Appl Intell
DOI 10.1007/s10489-016-0771-5

Dynamic task allocation for multi-robot search and retrieval tasks

Changyun Wei[1] · Koen V. Hindriks[2] · Catholijn M. Jonker[2]

Abstract Many application domains require search and retrieval, which is also known in the robotic domain as foraging. For example, in a search and rescue domain, a disaster area needs to be explored and transportation of survivors to a safe area needs to be arranged. Performing such a search and retrieval task by more than one robot increases performance if they are able to distribute their workload efficiently and evenly. In this work, we study the Multi-Robot Task Allocation (MRTA) problem in the search and retrieval domain, where a team of robots is required to cooperatively search for targets of interest in an environment and also retrieve them back to a home base. In comparison with typical foraging tasks, we look at a more general search and retrieval task in which the targets are distinguished with various types, and task allocation also requires taking into account temporal constraints on the team goal. As usual, robots have no prior knowledge about the location of targets in the environment but in addition they need to deliver targets to the home base in a specific order according to their types, which significantly increases the complexity of a foraging problem. We first use a graph-based model to analyse the search and retrieval problem and the dynamics of exploration and retrieval within a cooperative team. We then proceed to present an extended auction-based approach, as well as a prediction approach. The essential difference between these two approaches is that the task allocation and execution procedures in the auction approach are running in parallel, whereas a robot in the prediction approach only needs to choose a task to perform when it has no thing to do. The auction approach uses a winner determination mechanism to allocate tasks to each robot, whereas the robots in the prediction approach implicitly coordinate their activities by team reasoning that leads to consensuses about task allocation. We use the Blocks World for Teams (BW4T) simulator to evaluate the two approaches in our experimental study.

Keywords Multi-robot teamwork · Search and retrieval · Task allocation · Coordination

✉ Changyun Wei
weichangyun@hotmail.com

Koen V. Hindriks
k.v.hindriks@tudelft.nl

Catholijn M. Jonker
c.m.jonker@tudelft.nl

[1] College of Mechanical and Electrical Engineering, Hohai University, NO.200 Jinling Bei Road, Changzhou 213022, People's Republic of China

[2] Interactive Intelligence Group, Delft University of Technology, Mekelweg 4, Delft, The Netherlands

1 Introduction

Robot teams are expected to perform more complicated tasks that consist of multiple subtasks, and the subtasks may need to be completed concurrently or in sequence [15]. In this paper, we study the Multi-Robot Task Allocation (MRTA) problem in the search and retrieval (also called foraging) domain. Foraging is a canonical task for studying multi-robot teamwork [4, 5, 19, 30] in which a team of robots needs to search an environment for targets

Published online: 10 March 2016

of interest which need to be retrieved and brought back to a home base. The search and retrieval tasks can be motivated by many practical applications such as large-scale urban search and rescue robots [8], deep-sea mineral mining robots [31] and order picking robots for autonomous warehouses management [13].

Foraging has in particular been studied in various bio-inspired, swarm-based approaches in the literature [4, 19], where, typically, robots minimally interact with one another as in [4]. If they communicate explicitly, only basic information such as the locations of targets or their own locations are exchanged [25]. Most of the work has studied a basic foraging task where the targets to be collected are not distinguished, which reduces the need for cooperation and coordination. In contrast, we study a more general foraging problem where the targets are distinguished by various types. Moreover, we also assume that *temporal* constraints (also called ordering constraints throughout this work) may be present that require targets to be delivered to the home base in a specific order. Ordering constraints on the type of targets are useful for modelling practical applications, for example, how urgently a victim needs assistance, how valuable a mining resource is, or how urgently a package is needed.

The use of multiple robots may yield significant performance gains compared to the performance of a single robot [5, 10]. Realising such performance gains, however, poses a serious challenge for a robot team and requires effective coordination strategies for task allocation. Task allocation has been extensively addressed in various multi-agent/robot systems over the past few years, with the aim of finding an allocation of tasks to the robots so as to minimise the overall team cost. In general, however, even when the locations of targets are initially known and only an optimal solution for a multi-robot routing problem [16, 21, 32] needs to be found, the problem is NP-hard [7, 16, 21, 34]. The task allocation problem that we study here, moreover, is also harder than the multi-robot exploration problem studied in [3, 7, 27], in which the robots only need to search and locate the targets but do not need to deliver them back to a home base. In other words, multi-robot exploration is often studied without handing objects, i.e., the retrieval of an object from a location to a destination is not considered.

Many approaches to MRTA problems are based on market-based solutions that use auctions as such solutions are robust and can be performed in real-time [2, 33], but fully decentralized auction-based approaches can be very complex in design and implementation and have a high communication overhead [9, 16]. The first contribution of this work is that we present an auction-inspired approach extended from standard Sequential-Single-Item (SSI) auctions. In comparison with other auctions, the work [16, 21] has shown that the standard SSI auctions can provide a good compromise between computational complexity and solution quality if the set of tasks is initially known. The standard SSI auctions have been used to solve the *static* task allocation problem in the multi-robot routing domain as in [16, 21].

The second contribution of this work is that we in addition propose a prediction approach, in which the robots can predict what the others will do to reach the consensuses about task allocation, without using auctions or negotiations. More specifically, the robots have a tacit understanding of which task should be performed by which robot by team reasoning. The idea is that the most cost-effective robot is delegated to complete each task in a team. In comparison with the extended auction approach, this approach, in particular, allows the robots to employ a non-greedy strategy to explore the environment in a way that a robot has lower likelihood to explore faraway locations and higher likelihood to explore nearby ones.

We carry out an experimental study in a simulator to evaluate the two proposed approaches discussed above.

This paper is organised as follows. We discuss the state-of-the-art literatures in Section 2, and present a formal model of the search and retrieval problem that we deal with in this paper in Section 3. The auction-inspired approach is discussed in Section 4, and the prediction approach is presented in Section 5. In Section 6, we discuss the experiments in a simulated environment and analyse the results. Finally, we conclude this work in Section 7.

2 Related work

2.1 Multi-robot search and retrieval

Multi-robot search and retrieval tasks are more complex than multi-robot routing and pure exploration tasks. As the locations of targets in routing problems are given [16], it can be considered a variant of the multiple traveling salesman problem [1]. This is why *offline planning* methods can be used to find a reasonable approximation, though the computation still might be inefficient [1]. In contrast, the targets to be collected in search and retrieval impose *spatial* constraints, i.e., the robots do not have prior knowledge about the distribution of the targets in an environment. The information needed for offline planning is not initially available.

The solutions to typical exploration problems cannot be directly applied to search and retrieval problems, though the robots need to explore the environment. This is because typical exploration problems are studied without handing objects, i.e., the retrieval of an object from a source to a destination is not considered. The objective of task allocation for exploration problems is to assign a subset of

locations to each robot that they need to visit in a way that minimises the total travel cost. Thus, similar to routing problems, offline planning methods can also be used to find a possible solution. In contrast, search and retrieval robots have to consider not only where to explore but also when to explore because the robots may need to switch their working mode between exploring targets and retrieving targets. It means that exploration and retrieval need to be interleaved somehow in search and retrieval, e.g., after checking a location, a robot does not know whether it should go to explore another location or deliver an object found at the location. Thus, the task allocation for search and retrieval is *dynamic*, and the robots cannot complete the assignment of all the tasks before exploring the environment.

2.2 Task allocation

Auction-based approaches have been proposed to solve the MRTA problem in [16, 17, 21, 26, 27, 32]. In general, these auction approaches can be categorised into *parallel*, *combinatorial* and *sequential* auctions [16, 17, 26]. In combinatorial auctions, each robot bids on a subset of targets with the objective of minimising the path cost from its current location [17]. Often, a central auctioneer is used to determine the winner of auctions. Theoretically, as each robot can enumerate all possible combinations[1] of the subsets of tasks to bid on, the combinational auctions could provide optimal solutions to *static* task allocation problems where the tasks are known at the beginning of auctions. In practice, it is impossible to do so because the combinations of possible subsets of tasks can be exponential in the number of tasks.

In parallel auctions, every robot needs to bid for each available task in an auction round in parallel, and the auctioneer allocates each task to the robot who has submitted the smallest bid. Thus, all the available tasks can be allocated immediately in a round. In comparison with combinatorial auctions, the computation and implementation of such auctions would be efficient, but the problem is that the performance of robots is likely to be suboptimal because such an auction does not consider any synergies between targets [17, 26]. In other words, it does not take account of the topological distribution between two targets, which has a big influence on the actual travel cost of a robot. In addition, it should be noted that parallel auctions can only be applied in static problems in which the tasks are known in advance.

[1] As described in [16, 17], each robot can consider all positive and negative synergies, in which two targets have positive (negative) synergy for a robot if the minimum travel cost for visiting them is smaller (larger) than the simple sum of distances from the robot's current location to each target.

The Sequential-Single-Item (SSI) auctions provide a good compromise between computational complexity and solution quality for the problems where the set of tasks is initial known [16, 21]. In SSI auctions, the tasks are allocated through a multi-round auction, in which each robot bids on only one task in each round. In this work, we extend the existing SSI auction approach, discussed in [16, 21] for the routing problems. In order to adapt to *dynamic task allocation for search and retrieval*, the extended SSI auction approach also adopts some strategies from the work [26].

In SSI auction approaches, whenever the auctioneer initiates a new round of auctions, each robot can bid for one of available tasks. It means that each robot may have multiple allocated tasks at a moment, and in order to distribute the workload more evenly in a team, each robot can consider previously allocated tasks when bidding a new one.

