or track jam, and it is temporarily and can be recovered very soon. It can be seen that the communication failure didn't affect the system performance extensively. This is because once a robot detects a communication failure, it would send signals again in next cycle until the acknowledgement is received, which prevents the signal loss due to the

communication failure. In this simulation, it is assumed that R(S1,1) robot fails. The simulation results show that the system performance degraded at some level with a failure robot instead of being stuck forever.

All of these simulation results were based on the project with 8 tasks and 6 different robots. In real time applications, it can be easily extended to large scale systems with more robots and complicated tasks.

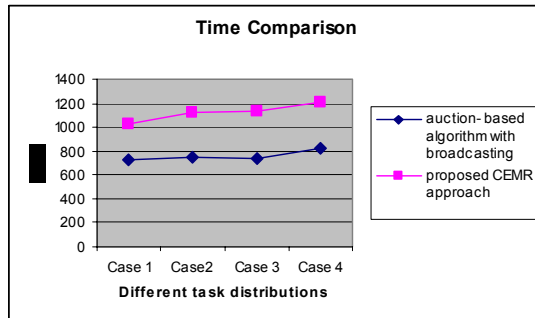


Fig.5. Time to complete the tasks compared with an auction-based method using broadcasting (the unit for time is the simulation iteration number)

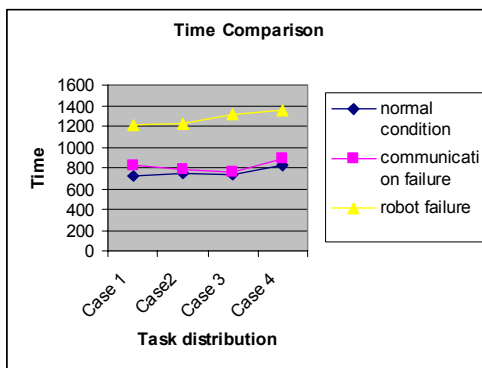


Fig.6. Time to complete the tasks with system failures (the unit for time is the simulation iteration number)

## VI. CONCLUSION AND FUTURE WORK

In this paper, a communication-efficient multi-robot (CEMR) task scheduling method is proposed, which improve the system efficiency and reduce the memory requirement using a global unit and reduce the communication overhead through a distributed decision making among the robots. This algorithm avoids unnecessary communication by broadcasting global information only to those robots who are interested in the tasks. To handle tasks under dynamic environment, two cases are considered in this paper. First, the environment of tasks has been changed. Each robot would dynamically allocate a task which is difficult for itself due to changed environment to robots which are capable and most available. Second, when a robot fails, the other robots will take over its unfinished task and update the global unit of this failure so that no task will be allocated to this failed robot. These features make the system robust against communication failures and robot failures.

Since only the proof-of-concept simulations have been

conducted in this paper, there are some real-world issues need to be investigated in our future work. For example, what kinds of on-board sensors can be applied for robots to detect different environment? How to detect target tasks? In the future work, we will also explore the cases that robots can be dynamically added or removed from the current team. We will also apply the proposed algorithm into a real-world multi-robot system for construction work under dynamic environment, where the robot control algorithm will be integrated into task allocation methods.

## REFERENCES

- [1] Vail and Veloso, "Dynamic multi-robot coordination," *Multi-Robot Systems: From Swarms to Intelligent Automata*, vol. II, pp. 87–100, 2003.
- [2] B. Gerkey and M. J. Mataric, "Murdoch: Publish/subscribe task allocation for heterogeneous agents," *Proceedings of Autonomous Agents*, Barcelona, Spain, June 3-7, pp. 203–204, 2000.
- [3] D. Rus, S. Singh, S.-V. B. Heidelberg, B. P. Gerkey., and M. J. Matarik, "Principal communication for multi robot task allocation," *Experimental Robotics VII*, LNCIS, pp. 353–362, 2001.
- [4] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation," *IEEE Trans. Robot. Automat.*, vol. 14, p. 220240, Apr 1997.
- [5] L. E. Parker, "Task-oriented multi-robot learning in behavior-based systems," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1996.
- [6] J. McLurkin and D. Yamins, "Dynamic task assignment in robot swarms," *IEEE Proceedings of Robotics: Science and Systems, June 2005*. MIT Computer Science and Artificial Intelligence Lab.
- [7] A. W. Stroupe, R. Ravichandran, and T. Balch, "Value-based action selection for exploration and dynamic target observation with robot teams," *Proceedings of the IEEE Conference on Robotics and Automation*, 2004.
- [8] B. Brumitt and A. Stentz, "Dynamic mission planning for multiple mobile robots," *Proceedings of the IEEE International Conference on Robotics and Automation*, 1996.
- [9] R. Davis and R. G. Smith, "Negotiation as a metaphor for distributed," *Conf. Artificial Intelligence Laboratory, Massachusetts Institute, of Technology, Cambridge, MA 02139*, ETATS-UNIS, vol. 20, no. 1, p. 63109, 1993.
- [10] N. Miyata, J. Ota, T. Arai, and H. Asama, "Cooperative transport by multiple mobile robots in unknown static environments associated with real-time task assignment," *IEEE transactions on robotics and automation*, vol. 18, no. 5, 2002.
- [11] D. J. Stilwell and J. S. Bay, "Toward the development of a material transport system using swarms of ant-like robots," *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 766-771, 1993.
- [12] C. Hsieh and R. Murray, "Experimental implementation of an algorithm for cooperative searching of target sites," *Proceedings of American Control Conference*, 2005.
- [13] A. Gage, D. Murphy, D. Valavanis, and M. Long, "Affective task allocation for distributed multi-robot teams", *Technical report TR2006-26 for Center for Robot-Assisted Search and Rescue (CRASAR)*.