Robotics and Autonomous Systems 103 (2018) 151–161

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Resource-based task allocation for multi-robot systems

Dong-Hyun Lee

Department of IT Convergence Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi, Gyeongbuk, Republic of Korea

HIGHLIGHTS

- A resource-based task allocation algorithm for multi-robot systems in a dynamic environment is proposed.
- The proposed approach has four processes: winner update process, task trade process, bid generation process, and list update process.
- The effectiveness of the proposed algorithm with respect to task completion time, resource consumption, and communication overhead is theoretically analyzed and is also demonstrated from the simulation of the delivery mission.

ARTICLE INFO

Article history:
Received 9 May 2017
Received in revised form 8 January 2018
Accepted 23 February 2018
Available online 3 March 2018

Keywords:
Multi-robot systems
Decentralized coordination
Resource-based task allocation

ABSTRACT

This paper proposes a resource-based task allocation algorithm for multi-robot systems. During mission operations, robots continuously consume their resources which must be refilled during their operations. Unlike other existing auction-based algorithms in which robots do not account for their resources in task allocations, the proposed algorithm considers the resources of the robots to generate their costs. In this algorithm, robots calculate the task performance estimation considering all the possibilities of visiting different combinations of refill stations based on their resource levels. This enables the robots to reduce unnecessary wastage of time and resources during the mission. The effectiveness of the proposed algorithm with respect to task completion time, resource consumption, and communication overhead is theoretically analyzed and is also demonstrated from the simulation of the delivery mission.

1. Introduction

Multi-robot systems have been used in various missions that require fast, efficient, and robust performance, such as surveillance, reconnaissance, and continuous area sweeping [1–4]. The multi-robot task allocation problem is one of the core issues in multi-robot systems. In particular, auction-based approaches in multi-robot task allocation (MRTA) have recently received significant attention and become a key research topic in multi-robot systems [5,6]. The auction-based approaches are based on market economy principles in which each individual tries to maximize its profit, thus leading to the redistribution of revenue and the efficient production of output [5,7]. They can provide flexible task allocation with low computational load and communication overhead with high scalability, in comparison to the centralized approaches, and they efficiently produce suboptimal solutions [8–11]. These approaches make no assumption about the heterogeneity or homogeneity of the robots since robots compute their bids based on their own local information, and the task allocation is solely determined by their bids. In the auction algorithms, an auctioneer agent evaluates the received bids from the bidder agents. The auctioneer then allocates the task to the bidder that has the best bid [12,13]. The decentralized auction algorithms based on local communication have also been developed to allow the agents to bid on a task asynchronously [14–16].

There have been numerous studies on recharging robots in long-term tasks [17,18]. They focus on developing energy efficient methods of recharging robots such as identifying the optimal location of charging stations, and using an energy-transporting robot for recharging worker robots [19–21]. Some of them deal with the problem of sharing charging stations among robots without conflict [22,23]. However, they consider robot resources in the task execution phase, rather than in the task allocation phase. The offline temporal interval propagation method is proposed in [24] for each robot to select a task to perform considering its resources. However, since this approach requires all the task information in advance of the task execution, it is not applicable for a mission in a dynamic environment where the information of tasks is not available a priori. In [25], a reinforcement learning based schedule planning system is proposed for a group of robots to perform tasks considering their resources. However, the system requires a fully connected communication network for all the robots to share their information and jointly plan a task. The energy-aware task allocation method in [26] utilizes the robot energy to generate the cost of

E-mail address: donglee@kumoh.ac.kr.

https://doi.org/10.1016/j.robot.2018.02.016
0921-8890/

the robot for a task. However, recharging is not considered in the approach such that the robots should perform the task with the initially given energy. Although some task allocation approaches considering multiple resources exist, they do not consider the execution order of multiple tasks since they execute tasks in the first-come first-served manner [27,28].

This paper proposes a resource-based task allocation algorithm for efficient multiple task execution in long-term operation missions with refill stations for the robot resources. The proposed work focuses on the auction-based task allocation problem for a long-term mission without or with little human intervention. During the mission operations, robots occasionally have to refill or recharge their resources. The robot resources represent the consumable resources that robots devour during the task execution, such as battery life and fuel. The resources that this paper considers are either refillable or replaceable, and if any of them are lower than certain thresholds, the robot will not be able to perform or complete its given tasks. For example, a floor-cleaning robot consumes battery energy while moving around and brushing the floor, and fills its dust bin with absorbed dust. Therefore, the remaining amount of battery and the remaining space in the dust bin can be considered as the resources of the robot. If the robots run out of resources, they should stop performing their tasks and visit the respective stations to refill their resources. This causes an unexpected increase in time and resource consumption during the task execution. Thus, if the cost of the robot does not reflect such situations, the task allocation algorithms do not guarantee the expected task completion results, such as time cost and resource consumption. In the proposed algorithm, each robot generates its cost, i.e., its bid value, for the task considering its resources. The proposed algorithm is able to maximize the efficiency of the resource usage of robots in terms of task completion time and communication among robots by considering the robot resources and applying decentralized bidding and assignment processes.

This paper is organized as follows: Section 2 describes the effects of the robot's resources in the task allocation problem. Section 3 introduces the four main processes of the proposed approach: bid generation process, winner update process, task trade process, and list update process. The performance of the proposed approach is analyzed in detail in Section 4. The simulation results are discussed in Section 5 and concluding remarks follow in Section 6.

2. Robot resources in task allocation

2.1. Effects of robot's resources in auction-based algorithms

Conceptual insights into the effects of a robot's resources in the auction-based algorithm are illustrated in Fig. 1. The bar in the square speech bubble represents the battery level of robot 1, which decreases as the robot moves around and performs tasks, and x_{min} is the threshold battery level. When the battery goes below the threshold, the robot should visit the station and wait until it is fully recharged. Suppose that robot 1 and robot 2 generate bids for task 1 and task 2 based on the estimated time to complete the tasks by following the shortest paths, and robot 1 wins both tasks because of its higher bids. Although robot 1 has enough battery to complete task 1, it should stop moving forward to task 2 and change its direction to the station at point P_S since the battery level went below the threshold, as shown in the figure. As a result, the actual time for robot 1 to complete task 2 becomes longer than its estimated time since it makes a detour to the station for recharge, waits until it is fully recharged, and then goes to task 2. It shows that the auction-based algorithms that do not consider the resources of the robots will not guarantee the optimal solution.

Other approaches are available to make up for the suboptimal results such as re-auctioning task 2 after completing task 1, not bidding on task 2 in the first place, etc. However, they may not guarantee that the tasks are completed in the estimated time of the winner since the result is highly dependent on the battery levels of the robots. For example, consider a cleaning robot that picks up debris from the floor. During the cleaning task, the robot consumes battery power while moving around and brushing the floor, and fills the dust bin with absorbed dust. Therefore, the battery level and the remaining space in the dust bin can be considered as the resources of the robot. If either of its resources falls below a certain threshold, for example, when the robot starts running out of battery or space in its dust bin, the robot becomes unable to clean the floor. Suppose that there are multiple cleaning robots in a house, which has many rooms, each robot should decide which room to clean. The conventional and common approach to allocate the room is to assign each room to the robot that is the closest. This approach can guarantee minimum task completion time under the assumption that all of the robots have enough resources to complete their tasks. However, the assumption is not always satisfied in reality. If one of their resources is running out while the robots are performing their tasks, they should stop working and spend their time and resources to recharge their batteries and empty the dust bins. As a result, assigning tasks without considering the resources of the robots can cause a significantly bad performance because of the unexpected waste of time and resources.

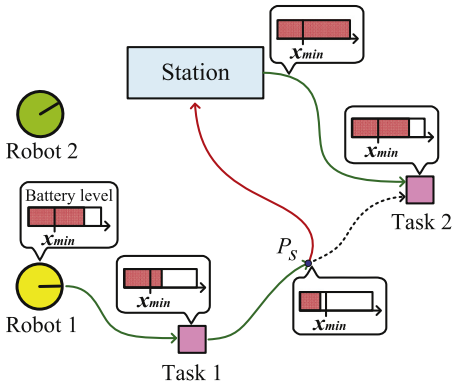

Fig. 1. Illustration of the effect of the robot resource in the auction-based algorithm.

