Auction-Based Dynamic Task Allocation for Foraging with a Cooperative Robot Team

Changyun Wei
Hohai University
13 PUBLICATIONS **51** CITATIONS

Koen V. Hindriks
Vrije Universiteit Amsterdam
229 PUBLICATIONS **3,054** CITATIONS

Catholijn M. Jonker
Delft University of Technology
542 PUBLICATIONS **6,325** CITATIONS

Auction-Based Dynamic Task Allocation for Foraging with a Cooperative Robot Team

Changyun Wei(✉), Koen V. Hindriks, and Catholijn M. Jonker

Interactive Intelligence Group, EEMCS, Delft University of Technology, Mekelweg 4, 2628 CD, Delft, The Netherlands
{c.wei,k.v.hindriks,c.m.jonker}@tudelft.nl

Abstract. Many application domains require search and retrieval, which is also known in the robotic domain as foraging. An example domain is search and rescue where a disaster area needs to be explored and transportation of survivors to a safe area needs to be arranged. Performing these tasks by more than one robot increases performance if tasks are allocated and executed efficiently. In this paper, we study the Multi-Robot Task Allocation (MRTA) problem in the foraging domain. We assume that a team of robots is cooperatively searching for targets of interest in an environment which need to be retrieved and brought back to a home base. We look at a more general foraging problem than is typically studied where coordination also requires to take temporal constraints into account. As usual, robots have no prior knowledge about the location of targets, but in addition need to deliver targets to the home base in a specific order. This significantly increases the complexity of a foraging problem. We use a graph-based model to analyse the problem and the dynamics of allocating exploration and retrieval tasks. Our main contribution is an extension of auction-based approaches to deal with dynamic foraging task allocation where not all tasks are initially known. We use the Blocks World for Teams (BW4T) simulator to evaluate the proposed approach.

Keywords: Multi-Robot task allocation · Foraging · Auctions

1 Introduction

Robot teams are expected to perform more complicated tasks consisting of multiple subtasks that need to be completed concurrently or in sequence [1]. In this paper, we investigate the Multi-Robot Task Allocation (MRTA) problem in the foraging domain. Foraging is a canonical task for studying multi-robot teamwork [2–5] in which a team of robots needs to search targets of interest in an environment and bring them back to a home base. Many applications need to perform this type of task such as urban search and rescue robots [6], deep-sea mineral mining robots [7] and order picking robots in warehouses [8].

Many bio-inspired, swarm-based approaches to foraging have been proposed in the literature [3,4], where, typically, robots minimally interact with one another

N. Bulling (Ed.): EUMAS 2014, LNAI 8953, pp. 159–174, 2015.
DOI: 10.1007/978-3-319-17130-2_11

as in [3], and, if they communicate explicitly, only basic information such as the locations of targets or their own location are exchanged [9]. Most of this work has focussed on foraging tasks where targets are not distinguished, which reduces the need for explicit cooperation and coordination. In contrast, we study a more general foraging problem where various types of targets are distinguished. Moreover, we also assume *ordering* constraints can be present that require targets to be delivered to the home base in a specific order. Ordering constraints on the types of targets that need to be retrieved are useful for modelling, for example, how urgently a victim needs assistance, how valuable a mining resource is, or how urgently a package is needed.

Task allocation has been extensively addressed in various multi-agent/robot domains over the past few years, with the aim of finding an allocation of tasks to robots that minimises the overall team cost. In general, however, even if the locations of the targets are initially known and only an optimal solution for a multi-robot routing problem [10–12] needs to be found, the problem is NP-hard [10,11,13,14]. The foraging task allocation problem that we study here, moreover, is also harder than the typical multi-robot exploration problem [14–16], where robots only need to search and locate the targets but do not need to deliver them back.

The main contribution of this paper is an auction-inspired approach for dynamic task allocation for foraging. The approach that we propose extends the Sequential-Single-Item (SSI) auctions which have been used to address the routing problem in [10,11]. Comparing with other auctions, [10,11] have shown that SSI auctions can provide a good compromise between computational complexity and solution quality for problems where the set of tasks is initially known. We build on these results but extend the approach to also handle dynamically arriving tasks that need to satisfy additional ordering constraints. In addition, an experimental study is performed to evaluate two heuristics, that is, whether or not robots should stop exploring when needed targets have been located.