In SSI auctions, a robot always needs to bid for the task with the smallest cost. Otherwise, it is hard for the robot to win a task in an auction round because the auctioneer allocates a task to the robot who made the lowest bid in an auction round [16]. Thus, the robots may put more effort into local minima to explore targets in nearby locations, due to the hill-climbing nature of greedy bidding strategies. This is why we propose the prediction approach, in which we do not clearly separate task execution from task allocation, and each robot selects a task to execute only when it has nothing to do. The approach aims at providing the robots with the possibility to explore faraway locations, but they more likely will choose nearby ones to explore.

The task allocation problem has also been addressed in literature by utility-based approaches for both cooperative robot teams and robot swarms. For example, each task is assigned to a robot based on various utility estimates, such as acquiescence and impatience [24], sensor relevance [35], net energy [22], positive and negative feedback assigned to the robot based on historical performance [20], coalition values [6], belief on the ability of the robot to perform a task [11], and costs [28]. Research in swarm robots for foraging focuses mostly on how the robots interact and cooperate to perform tasks, without handing objects: the retrieval of objects from a source to a destination is not considered, or simply abstracted into a trip between the two locations. Thus, typical task allocation approaches cannot be used to solve the search and retrieval problem that we study in this paper.

3 Multi-robot task allocation for search and retrieval

In this section, we first present a formal model of the search and retrieval problem that we study here, and then use it to precisely formulate the problem.

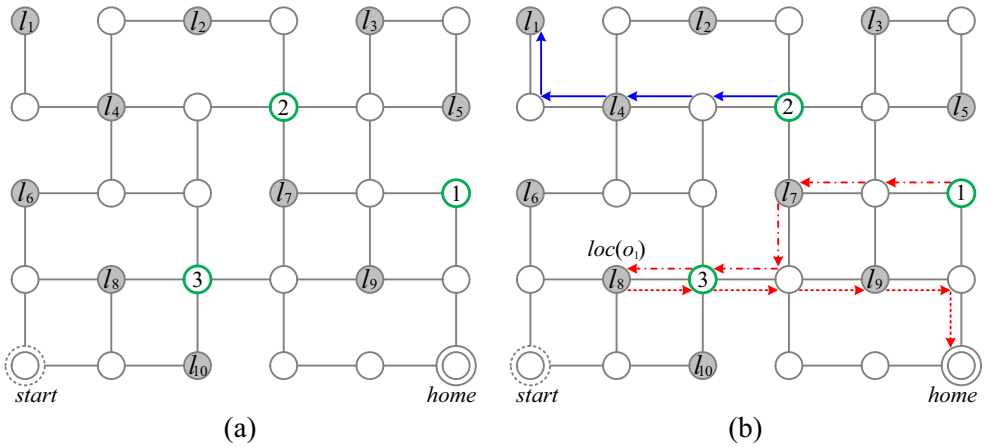


Fig. 1 Graph models of search and retrieval tasks and the estimated costs

3.1 Model

Our model of the task allocation for multi-robot search and retrieval is based on and extends [18, 21, 23] where a model of the task allocation for multi-robot routing is presented that requires robots to only visit target locations. We extend this model by adding retrieval and delivery tasks for target items. We use $Agt = \{1, 2, \ldots, k\}$ to denote the k robotic agents that are available for search and retrieval. We use an undirected graph $\mathscr{G}(\mathscr{V}, \mathscr{E})$ with a non-empty set of vertices $\mathscr{V} = \{v_1, v_2, \ldots, v_p\}$ and a set of edges $\mathscr{E}$ connecting vertices for representing the robots' environment, see Fig. 1a. Edges are assumed to have unit length.

In the environment a non-empty set of objects $O = \{o_1, o_2, \ldots, o_n\}$ is distributed. In different application contexts an object could be, for example, a victim in a search and rescue context, a resource to be mined in a mining context, or a package to be picked up in a warehouse context. These objects are located on a subset of vertices $L = \{l_1, l_2, \ldots, l_q\} \subseteq \mathscr{V}$ called *target locations*. We allow that no, one, or multiple objects are located at a target location, i.e., a vertex in L. We use $loc(o)$ to denote the location of an object o. Because objects initially can only be located at target locations, we have that $\bigcup_{o \in O} loc(o) \subseteq L$ in the beginning. For reasons of simplicity, we assume that robots need to deliver objects to a single home base.

An important difference which sets our work apart from that of others is that we explicitly model *object types*. As mentioned above, object types are useful for modelling the application context. In our model we abstract from the specific features of a domain and assume that objects can be differentiated by means of their color. Object types allow us to model ordering constraints on objects that need to be retrieved from the environment. We can say, for example, that a red object needs to be retrieved before a blue one. We thus study a more general search and retrieval problem here where types of objects that need to be retrieved can be distinguished from those that do not need to be retrieved, and types can be used to introduce ordering constraints. We use $type(o)$ to denote the type of object o.

The goal of the search and retrieval problem that we study here can be specified as a finite sequence of types $\langle \tau_1, \tau_2, \ldots, \tau_m \rangle$, i.e., colors of target objects that need to be retrieved. For example, $\langle$*red, blue, red, red, yellow, blue*$\rangle$ could be a goal. The idea is that the robots should search for objects in the environment of the right type and deliver them back to the home base in order. That is, the robots need to retrieve a sequence of m objects $\langle X_1, X_2, \ldots, X_m \rangle$ that match the sequence of types of the main goal, where X refers to an arbitrary object. In other words, we must have that for all $1 \leq i \leq m$:

$$type(X_i) = \tau_i. \tag{1}$$

It should be noted that in order for the robots to be able to successfully complete a search and retrieval task there must be enough objects in the environment of the right type to match the types needed to achieve the main goal. Over time, the sequence of types that needs to be delivered reduces if an object of the next matching type is delivered to the home base. We distinguish between three kinds of goals: (i) a goal such as $\langle red, red, red, red \rangle$ that requires all objects to be of the *same type*, i.e., $\tau_i = \tau_j$ for all i, j; (ii) a goal such as $\langle$*red, blue, yellow, white*$\rangle$ that requires objects to all have a *different type*, i.e., $\tau_i \neq \tau_j$ for any pair of indexes $i \neq j$, and (iii) a *mixed type* goal such as $\langle red, blue, red, yellow \rangle$ that requires some but not all objects to have different types, i.e., $\tau_i \neq \tau_j$ for some i, j. Note that a goal that has at least

one pair of indexes $i \neq j$ that $\tau_i = \tau_j$ must be a goal of type (i) or (iii).

Robots initially have no knowledge about the location of objects, and also do not know how many objects there are in total. They initially are assumed to know which locations are possible target locations where objects can be found. In order to locate objects, robots thus only need to explore these target locations.[2] Visiting a location to find out which objects are present at that location is called an *exploration task*. Exploration tasks can be identified with target locations.

Definition 1 An *exploration task* is a target location $l \in L$.

Exploration tasks need to be allocated to robots to execute, so the team will be able to find objects that are needed to achieve the team goal. The set of exploration tasks that have not been completed, i.e., have not been visited by any robot, is denoted by E. This set changes over time as follows. Initially, we have $E = L$ because none of the target locations have been visited. If a location l has been visited, that location is removed from E. The set E over time thus gets smaller but does not need to become empty before the team goal has been achieved; it may not be necessary to visit all target locations in order to find and deliver all the needed objects.

If an approach to the search and retrieval problem has a mechanism to *confirm the allocation* of exploration tasks, we use T_E^i to denote the exploration tasks that are allocated to robot i and use $T_E = \bigcup_{i \in Agt} T_E^i$ to represent the set of all allocated exploration tasks. Consequently, the robots can also know the currently available unallocated exploration tasks, denoted by U_E:

$$U_E = E \backslash T_E. \tag{2}$$

Once an allocated exploration task is completed, it will be removed from T_E.

To complete a search and retrieval task, the robots need to know what their team goal looks like. We assume that they know the goal sequence of types and understand what types of objects they need to retrieve from the environment and in which order these objects need to be retrieved. For both different and mixed type goals, it is important to understand the order in which objects need to be delivered, so we define retrieval tasks as pairs of objects o and indexes i into a goal sequence.

Definition 2 A *retrieval task* is a pair $\langle o, i \rangle$ where o is an object and i is an index into the goal sequence of types.

For each retrieval task we assume that $type(o) = \tau_i$ because it does not make sense to retrieve an object to match the i-th type in the main goal if the object type is different from τ_i. In other words, we assume that robots only perform retrieval tasks that at least potentially contribute to the overall goal. For instance, if the required target is a red box, the object that a robot should retrieve must also be a box of red color.

We use R to denote the set of all possible retrieval tasks that can be allocated at a particular time to a robot. This set changes over time as follows. Initially, we have $R = \emptyset$ because the robots initially do not know the location of any of the objects. If an object o is found and $\langle \tau_j, \ldots, \tau_m \rangle$ is the remaining goal sequence of types that still need to be delivered, all retrieval tasks $\langle o, i \rangle$ such that $type(o) = \tau_i$ for $j \leq i \leq m$ are added to R. An object thus is associated with all indexes of the same type and R can include multiple retrieval tasks for a single object. Because we can have multiple objects of the same type, it also can be the case that R includes more than one retrieval task for a particular index. If an object has been delivered to a home base that matches the type needed for the first index j that needs to be matched next, *all* retrieval tasks for that index are removed again from R. The set R thus includes all retrieval tasks that still might contribute to achieving the team goal. For example, if the team goal is $\langle red, blue, red \rangle$ and two red objects o_4 and o_5 are found at a moment, then the retrieval tasks will be $R = \{\langle o_4, 1\rangle, \langle o_4, 3\rangle, \langle o_5, 1\rangle, \langle o_5, 3\rangle\}$. If o_4 is delivered to the home base first, R is updated to $R = \{\langle o_5, 3\rangle\}$ because o_4 is not available any more and the first red object type in the goal has been matched.