2.2. Task allocation problem considering robot resources

The objective of task assignment in multi-robot systems is to either maximize the global reward or minimize the global cost by allocating tasks to the robots that can either maximize their local rewards or minimize their local costs. In the case of auction-based algorithms, the local cost can be considered as the sum of bids for the assigned tasks to the robot, and the global cost can be defined as the sum of the local costs of the robots. To consider the resources of a robot in task allocation, the resource vector of robot i, $\mathbf{x}^i$ is defined as

$$\mathbf{x}^i = \begin{bmatrix} x_1^i & x_2^i & \cdots & x_{N_L}^i \end{bmatrix}^T \tag{1}$$

where x_l^i is the lth resource level of robot i, and N_L is the number of resources. The task assignment problem considering the resources of the robots can be represented as minimizing the global objective function as follows:

$$\min \sum_{i \in \mathscr{R}} \sum_{j \in \mathbf{A}^i} b_j^i(\mathbf{x}^i, \mathbf{A}^i), \quad i \in \mathscr{R}, \ \mathbf{A}^i \subset \mathscr{T} \tag{2}$$

subject to

$$\sum_{i\in\mathscr{R}} e(j,\mathbf{A}^i)=1,\quad \forall j\in\mathscr{T}$$
$$\sum_{i\in\mathscr{R}}\sum_{j\in\mathscr{T}} e(j,\mathbf{A}^i)=N_T \tag{3}$$
$$x_l^i > \check{x}_l^i,\ \ \forall i\in\mathscr{R},\ \forall l\in\mathscr{L}$$

where $\mathscr{R}$ is the robot set ($\mathscr{R}=\{1,2,\ldots,N_R\}$), $\mathscr{T}$ is the task set ($\mathscr{T}=\{1,2,\ldots,N_T\}$), $\mathscr{L}$ is the resource set ($\mathscr{L}=\{1,2,\ldots,N_L\}$), N_R, N_T and N_L are the numbers of robots, tasks, and resources, respectively. $\mathbf{A}^i$ is the task list of robot i which represents an ordered sequence of tasks, and $b_j^i(\mathbf{x}^i,\mathbf{A}^i)$ is the bid of robot i for task j. x_l^i is the resource level of the lth resource element of robot i, $\check{x}_l^i$ is the threshold of the lth element, and $e(j,\mathbf{A}^i)=1$ if task j is in $\mathbf{A}^i$ and 0 otherwise. The first constraint in (3) shows that all the tasks should be allocated to the robots, and the second constraint shows that multiple robots should not have the same task. The third constraint shows that all the resource elements of robot i after completing any task in $\mathbf{A}^i$ should be higher than their thresholds. It does not necessarily mean that robot i should have enough resources to complete all the tasks in its task list since it can refill its resources during the mission.

3. Resource-based task allocation

The proposed approach consists of four processes: bid generation, winner update, task trade, and list update. Unlike the conventional bid generation process, in which the bid for a task is calculated deterministically under the assumption that a robot has enough resources to perform the task, the proposed bid generation process uses a probabilistic bid generation scheme that considers the robot resources. During iterations of the winner update and the task trade processes, each robot receives and transmits bid messages asynchronously, and determines whether to accept or reject the tasks. The list update process continuously updates three lists: the task list, bid list, and resource list. The task list consists of the tasks that the robot has accepted, and the bids of the tasks are stored in the bid list. The estimated residual resources after completing the tasks are recorded in the resource list. These lists are updated based on the robot state and the current residual resources. The bid generation process utilizes the lists for generating bids of new trading tasks.

Fig. 2 shows the robot architecture for task allocation and the data flow between the four processes. The winner update process receives bid messages of the trading tasks from other robots and requests a bid for a new trading task to the bid generation process. The bid generation process calculates the bid of the new task based on the task list, bid list, and resource list, which are provided by the list update process. The winner update process then updates the trade list, which consists of information about the current trading tasks. From the trading task, the task trade process determines what task to bid on, and transmits a bid for the task. After waiting for a certain amount of time for receiving bids for the task from other robots, the task trade process determines whether to accept or reject the task, and the accepted task is sent to the list update process. The task execution sequence can be changed depending on the task execution order of the accepted task. If there are already other accepted tasks in the task list, the tasks that have their execution orders changed by the new accepted task are considered to be outdated. After updating the task list with the new accepted task, the list update process sends the outdated tasks to the winner update process for trading.

3.1. Bid generation process

The proposed bid generation process not only considers the bid of a robot to directly perform the task without refill but also accounts for the bids for the cases in which the robot visits different combinations of refill stations. Thus, the locations of the refill stations should be provided to the robots before the start of the mission. Each path consists of none or different combinations of refill stations and the probability of the robot selecting one of the paths for performing the task is calculated considering the task execution order and the estimated residual resources after completing the task. To reduce the computation cost in bid generation, the task execution order of the accepted tasks and their bids are stored in the task list and the bid list, respectively. The estimated residual resources after completing the accepted tasks are stored in the resource list. The task list contains an ordered sequence of accepted tasks such that the task list of robot i, $\mathbf{A}^i$, is defined as

$$\mathbf{A}^i=[A_1^i,A_2^i,\ldots,A_{|\mathbf{A}^i|}^i] \tag{4}$$

where A_n^i represents that robot i plans to perform task A_n^i in the nth order, $|\mathbf{A}^i|$ ($0\le|\mathbf{A}^i|\le N_A$) is the cardinality of $\mathbf{A}^i$, $\mathbf{A}_{1:n}^i=[A_1^i,A_2^i,\ldots,A_n^i]$ ($n\in\{1,2,\ldots,|\mathbf{A}^i|\}$) is the subset of $\mathbf{A}^i$, and N_A is the maximum assignment size such that a robot can get at most N_A tasks. Note that $\mathbf{A}_{1:n}^i=[\,]$ if $n<1$.

The bids of the tasks in the task list are stored in the bid list such that the bid list of robot i, $\mathbf{B}^i$, is defined as

$$\mathbf{B}^i=[B_1^i,B_2^i,\ldots,B_{|\mathbf{A}^i|}^i] \tag{5}$$

where B_n^i is the bid of task A_n^i. Using the bid list, the robot can compare its bid with the other robots' bids. The bid list is also used to decide the execution order of the task that the robot bids on. The estimated residual resources after completing the task in the task list are represented as a random vector of the multivariate normal distribution. The resource vector of robot i after completing task A_n^i, X_n^i is defined as

$$X_n^i=\begin{bmatrix}x_{n,1}^i & x_{n,2}^i & \cdots & x_{n,N_L}^i\end{bmatrix}^T,\ X_n^i\sim\mathscr{N}_{N_L}(\mu_n^i,\Sigma_n^i) \tag{6}$$

where $x_{n,l}^i$ is the estimated value of the lth residual resource of robot i after completing task A_n^i, N_L is the number of resources, μ_n^i is an N_L-dimensional mean vector, and Σ_n^i is an $N_L\times N_L$ covariance matrix. The current residual resources of robot i is denoted as X_0^i. The resource vectors for the tasks in the task list are stored in the resource list such that the resource list of robot i, $\mathbf{X}^i$ is defined as

$$\mathbf{X}^i=[X_1^i,X_2^i,\ldots,X_{|\mathbf{A}^i|}^i] \tag{7}$$

where X_n^i is the resource vector of task A_n^i.

Fig. 3 illustrates the bid generation process of robot i for task j, where C_n^i is the cost of task A_n^i. To bid on task j, the robot first decides the execution order of task j considering the task list, the bid list and the resource list. If the robot plans to perform task j in the n_jth order, the resource vector of the n_j-1th task, $X_{n_j-1}^i$ is used to calculate the selection probability of each path to perform task j. Then, the estimated cost for the task is calculated using the path selection probabilities and the costs for the paths. The bid generation process is described in detail in the following.

3.1.1. Resource vector

As shown in Fig. 3, there are N_P ($=2^{N_L}$) paths for robot i to perform task j in which each of the paths consists of none or different combinations of refill stations. For example, if the robot uses a single resource, there are two paths: one that directly leads the robot to the task, and the other to the refill station before heading to the task. In this paper, it is assumed that there are N_L

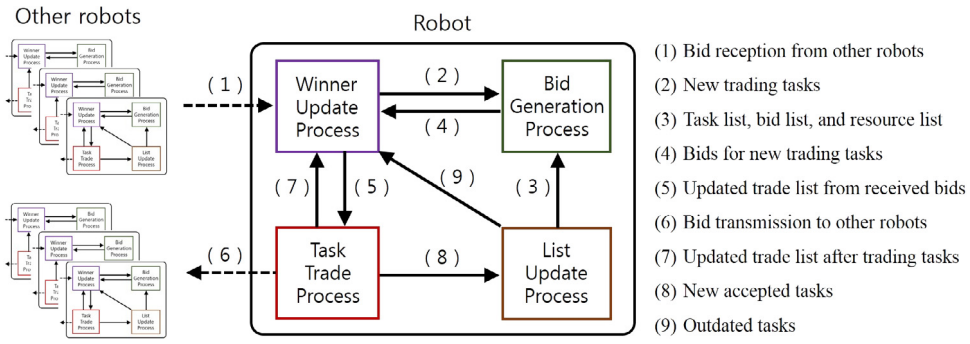

Fig. 2. Four processes in the proposed approach. Each robot consists of four processes: bid generation, winner update, task trade and list update processes.