This paper is organized as follows. We begin by analysing the MRTA problem for foraging and presenting a formal model of the problem we deal with in Sect. 2. The auction-inspired approach is discussed in Sect. 3. The experimental setup and results are presented in Sect. 4, and finally we conclude this work in Sect. 5.

2 Multi-robot Task Allocation for Foraging

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

2.1 Model

Our model of the task allocation problem for multi-robot foraging is based on and extends [11,17,18] where a model of the task allocation problem 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.

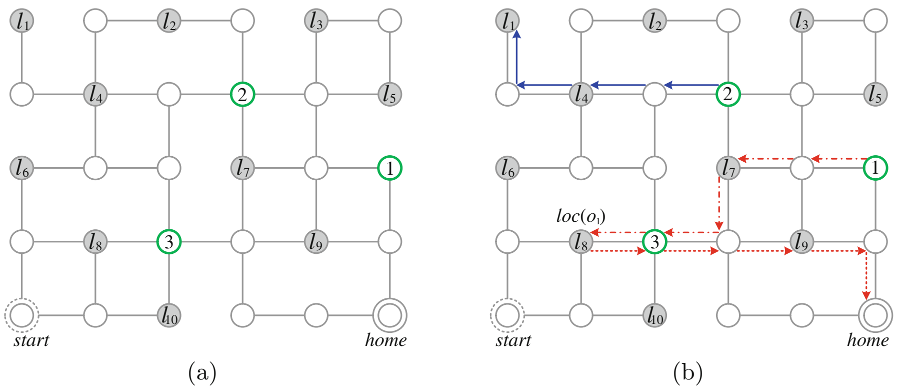


Fig. 1. Graph models of foraging tasks and the estimated costs.

For reasons of simplicity, we assume that robots need to deliver objects to a single home base (multiple bases introduce additional complexity). We use $Agt = \{1, 2, \ldots, k\}$ to denote the k robotic agents that are available for foraging. We use an undirected graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with a non-empty set of vertices $\mathcal{V} = \{v_1, v_2, \ldots, v_p\}$ and a set of edges $\mathcal{E}$ connecting vertices for representing the robots' environment, see Fig. 1(a). 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 $L = \{l_1, l_2, \ldots, l_q\} \subseteq \mathcal{V}$ 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. In our model, objects initially can only be located at target locations, so we must have that $\bigcup_{o \in O} loc(o) \subseteq L$ initially.

Another 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 foraging 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 foraging 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 delivered; $\langle red, blue, red, red, yellow, blue \rangle$ is an example goal. The idea is that the robots should search for objects in the environment of the right type and deliver these back to the home base in order. That is, the robots need to deliver a sequence of m found 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 $1 \leq i \leq m$:

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

We note that in order for the robots to be able to successfully complete a foraging 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 a matching type is delivered to satisfy the next needed type in the goal sequence. We distinguish between three kinds of goals: (i) a goal such as $\langle \mathit{red}, \mathit{red}, \mathit{red}, \mathit{red}\rangle$ 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 $\langle \mathit{red}, \mathit{blue}, \mathit{yellow}, \mathit{white}\rangle$ 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 $\langle \mathit{red}, \mathit{blue}, \mathit{red}, \mathit{yellow}\rangle$ that requires some but not all objects to have different types, i.e., $\tau_i \neq \tau_j$ for some i, j.

In the foraging problem, robots initially have no prior knowledge about the location of objects but do know which locations are possible target locations where objects can be found (i.e., they know which locations on the map are target locations). They also do not know how many objects there are in total. Robots thus need to *explore* target locations in order to locate objects. Because robots are cooperative, throughout we assume that they will inform each other about objects that are located. 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, so the robot team will be able to locate 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 has 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. This means that it may not be necessary to visit all target locations to find all the needed objects. We use T_E^i to denote the exploration tasks that are allocated to robot i, and $T_E = \bigcup_{i \in Agt} T_E^i$ for the set of all allocated exploration tasks. Note that a robot may have multiple allocated exploration tasks to perform at a moment, so it should also consider which one to execute. Once an exploration task is completed, it will be removed from T_E^i.

To complete a foraging problem, 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 delivered. For 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 r 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 in order 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 allocate retrieval tasks that at least potentially contribute to the overall goal.