We use T_R^i to denote the retrieval tasks allocated to robot i, and $T_R = \bigcup_{i \in Agt} T_R^i$ for the set of all allocated retrieval tasks. The allocated retrieval tasks that have been completed will be removed again from T_R. We can also get the currently available unallocated retrieval tasks denoted by U_R:

$$U_R = R \backslash T_R. \tag{3}$$

Depending on the solutions, a robot may have multiple allocated retrieval tasks at a moment, and if so, it should consider which object it should go to retrieve to fulfil the corresponding required target type. We also use $T^i = T_E^i \bigcup T_R^i$ to denote the set of exploration and retrieval tasks

[2] Note that only a subset of vertices in $\mathscr{G}$ indicates the possible locations of objects, which can be motivated by urban search and rescue applications. For instance, even though robots can have a map of a village, they may not know the precise locations of the survivors after an earthquake.

that have been allocated to robot i but still need to be completed.

3.1.1 Cost estimate for exploration tasks

We use $loc(i)$ to denote the current location of robot i. A robot is assumed to deliver objects to its home base $home(i)$. The cost function $cost_E(i, l)$ is used to indicate the travel costs for robot i to go to and explore a target location $l \in L$:

$$cost_E(i, l) = d(loc(i), l), \tag{4}$$

where d denotes the shortest cost for a robot to travel from one location to the other one.

Given a robot's location and the location that the robot wants to explore in the graph, we can calculate the shortest travel cost in (4) by performing a graph search, for example, using A^* algorithm. As shown in Fig. 1b, the estimated cost for robot 2 to explore l_1 takes 4 steps.

3.1.2 Cost estimate for retrieval tasks

We use the cost function $cost_R(i, r)$ to represent the shortest travel cost for robot i to complete a retrieval task $r = \langle o, j \rangle$ for a specific object o with $type(o) = \tau_j$:

$$cost_R(i, r) = d(loc(i), loc(o)) + d(loc(o), home(i)), \tag{5}$$

For instance, as shown in Fig. 1b, the estimated cost for robot 1 to collect object o_1 takes 10 steps. Note that the estimated cost using (4) or (5) is calculated according to the fact that robot i only has one exploration task or one retrieval task to execute from its current location. If a robot has multiple allocated tasks to perform, the estimated cost for a particular task may be affected by the other tasks, depending on the approaches used for dealing with the search and retrieval task allocation.

3.2 Problem formulation

Thesearch and retrieval problem that we study here is to find the most efficient solution to how a cooperative team of robots Agt can most efficiently locate and deliver objects needed to achieve a goal sequence $\langle \tau_1, \ldots, \tau_m \rangle$, where the τ_i refer to object types.

4 An auction-inspired task allocation approach

An auction-inspired coordination framework for multi-robot task allocation in the routing domain has been introduced in [16, 21, 32]. In these works, it is assumed that the robots already know the locations of the targets, and only need to visit these targets, but do not need to deliver them back to a home base. In this section, we extend the standard SSI auctions to an auction-inspired approach that is also able to handle *dynamic task allocation* for the search and retrieval problem with *ordering constraints*. We first briefly discuss the standard SSI auctions and then introduce our proposed extension.

4.1 Standard sequential-single-item auctions

Standard SSI auctions are designed for *static task allocation* problems, for example, in the context of multi-robot routing [16–18, 21], where all the tasks are known at the beginning of auctions. The tasks are allocated by a multi-round auction, in which each robot bids on *only one task in each round*, and a simple winner determination mechanism is used to allocate a task to the robot who *made the lowest bid*. The winner is typically determined by a central auctioneer, but a decentralized approach for winner determination is also possible [26]. SSI auctions can iteratively provide a complete solution to a problem, starting from a partial solution, though it is not guaranteed to find the optimal one.

When determining which task to bid on in a new round of auctions, each robot takes account of the tasks that have already been allocated to it in previous rounds because the cost for the robot to complete the new task depends on the tasks that it has already committed to. To determine which task to bid on, the MinMax team objective [18, 27] can be used to minimize the *maximum travel cost of the individual robots*. With the MinMax team objective, each robot bids the *total travel cost* for visiting both the targets allocated to it in previous rounds and the new target.

Figure 2 illustrates how a robot team uses the MinMax heuristic to bid for and allocate tasks by means of an example. We use a subgraph in Fig. 1a as the map of the environment to illustrate the details. In this example, robot 1 and 2 need to allocate locations l_1, l_5 and l_7 for exploration. The robots can obtain the estimated costs for each task using the map information.

In the first round, none of the locations have been allocated, so both robots can bid on all of these. Robot 2 will take 1, 4 or 2 steps to arrive at l_7, l_1 or l_5, respectively. As the robots bid for the task with the lowest cost, robot 2 will bid 1 for l_7. Likewise, robot 1 will bid 2 for l_7 because it takes 7 or 3 steps to go to l_1 or l_5. As a result, robot 2 wins the task in this round, i.e., l_7, as its bidding cost is the lowest.

In the second round, since l_7 has been allocated, the robots can only bid on l_1 or l_5. In this round robot 2 has to take into account its previous allocated tasks, i.e., l_7, when bidding on a new task. Consequently, its costs for l_1 will be $1 + 5 = 6$ (first move to l_7 then to l_1) and for l_5 will be $1 + 3 = 4$, and the robot will bid 4 for l_5. Robot 1 simply bids 3 for l_5, so it will win l_5 in this round. In the third round,

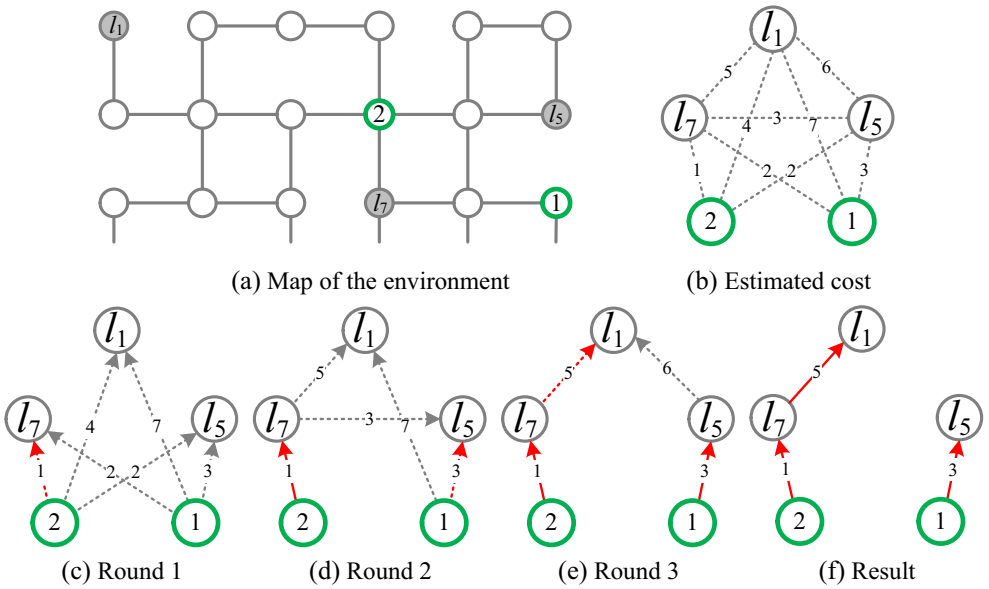


Fig. 2 With the MINMAX team objective, robots aim at minimizing the maximum cost that any of the individual robots will make

only l_1 still needs to be allocated and both robots have previously allocated tasks. As a result, robot 2 will bid $1 + 5 = 6$ for l_1, while robot 1 will bid $3 + 6 = 9$ for l_1, and robot 2 gets task l_1 assigned in this round. Finally, all the tasks are allocated.

To execute the allocated tasks, a robot is free to reorder tasks in any way that it wants to perform them, which is called *plan modification* [26]. Searching for an optimal execution order, however, is computationally prohibitive. Typically, a heuristic is used to determine where to insert a new task into the sequence of tasks that the robot is already committed to. Such a heuristic would determine the location where the robot should start performing a new task, if it would be allocated the task.

4.2 Extended SSI auctions for multi-robot search and retrieval

In standard sequential auctions, the tasks are known at the beginning of auctions, and, hence, such an approach cannot be directly applied to the search and retrieval problem in which retrieval tasks appear *dynamically* when target objects are located. The work [23] proposes a dynamic SSI auction approach to navigation tasks, focusing on the robustness of accomplishing the tasks. The robots thus are not expected to minimize the completion time in the sense that they only use current positions to bid on new tasks in each round, and the impact of the execution order of these tasks is not considered when choosing which allocated tasks to execute. In contrast, we in particular put effort into enhancing team performance, which means that we are concerned with optimizing the allocation and execution of the tasks. In addition, since the search and retrieval problem involves two types of tasks, in order to minimise the completion time, each robot needs to consider when and which task to be allocated and executed in this approach.