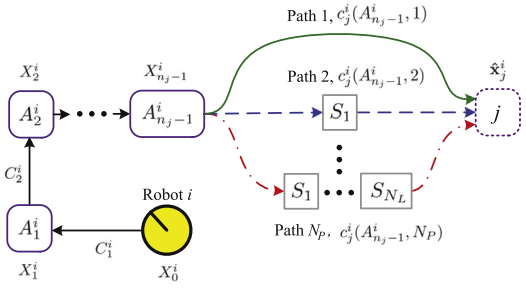

Fig. 3. Bid generation process of robot i for task j. In the figure, A_n^i is the nth task in the task list of robot i, X_n^i is the resource vector of robot i for the nth task in its task list, C_n^i is the cost to complete the nth task after completing the $n-1$th task. To calculate the estimated bid for task j, robot i firstly finds the optimal task execution order of task j. If n_j is selected as the execution order for task j, the robot calculates the estimated bid for the task using the costs and the probabilities of 2^{N_L} numbers of paths where N_L is the number of robot resources.

fixed refill stations for corresponding N_L resources and they are located close to each other such that visiting a different order of the stations has no effect on the path. The refill stations that lie on path k are represented by the station vector, $\mathbf{v}(k)$, defined as

$$\mathbf{v}(k) = \begin{bmatrix} v_1(k) & v_2(k) & \cdots & v_{N_L}(k) \end{bmatrix}, \; k \in \{1, 2, \ldots, N_P\} \tag{8}$$

with

$$v_l(k) = \text{rem}(\lfloor (k-1)/2^{l-1} \rfloor, 2), \;\; l \in \{1, 2, \ldots, N_L\} \tag{9}$$

where $\text{rem}(a, b)$ is the remainder of a divided by b, $\lfloor c \rfloor$ is the floor function which gives the largest integer less than or equal to c, and $v_l(k) (\in \{0, 1\})$ is the visit indicator to denote whether station l lies on path k. For example, if three resources are used, i.e., $N_L = 3$, no station lies on path 1, therefore all the visit indicators are set to zero. In the case of path 4, station 1 and station 2 lie on the path such that $v_1(4)$ and $v_2(4)$ are set to one. For path 8, all the stations lie on the path, therefore all the visit indicators are set to one. If the robot plans to perform task j after completing task $A^i_{n_j-1}$ $(1 \le n_j \le \min(|\mathbf{A}^i| + 1, N_A))$, the resource vector of the robot after completing task j by taking path 1, $\mathbf{x}_j^i(A^i_{n_j-1}, X^i_{n_j-1}, 1)$, is defined as

$$\mathbf{x}_j^i(A^i_{n_j-1}, X^i_{n_j-1}, 1) = X^i_{n_j-1} - \mathbf{D}^i \mathbf{U}_j^i(A^i_{n_j-1}) \tag{10}$$

where $\mathbf{U}_j^i(A^i_{n_j-1})$ is an N_U-dimensional resource consumption input vector of robot i for task j after completing $A^i_{n_j-1}$ and $\mathbf{D}^i$ is an $N_L \times N_U$ resource consumption rate matrix. If n_j is one, $\mathbf{U}_j^i(A_0^i)$ represents the resource consumption input vector of robot i for task j starting from the robot's current location. In the case of the other paths except for path 1, $\mathbf{x}_j^i(A^i_{n_j-1}, X^i_{n_j-1}, k)\,(1 < k \le N_P)$ is calculated as

$$\begin{aligned} \mathbf{x}_j^i(A^i_{n_j-1}, X^i_{n_j-1}, k) &= X^i_{n_j-1} - \mathbf{D}^i \mathbf{U}_S^i(A^i_{n_j-1}) \\ &+ \mathbf{V}(k)(\mathbf{x}^i_{max} - X^i_{n_j-1} + \mathbf{D}^i \mathbf{U}_S^i(A^i_{n_j-1})) - \mathbf{D}^i \mathbf{U}^i_{j,S} \end{aligned} \tag{11}$$

where $\mathbf{U}_S^i(A^i_{n_j-1})$ is the resource consumption input vector of robot i to visit the stations after completing $A^i_{n_j-1}$, $\mathbf{U}^i_{j,S}$ is the resource consumption input vector of robot i to perform task j starting from the stations, $\mathbf{x}^i_{max}$ is the maximum resource vector which represents the maximum resource levels of robot i after refilling, and $\mathbf{V}(k)$ is the diagonal matrix of $\mathbf{v}(k)$. In the right-hand side of the above equation, the subtraction of $\mathbf{D}^i \mathbf{U}_S^i(A^i_{n_j-1})$ from $X^i_{n_j-1}$ represents that the robot consumes its resources on its way to the stations, and the addition of $\mathbf{V}(k)(\mathbf{x}^i_{max} - X^i_{n_j-1} + \mathbf{D}^i \mathbf{U}_S^i(A^i_{n_j-1}))$ means that the robot fully refills the resources from the stations that the robot has visited by taking path k. After refill, the robot consumes its resources by $\mathbf{D}^i \mathbf{U}^i_{j,S}$ to complete task j.

From (10) and (11), the general form of the resource vector of robot i for task j with path k, $\mathbf{x}_j^i(A^i_{n_j-1}, X^i_{n_j-1}, k)$ is defined as

$$\begin{aligned} \mathbf{x}_j^i(A^i_{n_j-1}, X^i_{n_j-1}, k) &= (\mathbf{I} - \mathbf{V}(k))X^i_{n_j-1} - \mathbf{D}^i \tilde{\mathbf{U}}_j^i(k) \\ &+ \mathbf{V}(k)(\mathbf{x}^i_{max} + \mathbf{B}^i \mathbf{U}_S^i(A^i_{n_j-1})) \end{aligned} \tag{12}$$

with

$$\tilde{\mathbf{U}}_j^i(k) = \begin{cases} \mathbf{U}_j^i(A^i_{n_j-1}), & \text{if } k = 1 \\ \mathbf{U}_S^i(A^i_{n_j-1}) + \mathbf{U}^i_{j,S}, & \text{otherwise} \end{cases} \tag{13}$$

where $\mathbf{I}$ is an $N_L \times N_L$ identity matrix. The resource vectors and the path selection probabilities of the paths are used to calculate the estimated resource vector of the robot to complete the task. The path selection probability is described in detail in the next subsection.

3.1.2. Path selection probability

As each path consists of none or different combinations of refill stations, the residual resources of a robot after completing a task depends on which path the robot chooses for the task. Thus, the robot should choose the right path to complete the task without running out of any of its resources. The task assurance probability of the path is defined as the probability of a robot having higher residual resources than their corresponding thresholds after completing a task by taking the path. The task assurance probability of path k for robot i to perform task j, $H_j^i(k)$, is defined as

$$H_j^i(k) = \prod_{l=1}^{N_L} p(x^i_{j,l}(A^i_{n_j-1}, x^i_{n_j-1,l}, k) > \gamma^i_{j,l}; \mu^i_{j,l}, (\sigma^i_{j,l})^2) \tag{14}$$

with

$$\Gamma_j^i = \mathbf{D}^i \mathbf{U}_{j,S}^i + \mathbf{G} \tag{15}$$

where $x_{j,l}^i(A_{n_j-1}^i, x_{n_j-1,l}^i, k)$ is the lth resource element of $\mathbf{x}_j^i(A_{n_j-1}^i, X_{n_j-1}^i, k)$, Γ_j^i is the N_L-dimensional threshold vector which represents the minimum amount of required residual resources to visit the stations after completing task j. $\gamma_{j,l}^i$ is the lth element of Γ_j^i, and $\mathbf{G}$ is an N_L-dimensional marginal vector which provides margins to the threshold vector. As the path directs a robot to more refill stations, the task assurance probability of the path becomes higher, such that the task assurance probability of path N_L is always one. However, as the robot visits more refill stations, the average task completion time would be increased since the robot should wait a certain amount of time to refill in the stations. Thus, the robot should select the right path to refill only the necessary resources that might be drained during task execution. For the robot to select such a path in task execution planning, the path selection probability is calculated by normalizing the change rate of the task assurance probability of the path. The change rate of the task assurance probability of path k for robot i to perform task j, $\Delta H_j^i(k)$, is defined as

$$\Delta H_j^i(k) = H_j^i(k) - \lambda_{m(k)-1} \tag{16}$$

where $\lambda_{m(k)}$ ($m(k) \in \{1, 2, \ldots, N_P\}$) is the $m(k)$th smallest probability value among the task assurance probabilities of N_P paths. For example, if $H_j^i(1)$ is the second smallest and $H_j^i(2)$ is the smallest among the task assurance probabilities, $m(1) = 2$ and $m(2) = 1$ so that $\lambda_{m(2)} = \lambda_1 = H_j^i(2)$ and $\lambda_{m(1)} = \lambda_2 = H_j^i(1)$. Thus, $\Delta H_j^i(1) = H_j^i(1) - \lambda_{m(1)-1} = H_j^i(1) - H_j^i(2)$. In case where $m(k)-1 = 0$, λ_0 is set to zero so that if $H_j^i(k)$ is the smallest one, $\Delta H_j^i(k)$ equals to $H_j^i(k)$. If some of them have the identical probability value, the path with a lower *ID* number is considered to be smaller than the one with a higher path *ID* number. This enables the path that consists of fewer refill stations to have a higher change rate than the path with more stations. From (16), the path selection probability of robot i for task j with path k, $P_j^i(k)$, is defined as

$$P_j^i(k) = \frac{\Delta H_j^i(k)}{\sum_{k'=1}^{N_P} \Delta H_j^i(k')}. \tag{17}$$

From (12) and (17), the estimated resource vector of robot i for task j, $\hat{\mathbf{x}}_j^i$, is defined as

$$\hat{\mathbf{x}}_j^i = \sum_{k=1}^{N_P} P_j^i(k)\, \mathbf{x}_j^i(A_{n_j-1}^i, X_{n_j-1}^i, k). \tag{18}$$

If the robot becomes the winner for the task, the estimated resource vector for the task is stored in the resource list. By using the resource list, the computation cost for calculating the path selection probabilities and the estimated resource vectors for the new tasks can be reduced.