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 any of the object locations. 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. Similarly, a robot may have multiple allocated retrieval tasks to complete, and the retrieval tasks that have been completed by the robot will be removed again from T_R^i. We also use $T^i = T_E^i \bigcup T_R^i$ to denote the set of exploration and retrieval tasks that have been allocated to robot i but still need to be completed.

Cost Estimate for Exploration Tasks. We use $loc(i)$ to denote the real-time 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 $l \in L$:

$$cost_E(i, l) = \parallel loc(i) - l \parallel, \tag{2}$$

where $\parallel l_1 - l_2 \parallel$ denotes the shortest travel cost for a robot to get from location l_1 to location l_2. 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 Eq. 2 by performing a graph search, for example, using Dijkstra's algorithm. As shown in Fig. 1(b), the estimated cost for robot 2 to explore l_1 takes 4 steps.

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 the retrieval task $r = \langle o, i \rangle$ for a specific object o:

$$cost_R(i, r) = \parallel loc(i) - loc(o) \parallel + \parallel loc(o) - home(i) \parallel . \tag{3}$$

For instance, as shown in Fig. 1(b), the estimated cost for robot 1 to collect object o_1 and deliver it to the home base takes 10 steps.

2.2 Problem Formulation

The foraging problem that we study here is how a cooperative team of robots Agt can most efficiently locate and deliver objects in order to achieve a goal $\langle \tau_1, \ldots, \tau_m \rangle$, where the τ_i are object types. We assume that the objective is to *minimise total completion time*.

3 An Auction-Inspired Approach for Foraging

An auction-inspired coordination framework for multi-robot task allocation in the routing domain has been introduced in [10–12]. 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. We extend these standard SSI auctions to auctions that are also able to handle *dynamic task allocation* for the foraging problem with *ordering constraints*. We first briefly discuss the standard SSI auctions and then introduce our proposed extension.

3.1 Standard Sequential-Single-Item Auctions

Standard SSI auctions are designed for *static task allocation* problems, for example, in the context of multi-robot routing [10,11,17,19], 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 [20]. 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,

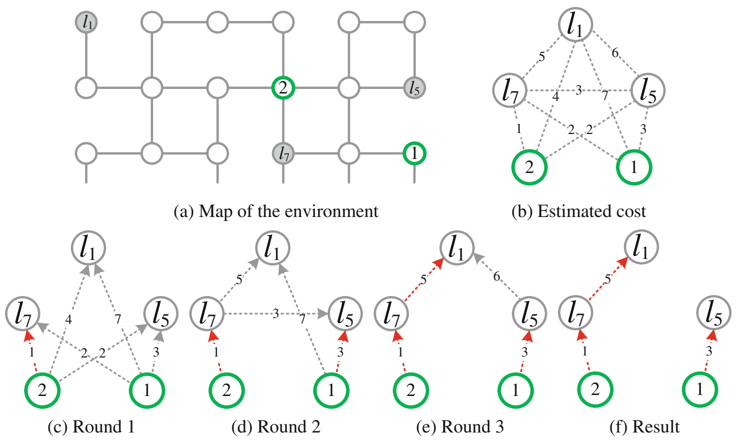


Fig. 2. With the MinMax team objective, robots aim at minimizing the maximum cost that any of the individual robots will make.

the MinMax team objective [16,17] 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 the effect of the MinMax team objective on bidding for tasks by means of an example. We use a subgraph in Fig. 1(a) 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 has 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, 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 can be free to order those tasks in any way that it wants to perform, which is called *plan modification* [20]. Searching for an optimal execution order, however, is computationally prohibitive. Typically, some heuristics are used to determine where to insert a new task into the sequence of tasks that the robot 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.

3.2 Extended SSI Auctions for Foraging

In standard sequential auctions, the tasks are known at the beginning of auctions, and, hence, such an approach cannot be directly applied to our foraging problem in which retrieval tasks *dynamically* appear when target objects are located. The work [18] 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 when choosing which allocated tasks to execute, the impact of the execution order of these tasks is not considered. 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 foraging 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 the our approach.