4.2.1 How to interleave exploring and retrieving?

In the search and retrieval problem, it is clear that initially the robots need to explore. Once they find objects that are needed, they can also start to deliver these objects. In other words, the robots need to consider how to interleave exploration and retrieval tasks. For example, suppose that robot 1 in Fig. 3 (a sub-graph of Fig. 1b) knows that object o_1 matches type τ_1 that needs to be matched next to achieve the goal sequence. It can then directly choose to go to collect object o_1 which will take 5 steps to collect, or it can choose to first explore locations l_9, l_3 or l_5, which are closer to its current position. Of course, the robot cannot be sure that it

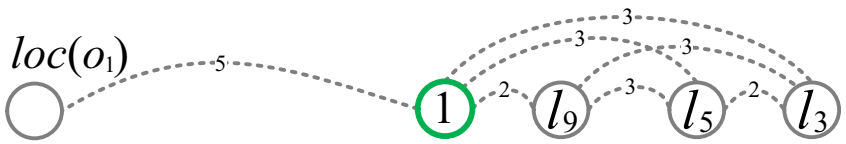


Fig. 3 First retrieve object o_1 or first explore locations close to the robot?

will find another object that matches τ_1 in these locations, but it may still be worthwhile because it may find other objects that it needs to achieve the main goal.

It is not a trivial problem to determine whether retrieval tasks should be performed first, even in a very simple instance where the number of objects n distributed in the environment is equal to the length m of the goal sequence, i.e., $n = m$, and all the objects have the same color. For instance, in Fig. 3, suppose that the main goal consists of two red boxes, and a red box o_1 has been found. Given $n = m$, o_1 must be retrieved from $loc(o_1)$ anyway, and the locations l_9, l_5 or l_3 need to be explored to find the other red box. The problem can be simplified as what is the best executing path. For instance, $l_9 \rightarrow loc(o_1) \rightarrow l_5 \rightarrow l_3$ can be considered as a path, and total $4! = 24$ paths need to be evaluated in this case. The position of $loc(o_1)$ is important in such a path because the robot may first need to bring the collected the box back to the home base, and then start exploring another target location. To calculate the cost of a path is thus not simple, we also need to consider the probability of finding the other red box, which increases the computational complexity.

As mentioned earlier, robots do not have the information, $n = m$, in the search and retrieval problem that we study here, so they cannot use offline planning methods to evaluate and consequently find an optimal path.

Since the robots cannot use offline planning to allocate all tasks upfront and then start executing them, task allocation and execution take place in parallel in the auction approach. This means that once a robot has been allocated a task, it can start executing the task, and while performing one task, it still can bid for another available task.

Allocation In the search and retrieval problem, since the robots do not initially know the locations of objects, they have to begin by bidding on exploration tasks. This means that only exploration tasks are available in the early stage of auctions, and one robot may be allocated multiple exploration tasks. Retrieval tasks appear dynamically when the robots are executing their allocated exploration tasks and target objects are found. As the indexed types in the goal sequence should be retrieved in the right order, we assume that the robots only bid on a discovered object to satisfy an indexed type when other objects have already been located to satisfy the preceding indexed types in the goal sequence.

In order to distribute the workload more evenly in the auction approach, when determining the bidding cost for a new task, each robot bids the *total travel cost* of completing all the previously allocated tasks as well as the new task. This means that when bidding on a retrieval (or exploration) task, each robot should also take account of the costs for completing all the exploration and retrieval tasks allocated to it in previous rounds. Note that it is possible that a robot has both exploration and retrieval tasks to bid on when an object has already been found, but not all the exploration tasks have been allocated. Robots in this case still use the MinMax criterion to determine which task to bid on. According to the MinMax criterion, a robot may still choose an exploration task to bid on even if there is a retrieval task available, implying that the robot would rather choose a nearby location to explore than directly go to retrieve a faraway object.

In the auction-based approach, once a task is allocated to a robot, the robot is committed to achieve it, and we do not consider re-allocating these tasks. Since *only one* task is allocated to the robot who made the lowest bid in *each* round, if there are q target locations and the team goal requires t types of objects, all the tasks can be allocated in $q + t$ rounds of auctions.

Execution Since a robot may have multiple allocated tasks to execute at any moment, we need to further consider the execution of allocated tasks. We give higher priority to the retrieval tasks because they directly contribute to achieving the team goal, and the robots do not have to explore all the locations in order to complete the team goal. Nevertheless, we still need to consider the execution order of each set of allocated tasks. For the retrieval tasks, since all the indexed types must be satisfied in the right order, the order of performing retrieval tasks should match the order of types in the goal sequence. The robots do not need to change the execution order of retrieval tasks because this type of tasks are allocated in accordance with the sequence of the team goal. But for the exploration tasks, when winning an exploration task, a robot may need to consider re-ordering these tasks because its current location might have changed. As each robot only bids on one task in each round, the *cheapest insertion heuristic* can be used to find the optimal position to insert the new winning task into the list of previously allocated ones, as in [16, 26]. In such a way, when a robot has completed one task, it can pick up another allocated task from the list to perform until the team goal is accomplished.

In summary, each robot only performs the tasks that are allocated to it in the auction approach. In order to satisfy the team goal, all allocated retrieval tasks must be completed, but all allocated exploration tasks do not have to accomplished if the team goal can be achieved by already located objects.

4.2.2 Algorithm

We formalize our extended SSI auction approach for the search and retrieval problem in Algorithm 1. It shows how an individual robot (e.g., robot i) performs a search and retrieval task, mainly consisting of bidding, re-ordering and executing procedures.

Algorithm 1 Extended SSI auction approach for the search and retrieval problem.

1: Input: • the goal sequence $\langle \tau_1, \ldots, \tau_m \rangle$ that the robot team has to accomplish.
2: • T_E^i: the exploration tasks allocated to robot i.
3: • T_R^i: the retrieval tasks allocated to robot i.
4: **while** the remaining goal sequence has not been achieved **do**
5: **Update** the currently available unallocated exploration task U_E. ▷ see (2).
6: **Update** the currently available unallocated retrieval tasks U_R. ▷ see (3).
7: **procedure** BIDDING
8: **if** $U_E \cup U_R \neq \emptyset$ **then** ▷ at least an exploration task or retrieval task is available.
9: **for all** $t \in U_E \cup U_R$ **do**
10: Estimate $cost(i, t) = cost(T_E^i \bigcup T_R^i \bigcup \{t\})$ ▷ estimate the total travel cost.
11: **end for**
12: Bid on the task t^* with the smallest cost.
13: **end if**
14: **end procedure**
15: **procedure** RE-ORDERING
16: **if** received winning task t^* **then**
17: Update $T_E^i \leftarrow T_E^i \bigcup \{t^*\}$ or $T_R^i \leftarrow T_R^i \bigcup \{t^*\}$ ▷ use *cheapest insertion heuristic*.
18: **end if**
19: **end procedure**
20: **procedure** EXECUTING
21: **if** $T_R^i \neq \emptyset$ **then**
22: Go to retrieve the first indexed object in T_R^i
23: **else if** $T_E^i \neq \emptyset$ and enough objects have not been located **then**
24: Go to explore the first indexed location in T_E^i
25: **end if**
26: **end procedure**
27: **end while**

In order to decide which task to bid on, a robot first has to estimate the total cost of performing each available task, taking into account the previously allocated but uncompleted ones (line 10). Note that since the retrieval tasks have ordering constraints, the robot will only bid on a discovered object to satisfy an indexed type when other objects have also been located to satisfy the preceding indexed types in the goal sequence. Thus, the ordering constraints must be taken into consideration when calculating the currently available unallocated retrieval tasks U_R in line 6.

With the MINMAX team objective, the robot will choose the task that minimizes the overall cost (line 12).

The re-ordering procedure is used to insert a winning task announced by the auctioneer into the list of allocated but uncompleted tasks (line 15–19). As mentioned above, we only need to re-order the exploration tasks by means of the cheapest insertion heuristic.

In the executing procedure, the robot decides which task to execute and when to stop. In this approach, each robot gives top priority to executing the retrieval tasks as delivering objects can directly contribute the team goal. For an individual robot, it has to complete all its allocated retrieval tasks (line 21), but it does not need to finish all the allocated exploration tasks if there are enough objects that have been located and found to satisfy the team goal (line 23). According to the algorithm, if the robot is not allocated an object that it just found, it will not pick it up for delivering. This case happens when the robot has already been allocated too many tasks to complete, so it cannot offer the smallest bid to win this object.

5 A prediction task allocation approach

In this section, we will discuss a prediction approach to the search and retrieval problem, an implicit coordination framework by means of team reasoning without auctions and negotiations. *Implicit coordination* means that the robots do not explicitly reach an agreement on the allocation of tasks. In this approach, we are interested in fully decentralized robot teams, though it is still called *task allocation*, no coordinator or determiner is used to *confirm the allocation of tasks* throughout this proposed approach.

With regard to task allocation, the major differences between the extended auction-based approach and this prediction approach are described as follows:

– In the auction approach, the robots can directly access the information about allocated tasks by means of, for example, the auctioneer; in the prediction approach, the robots need a method for agreeing on the allocation of tasks.
– In the auction approach, a robot may have multiple allocated tasks to perform; in the prediction approach, a robot only has at most one allocated task to perform.