3.1.3. Bid generation

The estimated bid of the robot for the task depends on the estimated cost and the team objectives [29,30]. If the team objective is to minimize the sum of the costs of all the robots (*MiniSum*), the estimated bid is defined as the estimated cost. However, if the team desires to minimize the maximum cost (*MiniMax*), the estimated bid is defined as the sum of the estimated costs of the preceding tasks in the execution order and the estimated cost of the bidding task.

The estimated cost of robot i for task j with path k after completing $A_{n_j-1}^i$ is denoted as $c_j^i(A_{n_j-1}^i, k)$. From the costs and the path selection probabilities of the N_P paths, the estimated cost of robot i for task j, $\hat{c}_j^i$, is defined as

$$\hat{c}_j^i = \sum_{k=1}^{N_P} P_j^i(k)\, c_j^i(A_{n_j-1}^i, k) \tag{19}$$

where n_j denotes that the robot plans to execute task j in the n_jth order. In the case when n_j is one, $c_j^i(A_0^i, k)$ represents the cost of robot i to perform task j with path k from its current location. From (19), the estimated bid of robot i for task j, $\hat{b}_j^i$, is defined as

$$\hat{b}_j^i = \begin{cases} \hat{c}_j^i, & \text{if } Obj = MiniSum \\ B_{n_j-1}^i + \hat{c}_j^i, & \text{else if } Obj = MiniMax \end{cases} \tag{20}$$

where Obj ($\in \{MiniSum, MiniMax\}$) denotes the team objective in task allocation and $B_0^i = 0$.

Algorithm 1 shows the bid generation procedure of robot i for task j. The algorithm firstly finds the optimal execution order of task j, n_j ($\in \{1, 2, \ldots, \min(|\mathbf{A}^i| + 1, N_A)\}$), in the task list. The estimated cost of task j, $\hat{c}_j^i$, in the n_j'th order is compared with the cost of $A_{n_j'}^i$ until n_j' is increased to $\min(|\mathbf{A}^i| + 1, N_A)$. The cost of $A_{n_j'}^i$, $C_{n_j'}^i$ is defined as

$$C_{n_j'}^i = \begin{cases} B_{n_j'}^i, & \text{if } Obj = MiniSum \\ B_{n_j'}^i - B_{n_j'-1}^i, & \text{if } Obj = MiniMax. \end{cases} \tag{21}$$

The algorithm then returns the estimated bid ($\hat{b}_j^i$ $\hat{x}_j^i$), resource vector ($\hat{x}_j^i$), and the execution order (n_j) of the task.

Algorithm 1 Bid generation process of robot i for task j

```
procedure BidGen(j)
    for n_j' = 1 to min(|A^i|+1, N_A) do
        ĉ_j^i = Σ_{k=1}^{N_P} P_j^i(k) c_j^i(A^i_{n_j'-1}, k)
        if ĉ_j^i < C^i_{n_j'} ∨ n_j' > |A^i| then
            n_j = n_j'
            if Obj=MiniSum then
                b̂_j^i = ĉ_j^i
            else if Obj=MiniMax then
                b̂_j^i = B^i_{n_j-1} + ĉ_j^i
            end if
            x̂_j^i = Σ_{k=1}^{N_P} P_j^i(k) x_j^i(A^i_{n_j-1}, X^i_{n_j-1}, k)
            break
        end if
    end for
    return b̂_j^i, x̂_j^i, n_j
end procedure
```

3.2. Winner update process

In the proposed approach, robots send and receive the bid messages among their neighbors and allocate the tasks in a decentralized manner. The received bid message is stored in the bid vector such that the nth bid vector of robot i is defined as

$$Z_n^i = \begin{bmatrix} z_{n,1}^i & z_{n,2}^i & z_{n,3}^i & z_{n,4}^i \end{bmatrix} \tag{22}$$

where the *ID* of the task, the *ID* of the candidate robot for the task and the bid of the robot for the task are stored in $z_{n,1}^i$, $z_{n,2}^i$ and

Algorithm 2 Winner update process of robot i

1: **procedure** *WinnerUpdate*($\mathbf{Q}^i$, $\mathbf{Z}^i$)
2: **for** $n = 1$ to $|\mathbf{Z}^i|$ **do**
3: $j = z^i_{n,1},\ k = z^i_{n,2},\ b^k_j = z^i_{n,3},\ auc = z^i_{n,4}$
4: **if** $auc =$ **True** $\wedge\ Q^i_j \notin \mathbf{Q}^i$ **then**
5: $[\hat{b}^i_j\ \hat{\mathbf{x}}^i_j\ n_j] = BidGen(j)$
6: **if** $i = k$ **then**
7: $Q^i_j = [i\ \hat{b}^i_j\ \hat{\mathbf{x}}^i_j\ n_j$ **True**$]$
8: **else if** $\hat{b}^i_j < b^k_j$ **then**
9: $Q^i_j = [i\ \hat{b}^i_j\ \hat{\mathbf{x}}^i_j\ n_j$ **False**$]$
10: **else**
11: $Q^i_j = [k\ b^k_j\ \vec{0}\ 0$ **False**$]$
12: **end if**
13: $\mathbf{Q}^i.PushBack(Q^i_j)$
14: **else if** $Q^i_j \in \mathbf{Q}^i \wedge b^k_j < q^i_{j,2}$ **then**
15: $Q^i_j = [k\ b^k_j\ \vec{0}\ 0\ q^i_{j,5}]$
16: **end if**
17: **end for**
18: $\mathbf{Z}^i = \emptyset$
19: **return** $\mathbf{Q}^i$
20: **end procedure**

Algorithm 3 Task trade process of robot i

1: **procedure** *TaskTrade*($\mathbf{Q}^i$)
2: $Q^i_m = \vec{0}$
3: **if** $j^\star = 0$ **then**
4: $j^\star = \arg\min_{j \in \{j | Q^i_j \in \mathbf{Q}^i, q^i_{j,1} = i\}} q^i_{j,2}$
5: $Z_{tmp} = [j^\star\ i\ q^i_{j^\star,2}\ q^i_{j^\star,5}]$
6: $Transmit(Z_{tmp})$
7: $\mathbf{Q}^i = [Q^i_{j^\star}]$
8: **else if** $t_{wait} > t_{delay}$ **then**
9: **if** $q^i_{j^\star,1} = i$ **then**
10: $Q^i_m = Q^i_{j^\star}$
11: **end if**
12: **if** $q^i_{j^\star,5} =$ **True then**
13: $Z_{tmp} = [j^\star\ q^i_{j^\star,1}\ q^i_{j^\star,2}$ **False**$]$
14: $Transmit(Z_{tmp})$
15: **end if**
16: $\mathbf{Q}^i.Remove(Q^i_{j^\star}),\ j^\star = 0$
17: **end if**
18: **return** $\mathbf{Q}^i,\ Q^i_m$
19: **end procedure**

$z^i_{n,3}$, respectively. If the message is from an auctioneer, $z^i_{n,4}$ is set to **True**, otherwise it is set to **False**. The auctioneer is the robot that first announces a task for trading and any robot can become an auctioneer to trade a task Using $z^i_{n,4}$ guarantees task trading between the robots within a single hop of communication. The bid list of robot i, $\mathbf{Z}^i$ is then defined as

$$\mathbf{Z}^i = [Z^i_1, Z^i_2, \ldots, Z^i_{|\mathbf{Z}^i|}]. \tag{23}$$

When robot i either detects a new task to be allocated or when it has outdated tasks, it desires to become an auctioneer for task j. To become an auctioneer for task j, robot i add a bid vector, $Z^i_n = [j\ i\ b_D$ **True**$]$, to its bid list, where b_D is a dummy bid that is a large constant value.