How to Interleave Exploring and Retrieving? In the foraging problem, although 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. 1(b)) 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 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 allocated or executed 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 robots have to explore l_9, l_5 or l_3 to find another red box. If a robot has been

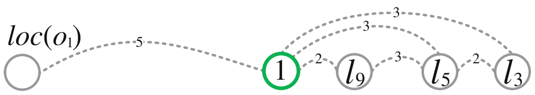


Fig. 3. First retrieve object o_1 or first explore locations close to the robot?

allocated the retrieval task to collect o_1 and the exploration tasks to explore l_9, l_5 and l_3, then the robot still needs to determine the optimal order for executing these tasks. In general, however, finding such an optimal order is NP-hard.

From the perspective of an individual robot, task allocation and execution take place in parallel in our 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 foraging 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 earlier stages of auctions, and one robot may be allocated multiple exploration tasks. Retrieval tasks appear dynamically when 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 also been located to satisfy the preceding indexed types in the goal sequence.

  In order to distribute the workload more evenly in our 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 into account the costs for completing all the exploration (or retrieval) tasks allocated to it in previous rounds. Note that it is possible that a robot has both exploration and retrieval tasks that it can bid on at the start of a session (when an object has already been found before all exploration tasks have been allocated). Robots in that case will also use the MinMax criterion to determine which task to bid on. According to the MinMax criterion, the 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 our 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 prioritise 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. For the exploration tasks, when winning an exploration task, a robot may consider re-ordering these tasks because its current location might have changed. As each robot only bids on one task in each round, it can find the optimal position to insert the new winning task into the list of previously allocated ones, which is also called *cheapest insertion heuristic* [10,20]. 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.

When to Stop Exploring? As we prioritise the execution of the retrieval tasks over exploration tasks, another issue that robots need to consider is when they should stop exploring, i.e., when to terminate the execution of allocated exploration tasks. In order to complete a foraging task, we know that the robots do not have to explore all the locations if they have already found enough objects to satisfy all the required types. If the number of available objects in the environment equals the number of objects needed, of course, it is no longer useful to explore after locating these objects, but the robots do not know how many objects there are in an environment. If the robots first explore, for example, target locations that are closest to their home base, intuitively it makes sense to stop exploring when all objects needed have been located. However, since there are two types of tasks, some robots may be allocated more exploration tasks, whereas the others may be allocated more retrieval tasks. For those who only have target locations to explore, it may still be worthwhile to continue exploring the remaining allocated but unvisited locations, because they might find another object that can be retrieved more efficiently than a teammate who is assigned to retrieve an already located object.

The issue that the robots need to decide on thus is whether they should stop exploring when all the needed objects have been located, but they still have allocated exploration tasks to perform. This involves making a decision on the trade-off to explore more at a certain cost for an individual robot but with a potential efficiency gain later for the entire team versus completing the goal at a known cost using the already located objects. Continuing exploration requires an individual robot to do more, but it is not clear whether this may benefit their teamwork. Therefore, we will investigate two heuristics, *stop-exploring* and *continue-exploring*, in our experimental study.

Algorithm. We formalize our extended SSI auction approach for the foraging problem in Algorithm 1. It shows how an individual robot (e.g., robot i) performs the foraging task, mainly consisting of bidding, re-ordering and executing procedures. 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 14). Note that since the retrieval tasks have ordering constraints, the robot 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

Algorithm 1. Extended SSI auction for the foraging problem.

```
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: • $T_E^i$: the exploration tasks allocated to robot $i$.
5: • $T_R^i$: the retrieval tasks allocated to robot $i$.
6: • $T_E$: all allocated exploration tasks.
7: • $T_R$: all allocated retrieval tasks.
8: while the remaining goal sequence has not been achieved do
9:     Update the currently available unallocated exploration tasks: $U_E = E \backslash T_E$.
10:    Update the currently available unallocated retrieval tasks: $U_R = R \backslash T_R$.
11:    procedure Bidding
12:        if $U_E \cup U_R \neq \emptyset$ then    ▷ at least an exploration task or retrieval task is available.
13:            for all $t \in U_E \cup U_R$ do
14:                Estimate $cost(i,t) = cost(T_E^i \bigcup T_R^i \bigcup \{t\})$    ▷ estimate the total travel cost.
15:            end for
16:            Bid on the task $t^*$ with the smallest cost.
17:        end if
18:    end procedure
19:    procedure Re-ordering
20:        if received winning task $t^*$ then
21:            Update $T_E^i \leftarrow T_E^i \bigcup \{t^*\}$ or $T_R^i \leftarrow T_R^i \bigcup \{t^*\}$    ▷ use *cheapest insertion heuristic*.
22:        end if
23:    end procedure
24:    procedure Executing
25:        if $T_R^i \neq \emptyset$ then
26:            Go to retrieve the first indexed object in $T_R^i$
27:        else if $T_E^i \neq \emptyset$ then    ▷ can decide when to stop executing exploration tasks.
28:            Go to explore the first indexed location in $T_E^i$
29:        end if
30:    end procedure
31: end while
```