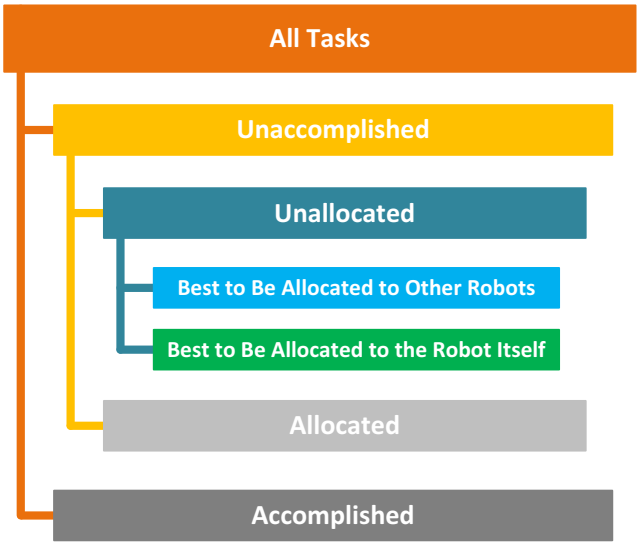


Fig. 4 The understanding of the progress of teamwork

- In the auction approach, a robot only performs a task that is allocated to it; in the prediction approach, a robot chooses a task to perform when it has noting to do.
- In the auction approach, once a task is allocated to one robot, it will not be re-allocated to other robots; In the prediction approach, task allocations can be reconsidered (using the prediction mechanism again).

5.1 Implicit coordination framework

Figure 4 shows the basic idea of our prediction approach as to how robots in a team understand the progress of their teamwork. All the tasks can be divided into either accomplished or unaccomplished ones, some of which have been allocated, while some of which have not yet been allocated. In order to achieve good team performance, the idea is to identify the robot that is most cost-effective to accomplish each unallocated task.

5.1.1 Allocation criterion

To this end, the robots first need to figure out which tasks can be treated as *allocated* ones, and they need a method for agreeing on the allocation of tasks. In this implicit coordination framework, a robot will consider a task as allocated *only if it cannot do better than its teammates*. For an individual robot, an allocated task means that it has not been accomplished yet, but one of its teammates has planned to achieve it, and, most importantly, the robot itself cannot achieve it at a lower cost than a teammate.

In order to achieve good team performance, each robot, moreover, should realise that some of the unallocated tasks are better to be performed by its teammates, while some of them are better to be achieved by the robot itself.

5.1.2 Commitment strategy

In order for most cost-effective robots to achieve the tasks in the search and retrieval problem, we introduce a *commitment strategy*. This strategy is employed to define when a robot is committed to perform a task and when it should drop its commitment. For the former, each robot needs to make a plan of achieving a task when it *has nothing to do*. For the later, each robot should have the *willingness to give up a planned task if its teammate can do better than itself*. More specifically, if a robot realizes that a teammate has the same plan of achieving a task, and that it cannot do better then the teammate, the robot will drop its original plan. Likewise, each robot should believe that its teammates also have the willingness to give up planned tasks if it can do better than them in the same circumstances.

The commitment strategy is thus used to deal with *conflicting plans* without extra communication or negotiation. With regard to the cooperative character as a team, the robots with conflicting plans for achieving a task can finally commit the most cost-effective robot to accomplish it.

5.2 Communication for team awareness and intentions

The notion of team awareness is understood here as the state of a robot's beliefs about itself, about other robots and about the environment, while team intentions refer to what the robots are planning to do. Team awareness and intentions can be built on various forms of observation, communication and reasoning. Keeping consistent information in decentralized teams is still a challenging issue [34]. Although the robots can directly communicate with each other, there exists a time lag between the time a robot sends a message and the time the other robots receive it in decentralized systems, even if the time lag is very short. For example, when robot 1 decides to explore location A, it may not know that its teammate, robot 2, is also planning to explore the same location at the moment. Even if robot 2 can send a message to inform robot 1 of its plan, it still might be too late for robot 1 to receive the message so as to avoid making a conflicting plan. In the prediction approach, since the commitment strategy can handle conflicting plans, we do not need to worry about communication delay in usual decentralized systems .

With regard to communication content, as in our previous work [29], the robots can share their beliefs, i.e., what they

have observed in the environment and where they are, and their intentions, i.e., what they are planning to do. Specifically, each robot will exchange the following messages in the prediction approach:

1. its current location,
2. an object (or objects) found by itself when checking a target location,
3. an object collected by itself,
4. a target location that it wants to explore, and
5. an object that it wants to collect.

In comparison with the extended auction-based approach discussed in Section 4, here the robots do not need to communicate their estimated costs for accomplishing specific tasks. This is because each robot can calculate the cost information of its teammates if it knows the current locations of them.

As working in a team, the robots will inform each other of found or removed objects. The robots need the fourth and fifth types of information to predict the unallocated tasks, which will be discussed in the following subsection.

5.3 Predicting unallocated tasks

Though the robots will inform each other about their plans, (i.e., the fourth and fifth types of messages discussed above), they still cannot simply confirm which tasks are sure to be allocated to which robots, and what still needs to be done in order to complete the team goal. This is because when a robot has made a plan to accomplish a task, it does not mean that all the others will agree that the robot is delegated to finish the task. According to the allocation criterion, if a teammate can achieve the task at a lower cost, the robot should allow the teammate to complete the task. Thus, in the prediction approach the robots do not directly have access to the information about the set of unallocated task by means of (2) and (3). They can, however, use the exchanged messages to infer this information as follows.

5.3.1 Predicting unallocated exploration tasks

It is not difficult for the robots to keep track of explored locations because they broadcast their real-time locations. Once a robot arrives at a target location, all the others can conclude it as an explored one. However, all the unexplored locations usually cannot be regarded as unallocated ones because some of them might have been planned to be visited. Although each robot can know the plans of teammates, but when a robot has decided to explore a location, it does not mean that this robot is the most cost-effective one to explore it. It is reasonable to accept that if there is a teammate which also has decided to explore the location with less cost, it is better to let the teammate explore the location. From another point of view, if a robot realises that it can explore a target location at a lower cost than a teammate that has decided to explore the location, the robot still can make a plan to explore the location.

In this approach, each robot can use (6) to predict which locations still can be planned to explore:

$$U_E = E \backslash L_{exploring}, \tag{6}$$

where E refers to the set of locations that has not been explored, $L_{exploring}$ is introduced here to indicate the set of locations that its teammates have planned to explore, and that the robot itself cannot explore faster than those teammates. In order to obtain $L_{exploring}$, a robot first needs to select the set of the target locations that its teammates want to explore, using the fourth type of information. Then, it needs to (use (4) to) compute whether it can explore each of those locations faster than respective teammates. The information of current locations of the teammates is needed to make this calculation. If the robot cannot explore a target location of the set more quickly, then the target location will be put into $L_{exploring}$.

5.3.2 Predicting unallocated retrieval tasks

Although the retrieval tasks appear dynamically when a discovered object can be retrieved to satisfy an indexed type of the team goal sequence, a robot cannot treat all the newly appearing retrieval tasks as unallocated ones. The set of unallocated retrieval tasks is often a subset of the overall available retrieval tasks, $U_R \subseteq R$. Robots need to know U_R because they need to understand what still needs to be done in order to complete the entire team goal. A robot can be informed of what objects have been retrieved by its teammates and what objects its teammates have planned to retrieve. But it does not mean that the robot would accept that the objects that its teammates have planned to retrieve are the best choices to satisfy the corresponding types in the goal sequence.

For better teamwork, according to (5), if a robot cannot do better to retrieve an object than a teammate that has planned to retrieve the same object, it should acknowledge that the object has been allocated to the teammate.

For example, if there are two discovered objects o_1 and o_2, and both of them can be retrieved to satisfy τ_3, then we have that $R = \{\langle o_1, 3\rangle, \langle o_2, 3\rangle\}$. Let robot 1 and 2 work as a team to fulfill τ_3. If robot 1 has made a plan to retrieve o_1, what should robot 2 do? Can it consider that the retrieval tasks $\langle o_1, 3\rangle$ and $\langle o_2, 3\rangle$ are still available?

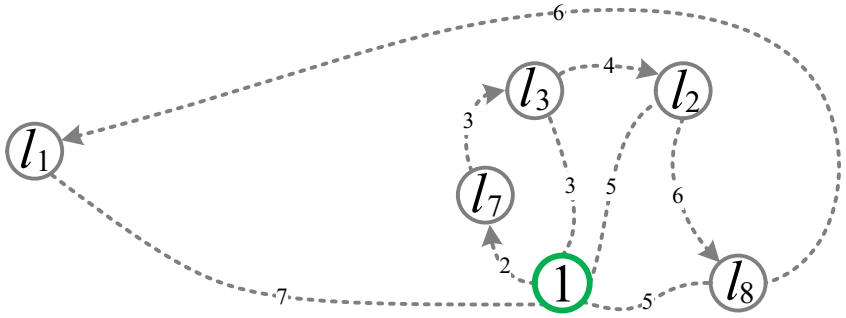


Fig. 5 Greedy & non-greedy strategies for exploration

In order for the most cost-effective robot to achieve each retrieval task in this approach, the idea is that if robot 2 can beat robot 1 to retrieve any object to satisfy the required type, then robot 2 would believe that $U_R = R = \{\langle o_1, 3\rangle, \langle o_2, 3\rangle\}$; otherwise, robot 2 will acknowledge that the required type has been allocated to robot 1 to fulfill, and, hence, it believes that no retrieval task is available (i.e., $U_R = \emptyset$) right now.