After receiving the bid messages, each robot compares the bids in the bid list with the bids of the best candidates in the trade list. The trade list consists of the trade vectors such that the trade vector of robot i for task j, Q^i_j, is defined as

$$Q^i_j = \begin{bmatrix} q^i_{j,1} & q^i_{j,2} & q^i_{j,3} & q^i_{j,4} & q^i_{j,5} \end{bmatrix} \tag{24}$$

where the *ID* and the bid of the best candidate, the resource vector and the execution order for task j are stored in $q^i_{j,1}$, $q^i_{j,2}$, $q^i_{j,3}$ and $q^i_{j,4}$, respectively. The Boolean value $q^i_{j,5}$ is used to indicate whether robot i is the auctioneer of task j. It is set to **True** if robot i is the auctioneer of task j, and **False** otherwise. When robot i accepts task j, it stores task j, $q^i_{j,2}$ and $q^i_{j,3}$ at the $q^i_{j,4}$th location in the task list, the bid list and the resource list, respectively. In the case where the best candidate for the task is not robot i, it does not use $q^i_{j,3}$ and $q^i_{j,4}$. The trade list of robot i, $\mathbf{Q}^i$ is defined as

$$\mathbf{Q}^i = [Q^i_1, Q^i_2, \ldots, Q^i_{|\mathbf{Q}^i|}]. \tag{25}$$

Algorithm 2 shows the winner update process of robot i. This process continuously updates the trade list to maintain the best candidates of trading tasks with the lowest bids. For all the bid vectors in the bid list, the new trading tasks from auctioneers, which are not in the trade list, are sent to the bid generation process to get their bid values. After receiving the bid values from the bid generation process, the trade vectors of the new trading tasks are added to the trade list. If the best candidate for the new trading task is the robot itself (line 6), it implies that the auctioneer of the new trading task is itself such that the last Boolean value of the trade vector is set to **True** (line 7). The condition in line 4 shows that a robot only considers the new trading tasks from auctioneers. This prevents the robots that are not the direct neighbors of an auctioneer, i.e., a single hop away from an auctioneer, from bidding on the task. Task trading within a single hop is required since the maximum communication delay between the auctioneer and the bidders can be predefined as a constant value. The maximum communication delay, denoted as t_{delay}, is used as the waiting time for the task trade process to determine the final winner for the new trading tasks. It represents the maximum communication delay of the round trip communication between an auctioneer and its single hop neighbors plus a small time margin to consider unexpected communication delay. The new trading tasks are added at the end of the trade list using the *PushBack* function (line 13).

3.3. Task trade process

The main role of the task trade process is twofold: (1) to select the best task among the new trading tasks and transmit the bid message for the task, (2) to determine the winner of the best task and transmit the final bid message if the robot is an auctioneer for the task. Algorithm 3 shows the task trade process of robot i. If the best task is not yet selected, i.e., $j^\star = 0$, the task that has the lowest bid among the tasks that the robot has won in the trade list is selected as the best task (line 4). Subsequently, the bid message for the best task is sent to the neighbors of robot i using the *Transmit* function (line 6). To determine the winner for the best task, the trade vector of the best task is stored in the trade list and is sent back to the winner update process (line 7, line 18). If the robot remains as the best candidate after waiting for the maximum communication delay, t_{delay}, it accepts the task and transmits the trade vector of the newly accepted task to the list update process (line 10, line 18). If the indicator $q^i_{j^\star,5}$ is **True**, it implies that robot i is the auctioneer of task j. An auctioneer is obligated to transmit the final winner of the task to its neighbors for the bidders, except the winner, to reject the task. To notify that trading task j is completed, the Boolean value of the last element in Z_{tmp} (line 13) is set to **False**, such that this bid message does not trigger other robots to bid on task j again. After trading task j, it is removed from the trade list using the *Remove* function (line 16).

Algorithm 4 List update process of robot i

1: **procedure** *ListUpdate*($\mathbf{Z}^i$, Q_m^i, R_S)
2: **if** $Q_m^i \neq \vec{0}$ **then**
3: $n_m = q_{m,4}^i$
4: $Z_a^i = [j\ i\ b_D\ \textbf{True}],\ \forall j \in \mathbf{A}^i_{n_m:|\mathbf{A}^i|}$
5: $\mathbf{Z}^i.PushBack(Z_a^i)$
6: $\mathbf{A}^i = \mathbf{A}^i_{1:n_m-1},\ \mathbf{B}^i = \mathbf{B}^i_{1:n_m-1},\ \mathbf{X}^i = \mathbf{X}^i_{1:n_m-1}$
7: $\mathbf{A}^i.PushBack(m),\ \mathbf{B}^i.PushBack(q_{m,2}^i),\ \mathbf{X}^i.PushBack(q_{m,3}^i)$
8: **end if**
9: **if** $R_S = Perform$ **then**
10: **if** $J = 0$ **then**
11: $J = A_1^i$
12: $k^\star = \arg\max_k P_J^i(k),\ k \in \{1, 2, ..., N_P\}$
13: **else if** J is completed **then**
14: **if** *Obj=MiniMax* **then**
15: $B_n^i = B_n^i - B_1^i,\ \forall n \in \{1, 2, ..., |\mathbf{A}^i|\}$
16: **end if**
17: $\mathbf{A}^i.PopFront(),\ \mathbf{B}^i.PopFront(),\ \mathbf{X}^i.PopFront()$
18: $J = 0$
19: **end if**
20: **else if** $R_S = Refill\ \wedge\ J \neq 0$ **then**
21: $J = 0$
22: $Z_a^i = [j\ i\ b_D\ \textbf{True}],\ \forall j \in \mathbf{A}^i$
23: $\mathbf{Z}^i.PushBack(Z_a^i)$
24: $\mathbf{A}^i = [\ ],\ \mathbf{B}^i = [\ ],\ \mathbf{X}^i = [\ ]$
25: **end if**
26: **return** $\mathbf{A}^i,\ \mathbf{B}^i,\ \mathbf{X}^i,\ \mathbf{Z}^i$
27: **end procedure**

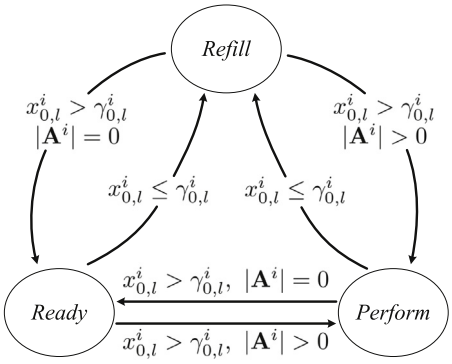

Fig. 4. The state transition graph of robot i, There are three robot states: *Ready*, *Perform* and *Refill*. The robot state is determined by the current residual resources and the number of the tasks in the task list.

3.4. List update process

The list update process updates the task list, the bid list and the resource list based on the robot state, denoted as R_S, and the current residual resources of the robot. After updating the lists, the list update process sends the lists to the bid generation process for bid generation. There are three robot states, i.e., *Ready*, *Perform*, and *Refill*, and the robot state is determined by the current residual resources and the number of tasks in the task list as shown in Fig. 4. When all the residual resources of the robot are higher than their thresholds with no task in the task list, the robot state is set to *Ready*. If the robot has one or more tasks in the task list while having higher residual resources than the thresholds, the robot state is set to *Perform*. If at least one of the residual resources is lower than the threshold, the robot state is set to *Refill*.

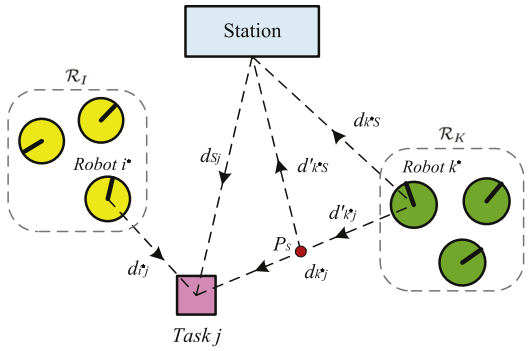

Fig. 5. The illustration shows the effects of robot resources in auction-based task allocations. The set $\mathscr{R}_I$ in the figure represents the set of robots which have enough resources to complete task j, and the others are in the set $\mathscr{R}_K$. The set $\mathscr{R}_U$ is defined as the union of $\mathscr{R}_I$ and $\mathscr{R}_K$. Robot $i^\star$ is the winner in $\mathscr{R}_I$ which has the best bid, and robot $k^\star$ is the winner in $\mathscr{R}_K$. P_S is the point in which robot $k^\star$ redirects from task j to the station due to the depletion of its resource.

Algorithm 4 shows the list update process of robot i. When a newly accepted task is received from the task trade process, its information is updated to the three lists (task list, bid list, and resource list), and the outdated tasks in the task list are sent to the bid list to auction them (lines 2 through 8). In the case when the robot state is *Perform* and the current task, denoted as J, is not yet selected, the first task in the task list is selected as the current task, and the path that has the highest selection probability is selected to perform task J (lines 11 and 12). If task J is completed, its information is removed from the three lists using the *PopFront* function (line 17). In the case when the robot state is *Refill* and the current task is not zero, the tasks in the task list is considered as the outdated tasks such that all the tasks in the task list are sent to the bid list to auction them (line 23).

4. Analysis

The effects of robot resources in auction-based task allocations are analyzed with an illustration shown in Fig. 5, where $\mathscr{R}_I$ is the set of robots which have enough resources to complete task j, and $\mathscr{R}_K$ is the set of robots which do not have enough resources to complete the task. For the sake of simplicity, it is assumed that the robots use one resource which is consumed while they travel around, and the task is completed when the robot arrives at the task location. In this section, two types of auction algorithms are compared and analyzed. In the first case, the auction algorithms without considering the robot resource are analyzed. They are further divided into two cases depending on task reallocation. In the second case, the proposed algorithm is analyzed. For each algorithm, the winner's estimated task completion time and the estimated resource consumption to complete the task are respectively compared with the final task completion time and the total resource consumption of the robots.

4.1. Case 1: Without resource consideration

In case where the robots do not consider their resources in bid generation, they calculate their bids based on the estimated time to reach the task location. The bid is defined as the estimated task completion time of the robot such that the bid of robot u for task j, b_j^u, is defined as

$$b_j^u = t_D^u(d_{uj}),\quad u \in \mathscr{R}_U \tag{26}$$

where $\mathscr{R}_U = \mathscr{R}_I \cup \mathscr{R}_K$, d_{uj} is the distance between robot u and task j, and $t_D^u(d)$ is the estimated time for robot u to travel distance d.

The estimated resource consumption to complete task j is defined as $c_D^u(d_{uj})$. Note that both $t_D^u(d)$ and $c_D^u(d)$ satisfy superposition such that $t_D^u(d_1+d_2) = t_D^u(d_1)+t_D^u(d_2)$ and $c_D^u(d_1+d_2) = c_D^u(d_1)+c_D^u(d_2)$. From the bids of all the robots in $\mathscr{R}_U$, the winner for task j, robot $u^\star$ is defined as

$$u^\star = \arg\min_{u\in\mathscr{R}_U} b_j^u. \tag{27}$$

In case where the task is allocated to the winner in $\mathscr{R}_I$, i.e., $u^\star \in \mathscr{R}_I$, then $u^\star$ equals to $i^\star$ $(= \arg\min_{i\in\mathscr{R}_I} b_j^i)$. Since robot $i^\star$ can perform the task without refill, it directly goes to the task with distance $d_{i^\star j}$. As a result, the final task completion time, t_f, and the final resource consumption, c_f, are respectively calculated as

$$\begin{aligned} t_f &= t_D^{i^\star}(d_{i^\star j}) \\ c_f &= c_D^{i^\star}(d_{i^\star j}). \end{aligned} \tag{28}$$

Thus, the difference between the estimated task completion time and the final task completion time, Δt, and the difference between the estimated resource consumption and the final resource consumption, Δc, are respectively calculated as

$$\begin{aligned} \Delta t &= t_f - t_D^{i^\star}(d_{i^\star j}) = 0 \\ \Delta c &= c_f - c_D^{i^\star}(d_{i^\star j}) = 0. \end{aligned} \tag{29}$$

The results imply that if the winner belongs to $\mathscr{R}_I$, its bid is credible since the estimated task completion time and the resource consumption are equal to the final results in (28). However, it does not guarantee that the winner belongs to $\mathscr{R}_I$ all the time since the robots do not consider their resources in bid generation phase. The simple solution to get the winner from $\mathscr{R}_I$ is to let each robot to simulate performing the task considering its resource, and let the one to bid on the task if it has succeeded in completing the task in the simulation without refill. A major drawback of this approach is that if none of them has enough resource to complete the task, i.e., $\mathscr{R}_I = \emptyset$, no one can bid on the task so that the task would never be completed.

In case where the winner is in $\mathscr{R}_K$, i.e., $u^\star = k^\star \in \mathscr{R}_K$, it should redirect itself from the task to the station at point P_S. There are two possible ways to deal with this situation. One way is to complete the task after refill without reallocation, and the other way is to reallocate the task to the other robot.

4.1.1. Case 1-1: Without task reallocation

If robot $k^\star$ does not reallocate task j and continues to perform it after refill, t_f and c_f are respectively calculated as

$$\begin{aligned} t_f &= t_D^{k^\star}(d'_{k^\star j} + d'_{k^\star S} + d_{Sj}) + t_S^{k^\star} \\ c_f &= c_D^{k^\star}(d'_{k^\star j} + d'_{k^\star S} + d_{Sj}) \end{aligned} \tag{30}$$

where $d'_{k^\star j}$ $(0 \le d'_{k^\star j} \le d_{k^\star j})$ is the distance from the initial position of robot $k^\star$ to P_S, $d'_{k^\star S}$ is the distance from P_S to the station, d_{Sj} is the distance from the station to task j, and $t_S^{k^\star}$ is the resource refilling time of robot $k^\star$ at the station. The distances and P_S are shown in Fig. 5. The point P_S is the location at which the robot redirects itself to the refill station due to the depletion of its resource. The refill-transition time represents the time that a robot takes to change its state from *Perform* to *Refill* such that $t_D^{k^\star}(d'_{k^\star j})$ is the refill-transition time in this example. The refill-transition resource consumption is defined as the amount of resource that the robot spends during the refill-transition time such that $c_D^{k^\star}(d'_{k^\star j})$ is the refill-transition resource consumption. The difference between t_f and $t_D^{k^\star}(d_{k^\star j})$, Δt, is then calculated as

$$\begin{aligned} \Delta t &= t_f - t_D^{k^\star}(d_{k^\star j}) \\ &= t_D^{k^\star}(d'_{k^\star j} + d'_{k^\star S} + d_{Sj} - d_{k^\star j}) + t_S^{k^\star}. \end{aligned} \tag{31}$$

Since $d_{k^\star S} \le d'_{k^\star j} + d'_{k^\star S} \le d_{k^\star j} + d_{Sj}$ and $d_{k^\star S} + d_{Sj} - d_{k^\star j} \ge 0$, the range of Δt is defined as

$$t_D^{k^\star}(d_{k^\star S} + d_{Sj} - d_{k^\star j}) + t_S^{k^\star} \le \Delta t \le t_D^{k^\star}(2d_{Sj}) + t_S^{k^\star}. \tag{32}$$

Similarly, the range of Δc is defined as

$$c_D^{k^\star}(d_{k^\star S} + d_{Sj} - d_{k^\star j}) \le \Delta c \le c_D^{k^\star}(2d_{Sj}). \tag{33}$$

The results in (32) and (33) imply that if the winner belongs to $\mathscr{R}_K$, its bid is no more reliable such that the final task completion time and the resource consumption are larger than the winner's estimations due to the refill-transition time and the refill-transition resource consumption of the winner.

4.1.2. Case 1-2: With task reallocation

If the winner belongs to $\mathscr{R}_K$ and it can reallocate its task when its residual resource goes below the threshold, it will reallocate the task to the new winner and then goes to the station for refill. However, it is unknown whether the new winner has enough resource to complete the task. In this case, unlike the previous cases in which only one robot is involved in the task, the final task completion time and the final resource consumption may vary by the number of winners in $\mathscr{R}_K$. Since this case assumes that the winner belongs to $\mathscr{R}_K$, the cardinality of $\mathscr{R}_K$, $|\mathscr{R}_K|$ is equal to or greater than one. Based on this, the ranges of t_f and c_f can be respectively calculated as

$$\begin{aligned} t_D^{k_1^\star}(d'_{k_1^\star j}) + t_D^{i^\star}(d_{i^\star j}) \le t_f \le \sum_{m=1}^{|\mathscr{R}_K|} t_D^{k_m^\star}(d'_{k_m^\star j}) + t_D^{i^\star}(d_{i^\star j}) \\ c_D^{k_1^\star}(d'_{k_1^\star j} + d'_{k_1^\star S}) + c_D^{i^\star}(d_{i^\star j}) \le c_f \le \\ \sum_{m=1}^{|\mathscr{R}_K|} c_D^{k_m^\star}(d'_{k_m^\star j} + d'_{k_m^\star S}) + c_D^{i^\star}(d_{i^\star j}) \end{aligned} \tag{34}$$

where $k_m^\star$ represents the mth winner in $\mathscr{R}_K$ and $i^\star$ denotes the final winner from $\mathscr{R}_I$. From (34), the ranges of Δt and Δc are respectively calculated as

$$\begin{aligned} t_D^{k_1^\star}(d'_{k_1^\star j}) \le \Delta t \le \sum_{m=1}^{|\mathscr{R}_K|} t_D^{k_m^\star}(d'_{k_m^\star j}) \\ c_D^{k_1^\star}(d'_{k_1^\star j} + d'_{k_1^\star S}) \le \Delta c \le \sum_{m=1}^{|\mathscr{R}_K|} c_D^{k_m^\star}(d'_{k_m^\star j} + d'_{k_m^\star S}). \end{aligned} \tag{35}$$

Similar to the results from the auction algorithm without reallocation, the results in (35) show that reallocating the task when the winner runs out of the resource can cause more time and resource spendings than the winner's estimations due to the refill-transition time and the refill-transition resource consumption of the winner.