In this approach, each robot can use (7) to predict the currently available unallocated retrieval tasks:

$$U_R = R \backslash R_{retrieving}, \tag{7}$$

where R refers to the set of retrieval tasks that has not been retrieved, $R_{retrieving}$ is introduced here to indicate the set of retrieval tasks that its teammates have planned to retrieve, and that the robot itself cannot beat the teammates at retrieving. In order to obtain $R_{retrieving}$, a robot first needs to select the set of objects that its teammates want to collect, using the fifth type of information. Then, it needs to (use (5) to) computer whether it can collect each of those objects faster than respective teammates. Similarly, the information of current locations of the teammates is needed to make this calculation. If the robot cannot go to collect an object of the set more quickly, then the object will be put into $R_{retrieving}$. For the final set of U_R, the robot will eliminate each element of $R_{retrieving}$ from R sequentially.

5.4 Plan generation

Since the robots can predict the set of unallocated tasks, they can choose tasks to accomplish when they have nothing to do. As in the extended auction-based approach, robots prefer to execute retrieval tasks first because this type of task directly contributes to the team goal. In the prediction approach, a robot will also first consider choosing an unallocated retrieval task to perform. If no retrieval task is available, the robot will explore the environment by choosing an unallocated exploration task to visit. In this section, we will first discuss how to select a target location to explore and then how to choose an object to collect.

5.4.1 Non-greedy strategy for selecting exploration tasks

Auction-based approaches often suffer from greedy solutions [16], making use of *hill-climbing principle to bid for the task with the smallest cost*. A robot cannot win the tasks with larger costs because the winner determination mechanism allocates a task to the robot who made the lowest bid in each auction round [16]. Greedy exploration means that the robots often put more effort into local minima if needed objects are actually distributed in faraway locations. For example, as shown in Fig. 5 (a sub-graph of Fig. 1b), robot 1 needs to search and retrieve an object to satisfy τ_1, and such an object can be found in location l_1. If the robot uses the greedy strategy, it will follow the exploration path: $l_7 \rightarrow l_3 \rightarrow l_2 \rightarrow l_8 \rightarrow l_1$, with the total cost of 21 to find the object. Actually, if the robot is able to directly choose l_1, it only takes 7 steps to find the object.

The robots in the proposed prediction approach can also apply a greedy strategy to choose the nearest one from the set of the unallocated exploration tasks to explore. Moreover, it can produce a non-greedy strategy, in which each robot has higher likelihood to choose a nearby location than a faraway one, but it at least has a chance to choose a faraway location to explore.

In the prediction approach, we employ the fitness function used in genetic algorithms to calculate the likelihood for a robot to choose an unallocated exploration task. Table 1 is an example that shows the detailed steps of how a robot calculates the likelihood for choosing the exploration tasks depicted in Fig. 5:

Table 1 Calculate the likelihood for choosing exploration tasks depicted in Fig. 5

Exploration tasks: E	l_7	l_3	l_2	l_8	l_1
cost to each location: $cost_E(1, l_i)$	2	3	5	5	7
reverse cost (fitness): $f(x_i) = \frac{1}{cost_E(1,l_i)}$	0.5	0.3333	0.2	0.2	0.1429
likelihood (fitness ratio): $p_i = \frac{f(x_i)}{\sum_i f(x_i)}$	36.33 %	24.22 %	14.53 %	14.53 %	10.38 %

1. enumerate all the available exploration tasks;
2. for each exploration task $l_i \in E$, calculate its costs: $cost_E(1, l_i)$ using (4);
3. reverse the cost and get the fitness of each exploration task: $f(x_i) = \frac{1}{cost_E(1,l_i)}$;
4. calculate the likelihood of each exploration task: $p_i = \frac{f(x_i)}{\sum_i f(x_i)}$.

The probability of selecting an exploration task depends on the distance to reach it. Using the non-greedy strategy, we can see that a robot has greater likelihood to choose a nearby location, but still has a possibility to explore a faraway one.

If several robots plan to explore the same whatever nearby or faraway location, according to the commitment strategy, the location will be finally explored by the robot who has the lowest cost.

5.4.2 Individual optimal strategy for selecting retrieval tasks

Since the team goal may have ordering constraints that require targets to be delivered to the home base in a specific order, the unallocated retrieval tasks should be completed according to the indexed types of the goal sequence. In this approach, a robot will choose an unallocated retrieval task to perform, which satisfies the next indexed type. It will also inform its teammates of its plan. In a search and retrieval environment, the robot may have multiple choices to do so, i.e., there might be multiple located objects that can be retrieved to satisfy the same indexed type. To fulfill an indexed type at a lower cost, each robot will choose the nearest object to retrieve.

It happens that several robots may plan to retrieve the same object (or another object with the same type) at the same time. From the perspective of an individual robot, if a robot realizes that it has the exact same plan with a teammate and that it cannot beat the teammate, it will give up its original plan, drop the commitment and also inform the team. At the same time, the robot will treat the object as an allocated one and update the information about the set of the unallocated retrieval tasks, using (7).

If a robot realizes that a teammate wants to retrieve a different object with the same type, and that robot cannot beat the teammate to satisfy the same type, the robot may not have to drop its commitment. This is because its intended object may be useful for satisfying the next indexed type in the goal sequence. To further check whether it should drop the commitment or not, the robot needs to update the set of the unallocated retrieval tasks, using (7).

On the contrary, if a robot realizes that it beats a teammate, it can continue with its original plan. The robot, moveover, believes that the teammate will drop the commitment. In such a way, each retrieval task will be finally allocated and completed by the most cost-effective robot.

5.5 Algorithm of the prediction approach

We use Algorithm 2 to describe the proposed prediction approach in detail. In comparison with the extended auction-based approach, the important feature is that each robot only has one allocated task at any moment. In general, the algorithm consists of plan generation (line 5–16) and the commitment strategy (line 17–24). The robot needs to predict the set of unallocated tasks first when determining which task to choose (line 6 and 7). And because the retrieval tasks can directly contribute to the team goal, we prioritise the retrieval tasks over the exploration tasks, which is the same as in the auction approach. It means that in order to fulfill an indexed type in the goal sequence, a robot will first retrieve an found object if it knows such information (line 8–12); otherwise, it will choose a target location to explore (13–15). It should be noted that since the team goal may have ordering constraints, for the retrieval tasks, a robot needs to first consider achieving the type with the smallest index in the remaining goal sequence (line 9). Choosing the exploration tasks, robots can apply the non-greedy strategy, see Section 5.4.1, to make decisions (line 14).

Algorithm 2 Prediction approach for search and retrieval task allocation.

1: Input: • the goal sequence $\langle \tau_1, \ldots, \tau_m \rangle$ that the robot team has to accomplish.
2: • E: the set of all currently available uncompleted exploration tasks.
3: • R: the set of all currently available uncompleted retrieval tasks.
4: **while** the remaining goal sequence has not been achieved **do**
5: **if** robot r_i has no plan of achieving any task **then**
6: **Predict** the currently available unallocated exploration tasks U_E ▷ see (6).
7: **Predict** the currently available unallocated retrieval tasks U_R ▷ see (7).
8: **if** $U_R \neq \emptyset$ **then**
9: **for all** retrieval tasks $r = \langle o, j \rangle$ such that $type(o) = \tau_j$ **do** ▷ j is the smallest index in U_R.
10: Estimate $cost(i, r)$ ▷ see (5).
11: **end for**
12: Make a plan to retrieve the nearest object and inform the others
13: **else if** $U_E \neq \emptyset$ **then**
14: Choose the location to explore ▷ non-greedy exploration, see Section 5.4.1.
15: Make a plan to explore the location and inform the others
16: **end if**
17: **else if** robot r_i has a plan of achieving an exploration task or a retrieval task **then**
18: **if** a teammate has the same plan and can be faster than itself **then**
19: drop the commitment of plan
20: **else if** has the plan of retrieving an object, while a teammate has the plan of retrieving another object of the same color, and can be faster than itself **then**
21: Check whether the object still can be used to satisfy the next indexed type
22: **end if**
23: Execute the current plan to complete the task
24: **end if**
25: **end while**

If a robot has conflicting plans with its a teammate which can do better than itself, then the robot will drop its commitment (line 18). Without conflicting plans, in the case of having a plan of retrieving an object, a robot still needs to further check whether a teammate has a plan of retrieving another object with the same type, and if so, whether it can be faster than itself (line 20). Such a case occurs when two robots want to fulfill a same indexed type by retrieving two objects at the same time. Even if a robot may not do it faster than the teammate, its current plan still can satisfy the next indexed type. Hence, it needs to inspect such a possibility before thoroughly giving up the plan (line 21). Otherwise, the robot can execute its plan to complete the task (line 23).