4.2. Case 2: Proposed approach

In the case of the proposed approach, the estimated bid of robot u for task j, $\hat{b}_j^u$, is defined as

$$\hat{b}_j^u = \hat{c}_j^u = P_j^u(1)\, c_j^u(A_0^u, 1) + P_j^u(2)\, c_j^u(A_0^u, 2) \tag{36}$$

with

$$\begin{aligned} c_j^u(A_0^u, 1) &= t_D^u(d_{uj}) \\ c_j^u(A_0^u, 2) &= t_D^u(d_{uS} + d_{Sj}) + t_S^u \end{aligned} \tag{37}$$

where $c_j^u(A_0^u, k)$ represents the cost of robot u for task j with path k from its current location, $\hat{c}_j^u$ is the estimated cost from (19), and $P_j^u(k)$ is the path selection probability of path k from (17). The path

1 represents the direct path from robot u to task j, and path 2 is the combination of the path from robot u to the station and the path from the station to task j. As shown in Fig. 5, the distances of path 1 and path 2 are d_{uj} and $d_{uS} + d_{Sj}$, respectively. Since the robots take account of their own resources in the bid generation process, the robots in $\mathscr{R}_I$ ($\mathscr{R}_K$) would get higher probability to select path 1 (path 2) than that of path 2 (path 1). For simplicity, it is assumed that the probability of the robots in $\mathscr{R}_I$ ($\mathscr{R}_K$) to select path 1 is one (zero), and the probability of them to select path 2 is zero (one). Based on the assumption, the estimated task completion time and the estimated resource consumption of the robot in $\mathscr{R}_I$ are calculated as $t_D^i(d_{ij})$ and $c_D^i(d_{ij})$, respectively. In the case when the robot is in $\mathscr{R}_K$, it plans to go directly to the station for refill and then goes to the task location, the estimated task completion time and the estimated resource consumption are respectively calculated as $t_D^k(d_{kS} + d_{Sj}) + t_S^k$ and $c_D^k(d_{kS} + d_{Sj})$. This implies that the refill-transition time and the refill-transition resource consumption are zero in the proposed approach since the robot in $\mathscr{R}_K$ directly heads to the station from its current location.

From the estimated bids of the robots in $\mathscr{R}_I$ and $\mathscr{R}_K$, the winner of task j is selected as

$$u^\star = \arg\min_{u \in \mathscr{R}_U} \hat{b}_j^u. \tag{38}$$

Thus, the robot that has the minimum estimated bid becomes a winner regardless of whether it belongs to $\mathscr{R}_I$ or $\mathscr{R}_K$. As a result, the estimated task completion time and the estimated resource consumption are determined as $\min(t_D^{i^\star}(d_{i^\star j}), t_D^{k^\star}(d_{k^\star S} + d_{Sj}) + t_S^{k^\star})$ and $\min(c_D^{i^\star}(d_{i^\star j}), c_D^{k^\star}(d_{k^\star S} + d_{Sj}))$, respectively.

Since the refill-transition time and the refill-transition resource consumption are zero in the proposed approach, both Δt and Δc are respectively determined as

$$\begin{aligned} \Delta t &= t_f - \min(t_D^{i^\star}(d_{i^\star j}), t_D^{k^\star}(d_{k^\star S} + d_{Sj}) + t_S^{k^\star}) = 0 \\ \Delta c &= c_f - \min(c_D^{i^\star}(d_{i^\star j}), c_D^{k^\star}(d_{k^\star S} + d_{Sj})) = 0. \end{aligned} \tag{39}$$

These results demonstrate that the robots with the proposed approach are able to estimate the task completion time and the resource consumption correctly regardless of whether they belong to $\mathscr{R}_I$ or $\mathscr{R}_K$.

5. Results

The proposed algorithm is tested in a simulation of delivery mission where robots should deliver goods to dispersed households. The goal of the mission is to minimize the average waiting time of delivering the requested goods along with minimizing unnecessary resource consumption and communications. The households that request the delivery service are randomly generated at any time in the simulation. Thus, the robots do not have any prior information about which household would request the delivery task and at what time. The resources that the robots consider are gas, battery, and goods. While moving around and delivering goods, robots consume gas, and battery, and the number of goods that the robots are carrying decreases as they deliver the goods. The simulation environment consists of 72 households, located as a 9×8 grid form and the three stations, i.e., gas, battery, and goods stations as shown in Fig. 6, where the circles at the corners of the grid represent the households. In the simulations, the proposed algorithm was compared with three different auction algorithms: the sequential single-item auction (SSIA) [31–33], the repeated sequential single item auction (RSSIA) [34], and the consensus-based bundle algorithm (CBBA) [16]. The simulations were carried out 50 times for each algorithm with a different number of robots, i.e., 3, 5, 7, 9, and 11 robots.

Fig. 7 shows the number of completed tasks for each algorithm, and the robots with the proposed algorithm were able to perform

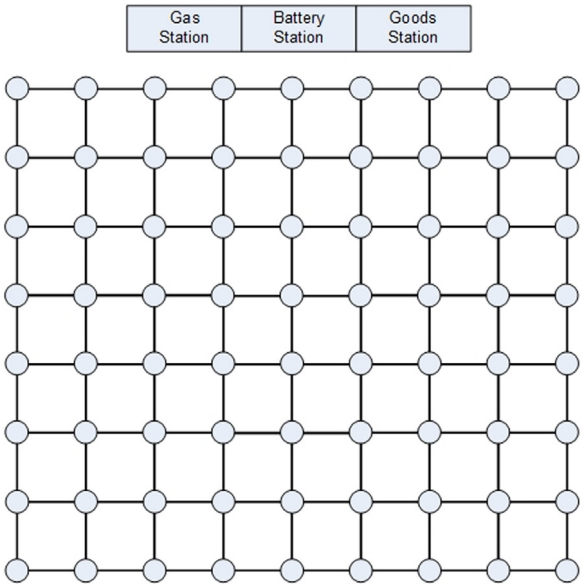

Fig. 6. The simulation environment of the delivery mission. The circles in a 9×8 grid form represent households and the three rectangles above the grid form represent the gas, battery, and goods stations where the robots recharge their resources.

more tasks than those with other algorithms in all cases of the robot numbers. The result implies that the proposed algorithm enables the robots to reduce their time for managing their resources, such as visiting the stations and waiting for refill, since they only spent their time either for performing the tasks or for managing their resources. To verify the implication, the resource management time with a different number of robots were measured as shown in Fig. 8. The resource management time represents the average time a robot spends to maintain its resources, such as visiting the stations and waiting for refill, to complete a task. As shown in Fig. 8, the resource management time of the proposed algorithm was the shortest among all algorithms for all different number of robots. Since the resource management time decreases as the robots consume less resources during the mission, the result gives an implication that the proposed algorithm leads the robots to use their resources more efficiently during the mission.

To compare the efficiency of the resources among the algorithms, the normalized resource consumption $\bar{c}$ was defined as

$$\bar{c} = \frac{1}{N_L} \sum_{l=1}^{N_L} \sum_{i=1}^{N_R} \frac{c_l^i}{C_l^{max}} \tag{40}$$

where N_L is the number of resources, N_R is the number of robots, C_l^{max} is the maximum value of resource l for normalization, and c_l^i is the total consumption of the lth resource of robot i during the mission. Fig. 9 shows the normalized resource consumption with a different number of robots where the robots with the proposed algorithm consumed less resources than those of the other algorithms. This is because the proposed algorithm enables the robots to consider their resources during the task allocation. As a result, if one of the expected resources after completing the task was lower than the threshold, it means that they visited stations first before starting the task. For the other algorithms, however, the robots did not consider their resources. Thus, they had to visit the stations more often on their way to the task, and this caused a waste of time and unnecessary resource consumption during the mission.

Fig. 10 shows the average communications of the robots per task. In the case of the SSIA and RSSIA, since the robots did not consider their resources, they had to auction their tasks again

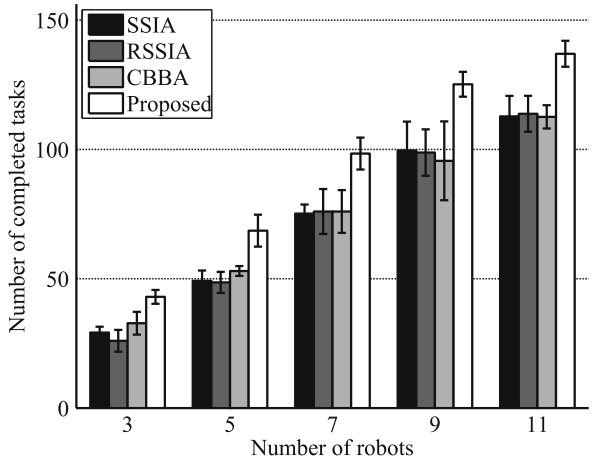

Fig. 7. The number of completed tasks with a different number of robots.

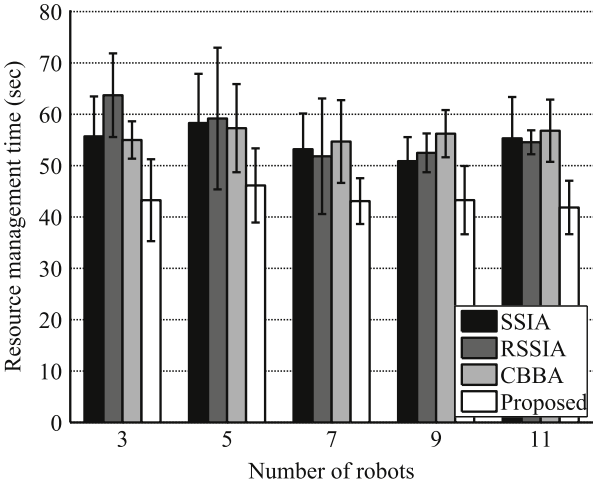

Fig. 8. The resource management time with a different number of robots.