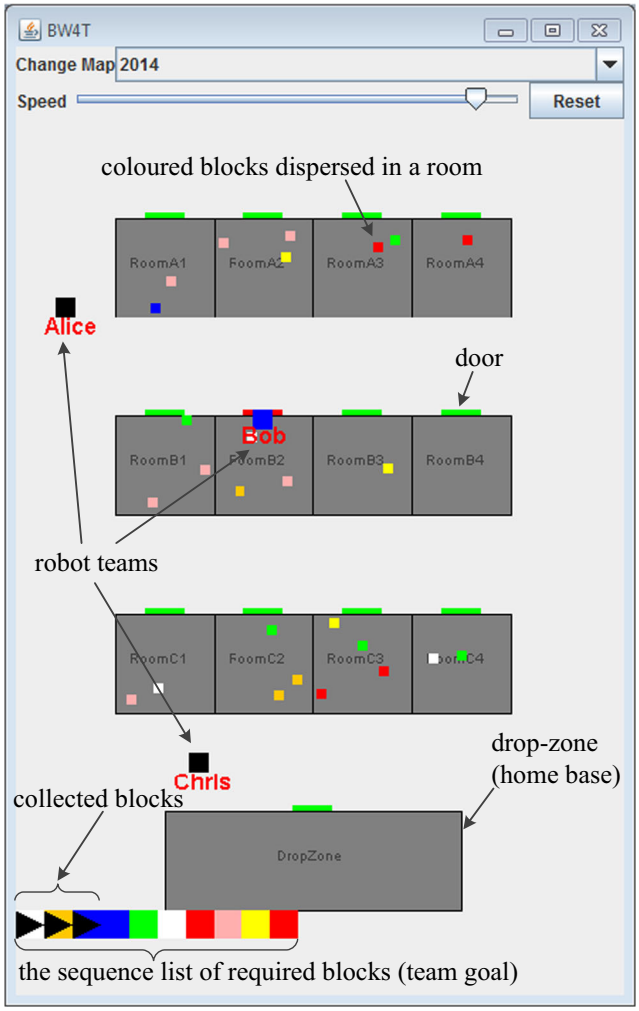


Fig. 6 The Blocks World for Teams simulator for multi-robot search and retrieval

6 Experiments

6.1 Simulator: the blocks world for teams

For the sake of repeatability and accessibility, we use a simulator, called the Blocks World for Teams (BW4T)[3], to study a search and rescue task in this work. The BW4T has office-like environments consisting of *rooms* in which colored *blocks* are randomly distributed for each simulation (see Fig. 6). The colored blocks represent survivors that

[3]The BW4T introduced in [14] has been integrated into the agent environments in GOAL [12], and more information of it can be found from https://github.com/eishub/BW4T.

can be humans, various animals, etc., in a village after an earthquake. We use different colors to distinguish survivors so that they can be taken care of with different priorities (e.g., it is reasonable to give a high priority to humans). The *rooms* are considered as the rough areas where the survivors might be located. We assumed that in the simulator the robots have the information about the locations of the rooms in advance, but they do not know what kinds of block is in a room before checking it. In real conditions, even though the robots may have the map information of the village, they are not likely to know the precise locations of the survivors before searching them. Robots are supposed to search, locate, and collect found blocks from the rooms and return them back to a so-called *drop-zone*. The *drop-zone* is treated as a hospital, where the survivors need to be transported to. Moreover, due to the limited carrying capability of real robots, we also assume that a robot can only transport one survivor at one time. Correspondingly, in the simulator the robots are able to carry at most one block at a time in this work.

As indicated at the bottom of the simulator in Fig. 6, the team goal of a search and rescue mission is indicated by a sequence of required blocks. The required blocks need to be delivered to the drop-zone in the specific order as displayed. Access to the rooms is limited in the BW4T. At any time at most one robot can be present in a room or the drop-zone, and the robots have to go through a door to enter a room. For example, we can imagine that the treatment to survivors in the hospital should be well-organised according to their priorities. When a robot arrives at a target location, it usually needs to employ its manipulators to rescue a survivor, which requires that each robot must have a safety region for using its actuators.

To complete a search and retrieval mission, each robot is informed of the team goal, i.e., the sequence of the required blocks, at the start of a simulation. The robots have the information about the locations of the rooms, but they do not initially know which blocks are located in which rooms. This knowledge is obtained for a particular room by a robot when it visits that room. While interacting with the BW4T environment, each robot gets various percepts that allow it to keep track of the current environment state. Each robot has its own localization function, which allows it to update its current location. Due to the limited perception, a robot in the BW4T can only perceive a colored block when entering a room where the block is located. Blocks are identified by a unique ID and a robot in a room can perceive which blocks of what color are in that room.

In order to implement and evaluate the task allocation solutions discussed in this paper, we believe that cognitive agents are particularly suitable for controlling the robots in the BW4T, and the GOAL [12] language allows programming agents that reason, communicate, and can interact with the BW4T environment.

6.2 Experimental design

In the experimental study, we evaluate the extended SSI auction approach and the novel prediction approach. In our study, we have taken the following factors into account: the environmental size, the team size, and the deployment (i.e., the initial starting location) of the robots.

(a) the small map (b) the large map

Fig. 7 The maps and the deployment of robots

Table 2 The experimental setups for the search and retrieval problem

Approach	Map	Deployment	Robots
[auction, novel]	[small, large]	[close, dispersal]	[1, 5, 10]

We design two maps with a more general layout of rooms and doors (see Fig. 7), a small one with 12 rooms (see Fig. 7a) and a large one with 30 rooms (see Fig. 7b). Both approaches are examined by a single, five and ten robots. For the deployment of robots, we test two proposals: the robots will start from more or less the same location (i.e., the shaded areas in Fig. 7a, b), and they are initially distributed over the environment and start in different locations as shown in Fig. 7a, b.

Table 2 shows the experimental setups. For a single robot, it starts from the shaded areas in Fig. 7 for both the small and large map.

The team goal of the robots is to collect 15 colored blocks from the environment, and the set of required blocks is set randomly in each simulation. The environment of the BW4T will randomly generate 30 blocks in total in each simulation. Initially, the robots have no knowledge about this distribution. In our experiments, we measure the *completion time* and the *steps* indicating the level of consumed energy. Each condition has been run for 50 times to reduce variance and filter out random effects in our experiments. The experiments run on an Intel i7-3720QM at 2.6 GHz with 8 GB of RAM, using GOAL version 8024.[4]

6.3 Results

Figure 8 shows the results of the experimental study, in which we have evaluated the extended auction-based approach for the search and retrieval problem, labelled as "auction", and the novel prediction approach, labelled as "prediction".

6.3.1 Completion time

In order to compare the two approaches, we use the single robot case as a baseline because no workload needs to be shared. We would expect no difference between the two approaches, but with respect to the completion time, the running time of agent programs can have an influence. As shown in Fig. 8a, the auction approach runs faster in both the small and large maps than the prediction approach. For example, a single robot that applies the auction approach takes 89.54 seconds on average to accomplish all the required blocks in the small map, and 162.50 seconds in the large map. In contrast, if the robot applies the prediction approach, it takes 101.27 and 173.23 seconds, respectively.

When more robots engage in the teamwork, the prediction approach performs better than the auction approach. In particular, the prediction approach significantly reduces completion time when the robots are working in a large map. As shown in Fig. 8a, when the team has five robots, the completion time of the auction approach is almost two times as much as the prediction approach needs. This trend becomes more clear with increasing numbers of robots. For instance, when ten robots start from more or less the same location (i.e., close depots) to execute the search and retrieval problem in the large map, they only need 50.93 seconds to finish the task when they use the prediction approach, compared to 150.57 seconds for the auction approach.

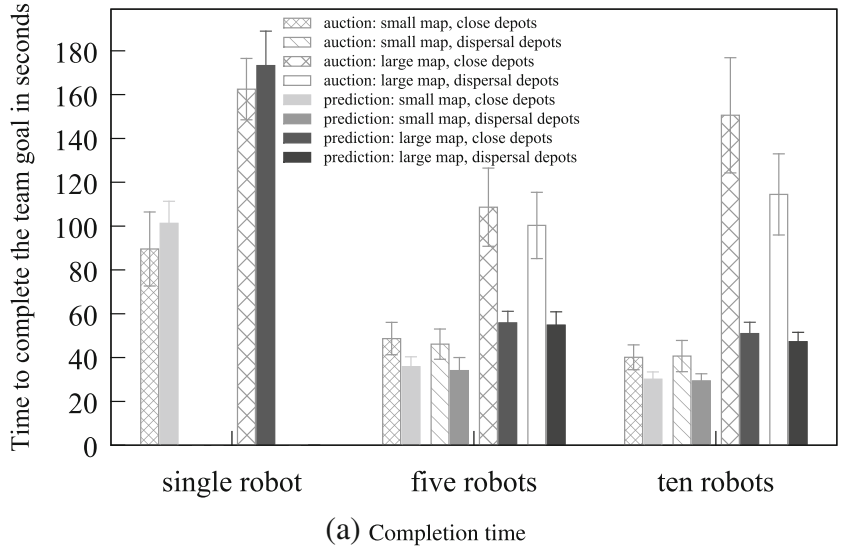
We believe there are two reasons that can explain why the prediction approach performs better with increasing number of robots. First, the auctioneer needs to wait for all the robots to submit their bids in a round before determining a winner, which increases the time for allocating tasks. Therefore, the time needed to complete an auction round increases with the number of robots and introduces real time delay.

Second, we have noticed that in the auction approach when a robot explores a room and finds the next block that is needed, the associated retrieval task for this block may be allocated to another robot because the robot already has a heavy workload. It happens, for example, when a robot has already been allocated many rooms to explore, which causes its bid (based on total estimated costs) to be rather high for a newly found block. This is because the auction approach allows a robot to bid on a new task when executing one, and each robot may have multiple allocated tasks when bidding on a new task. In contrast, a robot that applies the prediction approach only determines which task to perform when it has nothing to do. Thus, once a robot enters a room and finds the next needed block, the robot itself must be the most cost-effective one to collect the block.

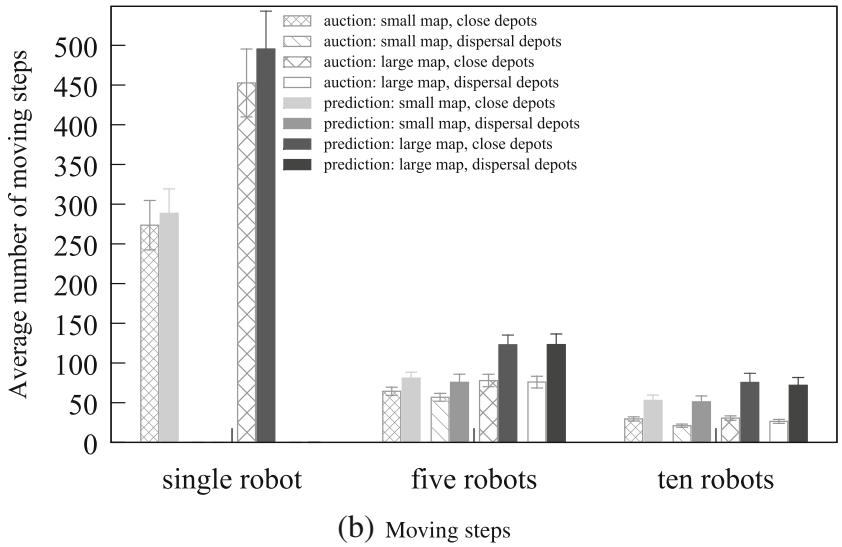
With regard to the influence of the initial deployment of the robots, we can see in Fig. 8a that it also relates to the map size and team size. In a large environment, the robots starting from the different locations save the completion time. For instance, in the auction approach, if five robots start from the same location (i.e., close depots) in the large map, they need 108.61 seconds to finish the task, compared to 100.29 seconds if they start from different location (i.e., dispersal depots). If we put ten robots in the small map, their starting locations do not significantly differ from each other. We can conclude that the initial

[4] GOAL version 8024 can be downloaded from http://ii.tudelft.nl/trac/goal/wiki/Releases.

Fig. 8 Experimental results



(a) Completion time



(b) Moving steps

deployment of the robots is important for large-scale search and retrieval.

6.3.2 Steps

From Fig. 8b, we conclude that the robots need fewer steps if they use the auction approach to perform the search and retrieval task. The main reason is that using the auction approach, a robot does not move unless a task is allocated to itself, whereas using the prediction approach, a robot first makes quick decisionsto perform a task, and then checks its commitment while executing the task. Thus, the robot may drop an original plan if it finds that it cannot do better than a teammate for a specific task, which can result in more steps than the auction approach.

Therefore,deciding which approach should be applied to deal with the dynamic search and retrieval problem involves making a decision on the trade-off to minimize the completion time versus minimizing the total fuel consumption. In different application domains, the search and retrieval robots may focus on different team objectives. For instance, in the search and rescue domain, the completion time may be the major team objective because the survivors need urgent assistance. However, in the deep-sea mining domain, the fuel consumption may be the major concern for huge mining robots.

7 Conclusions

In this work, we investigated the *dynamic task allocation for a more general search and retrieval problem* and presented an auction-based approach and a novel prediction approach to deal with the problem. The two approaches are evaluated in a simulated environment, called the BW4T, and the experimental results show that both of them provide an efficient solution to the problem. The prediction approach performs better with respect to completion time, while the auction-based approach performs better with respect to moving steps. Thus, there is a trade-off between minimizing the completion time and minimizing the fuel consumption for different application domains. In the future, we would like to use real robots to evaluate the proposed approaches.

Acknowledgments This research was partially supported by the Fundamental Research Funds for the Central Universities (2015B30114). We would like to thank the anonymous reviewers for providing us with constructive comments and suggestions.

References

1. Bektas T (2006) The multiple traveling salesman problem: an overview of formulations and solution procedures. Omega 34:209–219
2. Bernardine Dias M, Zinck M, Zlot R, Stentz A (2004) Robust multirobot coordination in dynamic environments. In: Proceedings of the 2004 IEEE international conference on robotics and automation (ICRA), vol 4. IEEE, pp 3435–3442
3. Burgard W, Moors M, Stachniss C, Schneider FE (2005) Coordinated multi-robot exploration. IEEE Trans Robot 21(3):376–386
4. Campo A, Dorigo M (2007) Efficient multi-foraging in swarm robotics. In: Advances in artificial life. Springer, pp 696–705
5. Cao YU, Fukunaga AS, Kahng AB (1997) Cooperative mobile robotics: antecedents and directions. Auton Robot 4:1–23
6. Chen J, Sun D (2011) Resource constrained multirobot task allocation based on leader–follower coalition methodology. Int J Robot Res 30(12):1423–1434
7. Dasgupta P (2011) Multi-robot task allocation for performing cooperative foraging tasks in an initially unknown environment. In: Innovations in defence support systems-2. Springer, pp 5–20
8. Davids A (2002) Urban search and rescue robots: from tragedy to technology. Intelligent Systems. IEEE 17(2):81–83
9. Dias MB, Zlot R, Kalra N, Stentz A (2006) Market-based multirobot coordination: a survey and analysis. Proc IEEE 94(7):1257–1270
10. Farinelli A, Iocchi L, Nardi D (2004) Multirobot systems: a classification focused on coordination. IEEE Trans Syst Man Cybern 34(5):2015–2028
11. Han Y, Li D, Chen J, Yang X, Hu Y (2009) Task allocation algorithm based on robot ability and relevance with group collaboration in a robot team. In: Second international conference on intelligent networks and intelligent systems. IEEE, pp 273–277
12. Hindriks K (2013) The goal agent programming language. http://ii.tudelft.nl/trac/goal
13. Hompel MT, Schmidt T (2006) Warehouse management: automation and organisation of warehouse and order picking systems (intralogistik). Springer, New York
14. Johnson M, Jonker C, Riemsdijk B, Feltovich P, Bradshaw J (2009) Joint activity testbed: blocks world for teams (bw4t). In: Engineering societies in the agents world x, LNCS, vol 5881. Springer, pp 254–256
15. Kaminka GA (2012) Autonomous agents research in robotics: a report from the trenches. In: 2012 AAAI Spring symposium series
16. Koenig S, Keskinocak P, Tovey CA (2010) Progress on agent coordination with cooperative auctions. In: AAAI
17. Koenig S, Tovey C, Lagoudakis M, Markakis V, Kempe D, Keskinocak P, Kleywegt A, Meyerson A, Jain S (2006) The power of sequential single-item auctions for agent coordination. In: Proceedings of the national conference on artificial intelligence, vol 21, p 1625. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999
18. Koenig S, Zheng X, Tovey C, Borie R, Kilby P, Markakis V, Keskinocak P (2008) Agent coordination with regret clearing. In: Proceedings of the 23rd national conference on artificial intelligence, vol 1, AAAI'08, pp 101–107. AAAI Press
19. Krannich S, Maehle E (2009) Analysis of spatially limited local communication for multi-robot foraging. In: Progress in robotics, pp 322–331. Springer
20. Labella TH, Dorigo M, Deneubourg JL (2007) Self-organised task allocation in a group of robots. In: Distributed autonomous robotic systems 6, pp 389–398. Springer
21. Lagoudakis MG, Markakis E, Kempe D, Keskinocak P, Kleywegt A, Koenig S, Tovey C, Meyerson A, Jain S (2005) Auction-based multi-robot routing. In: Proceedings of robotics: Science and systems, pp 343–350
22. Liu W, Winfield AF, Sa J, Chen J, Dou L (2007) Towards energy optimization: emergent task allocation in a swarm of foraging robots. Adapt Behav 15(3):289–305
23. Nanjanath M, Gini M (2008) Performance evaluation of repeated auctions for robust task execution. In: Simulation, modeling, and programming for autonomous robots, pp 317–327. Springer
24. Parker LE (1998) Alliance: An architecture for fault tolerant multirobot cooperation. IEEE Trans Robot Autom 14(2):220–240
25. Parker LE (2008) Distributed intelligence: overview of the field and its application in multi-robot systems. J Phys Agents 2(1):5–14
26. Schoenig A, Pagnucco M (2011) Evaluating sequential single-item auctions for dynamic task allocation. In: AI 2010: advances in artificial intelligence, pp 506–515. Springer
27. Tovey C, Lagoudakis MG, Jain S, Koenig S (2005) The generation of bidding rules for auction-based robot coordination. In: Multi-robot systems. From swarms to intelligent automata, vol III, pp 3–14. Springer
28. Tsalatsanis A, Yalcin A, Valavanis KP (2009) Optimized task allocation in cooperative robot teams. In: 17Th mediterranean conference on control and automation, pp 270–275. IEEE
29. Wei C, Hindriks K, Jonker CM (2014) The role of communication in coordination protocols for cooperative robot teams. In: Proceedings of international conference on agents and artifical intelligence (ICAART), pp 28–39
30. Winfield A (2009) Foraging robots. In: Meyers R. A. (ed) Encyclopedia of complexity and systems science. Springer, New York, pp 3682–3700
31. Yuh J (2000) Design and control of autonomous underwater robots: a survey. Auton Robot 8(1):7–24

32. Zheng X, Koenig S (2009) K-swaps: cooperative negotiation for solving task-allocation problems. In: Proceedings of the 21st international jont conference on artifical intelligence, pp 373–378. Morgan Kaufmann Publishers Inc
33. Zlot R, Stentz A (2005) Complex task allocation for multiple robots. In: Proceedings of the 2005 IEEE international conference on robotics and automation (ICRA), pp 1515–1522. IEEE
34. Zlot R, Stentz A (2006) Market-based multirobot coordination for complex tasks. Int J Robot Res 25(1): 73–101
35. Zlot R, Stentz AT, Dias MB, Thayer S (2002) Multi-robot exploration controlled by a market economy. In: IEEE International conference on robotics and automation, vol 3, pp 3016–3023. IEEE