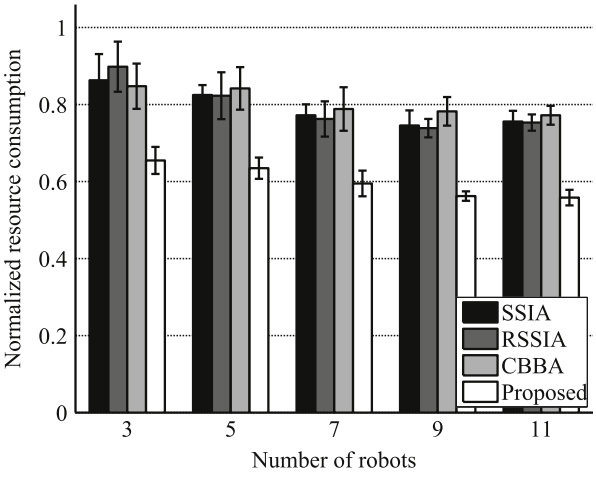

Fig. 9. Normalized resource consumption with a different number of robots.

whenever one of their resources were lower than its threshold. In the case of the CBBA, since each robot should continuously broadcast the winners along with their bids for all the tasks, it also required more communications than the proposed algorithm.

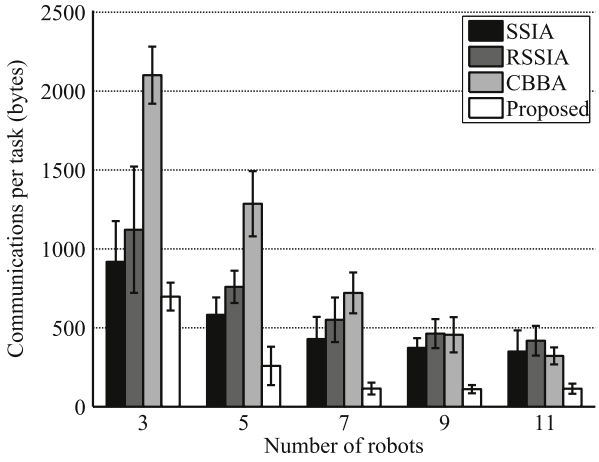

Fig. 10. Average communications per task with a different number of robots.

Compared to the other algorithms, the robots with the proposed algorithm were able to reduce the communications since the robots bid on the tasks only when their bids were higher than the bids of the winners.

6. Conclusion

This paper presented a resource-based task allocation algorithm for multirobot coordination. The effect of the robot resources in the auction-based approach was considered, and the proposed approach was shown to be able to minimize both the task completion time as well as the resource consumption. The effectiveness of the proposed algorithm was demonstrated by the simulation of the delivery mission, and the results showed that the proposed algorithm minimizes the task completion time, requires less communication among robots, and results in the efficient consumption of the robots' resources.

Acknowledgments

This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2014-0-00639) supervised by the IITP (Institute for Information & communications Technology Promotion), and Kumoh National Institute of Technology.

References

[1] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, H. Younes, Coordination for multi-robot exploration and mapping, Proc. Natl. Conf. Artif. Intell. (2000) 852–858.
[2] A. Kolling, S. Carpin, Multi-robot surveillance: An improved algorithm for the graph-clear problem, Proc. IEEE Conf. Robot. Autom. (2008) 2360–2365.
[3] B. Doroodgar, M. Ficocelli, B. Mobedi, G. Nejat, The search for survivors: Cooperative human–robot interaction in search and rescue environments using semi-autonomous robots, Proc. IEEE Conf. Robot. Autom. (2010) 2858–2863.
[4] X. Liang, Y. Xiao, Studying bio-inspired coalition formation of robots for detecting intrusions using game theory, IEEE Trans. Syst. Man, Cybern. B (2010) 683–693.
[5] M.B. Dias, R. Zlot, N. Kalra, A. Stentz, Market-based multi-robot coordination: A survey and analysis, Proc. IEEE (2006) 1257–1270.
[6] R. Zlot, A. Stentz, Multirobot control using task abstraction in a market framework, Proc. Int. Conf. Collab. Tech. Allian (2003).
[7] L.E. Parker, Distributed intelligence: Overview of the field and its application in multi-robot systems, J. Phys. Agent. (2008) 5–14.
[8] S. Sariel, T. Balch, Efficient bids on task allocation for multi-robot exploration, Proc. Int. Conf. Florida Artif. Intell. Res. Soc. (2006) 116–121.
[9] Y. Mei, Y.H. Lu, Y.C. Hu, C.G.S. Lee, Deployment of mobile robots with energy and timing constraints, IEEE Trans. Rob. (2006) 507–522.

[10] D.-H. Lee, J.H. Han, J.-H. Kim, A preference-based task allocation framework for multi-robot coordination, Proc. IEEE Conf. Robot. Bio. (2011).
[11] B.P. Gerkey, M.J. Mataric, A formal analysis and taxonomy of task allocation in multi-robot systems, Int. J. Robot. Res. (2004) 939–954.
[12] S. Sariel, T. Balch, J.R. Stack, Real time auction based allocation of tasks for multi-robot exploration problem in dynamic environments, AAAI Workshop, Technical Report WS-05-06, 2005, 27–33.
[13] B.P. Gerkey, M.J. Mataric, Sold!: Auction methods for multirobot coordination, IEEE Trans. Robot. Autom. (2002) 758–768.
[14] S.L. Smith, F. Bullo, Target assignment for robotic networks: Asymptotic performance under limited communication, Proc. Amer. Control Conf. (2007) 1155–1160.
[15] S.L. Smith, F. Bullo, Monotonic target assignment for robotic networks, IEEE Trans. Autom. Control. (2009) 2042–2057.
[16] H.L. Choi, L. Brunet, J.P. How, Consensus-based decentralized auctions for robust task allocation, IEEE Trans. Robot. (2009) 912–926.
[17] F. Dressler, G. Fuchs, Energy-aware operation and task allocation of autonomous robots, Proc. IEEE Workshop Robot Motion Control (2005) 163–168.
[18] B. Bethke, M. Valenti, J.P. How, An experimental demonstration with integrated health monitoring, IEEE Robot. Autom. Mag. (2008) 39–44.
[19] M. Valenti, B. Bethke, J.P. How, D.P. Farias, J. Vian, Embedding health management into mission tasking for UAV teams, Proc. Amer. Control Conf. (2007) 5777–5783.
[20] A. Couture-Beil, R.T. Vaughan, Adaptive mobile charging stations for multi-robot systems, Proc. IEEE Conf. Intel. Rob. Syst. (2009) 1363–1368.
[21] P. Zebrowski, R.T. Vaughan, Recharging robot teams: A tanker approach, Proc. Int. Conf. Adv. Rob. (2005) 803–810.
[22] A. Munoz, F. Sempe, A. Drogoul, Autonomous robots sharing a charging station with no communication: a Case Study, Dist. Auton. Rob. Syst. (2002) 91–100.
[23] F. Michaud, E. Robichaud, Sharing charging stations for long-term activity of autonomous robots, Proc. IEEE Conf. Rob. Syst. (2002) 2746–2751.
[24] A. Beynier, A.-I. Mouaddib, Decentralized Markov decision processes for handling temporal and resource constraints in a multiple robot system, Distrib. Auton. Robot. Syst. (2007) 191–200.
[25] M. Strens, N. Windelinckx, Combining planning with reinforcement learning for multi-robot task allocation, Proc. Adap. Agents Multi-Agent Syst. (2005) 260–274.
[26] B. Kaleci, O. Parlaktuna, Performance analysis of bid calculation methods in multirobot market-based task allocation, Turkish J. Elect. Eng. Comput. Sci. (2013) 565–585.
[27] D.-H. Lee, S.A. Zaheer, J.-H. Kim, A resource-oriented, decentralized auction algorithm for multi-robot task allocation, IEEE Trans. Autom. Sci. Eng. (2015) 1469–1481.
[28] D.-H. Lee, S.A. Zaheer, J.-H. Kim, Ad-hoc network-based task allocation with resource-aware cost generation for multi-robot systems, IEEE Trans. Ind. Elec. (2014) 6871–6881.
[29] M.G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, S. Jain, Auction-based multi-robot routing, Robot.: Sci. Syst. (2005) 343–350.
[30] C. Tovey, M.G. Lagoudakis, S. Jain, S. Koenig, The generation of bidding rules for auction-based robot coordination, Proc. Multi-Robot Syst. Workshop (2005) 1–12.
[31] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, S. Jain, The power of sequential single-item auctions for agent coordination, Proc. Natl. Conf. Artif. Intell. (2005) 1625–1629.
[32] X. Zheng, S. Koenig, C. Tovey, Improving sequential singleitem auctions, Proc. IEEE Conf. Intell. Robot. Syst. (2006) 2238–2244.
[33] P.B. Sujit, R. Beard, Distributed sequential auctions formultiple UAV task allocation, Proc. Amer. Control Conf. (2007) 3955–3960.
[34] M. Nanjanath, M. Gini, Repeated auctions for robust task execution by a robot team, Robot. Auton. Syst. (2010) 900–909.

Dong-Hyun Lee received the Ph.D. degree in electrical engineering from KAIST, Daejeon, Korea, in 2015. Since 2016, he has been with the Department of IT Convergence Engineering, Kumoh National Institute of Technology.