tasks U_R in line 10. With the MinMax team objective, the robot will choose the task that minimizes the overall cost (line 16). The re-ordering procedure is used to insert a winning task announced by the auctioneer into the list of allocated but uncompleted tasks (line 19–23). In the executing procedure, the robot decides which task to execute and when to stop. In our approach, each robot gives top priority to executing the retrieval tasks as delivering objects can directly contribute the team goal. According to line 27, a robot stops executing exploration tasks if there is no exploration task to execute any more. As a result, the robot would continue with its exploration tasks until the team goal has been fulfilled by the objects delivered to the home base. If we require the robot to immediately stop executing its exploration tasks when all the required types of objects have been located, the condition of line 27 needs to be changed accordingly. 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 much tasks to complete, so it cannot offer the smallest bid to win this object.

4 Experimental Study

4.1 Simulator: The Blocks World for Teams

For the sake of repeatability and accessibility, we use a simulator, called the Blocks World for Teams (BW4T)[1], to study the foraging problem in this work. The BW4T is an office-like environment that consists of *rooms* in which coloured *blocks* are randomly distributed differently in each simulation run (see Fig. 4). The *rooms* correspond to the target locations in the foraging problem, and the colored *blocks* are the objects that can be retrieved to satisfy the team goal. Robots are supposed to search, locate, and collect blocks from the rooms and return them to a so-called *drop-zone*.

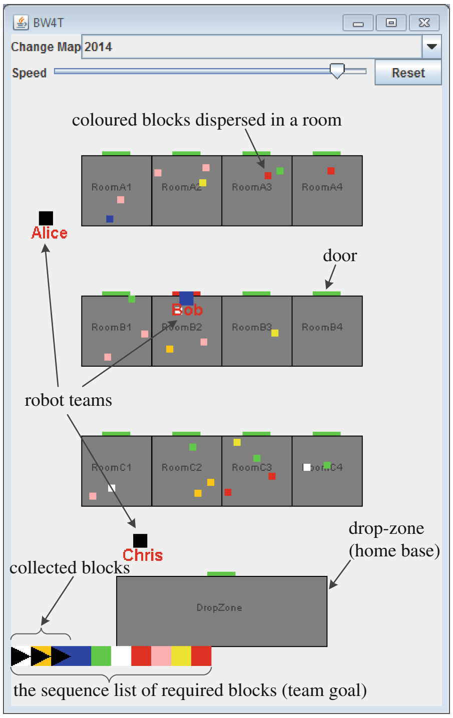


Fig. 4. The BW4T simulator.

At the bottom of the simulator in Fig. 4 the team goal of a foraging mission is indicated by the sequence of the required blocks. The required blocks need to be delivered to the drop-zone in the order indicated. Access to rooms is limited in the BW4T, and at any time at most one robot can be present in a room or the drop-zone, and robots have to go through a door to enter a room. Robots have a limited carrying capability and can only carry one block at a time.

To complete a foraging task, each robot is informed of the team goal, i.e., the sequence of the required blocks, at the start of a simulation. The robots also obtain information about room locations, but they do not initially know which blocks are present 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. In the BW4T, whenever a robot arrives at a place, in a room, or near a block, it will receive a corresponding percept. A robot in a room can perceive which blocks of what color are in that room. In our experiments, each robot in the BW4T is controlled by an agent written in Goal [22], the agent programming language that we have used for

[1] BW4T introduced in [21] has been integrated into the agent environments in Goal [22], http://ii.tudelft.nl/trac/goal.

implementing and evaluating our proposed approach in this paper. Goal also facilitates communication among the agents. For simplicity, we implemented the approach with a centralized auctioneer agent for winner determination.

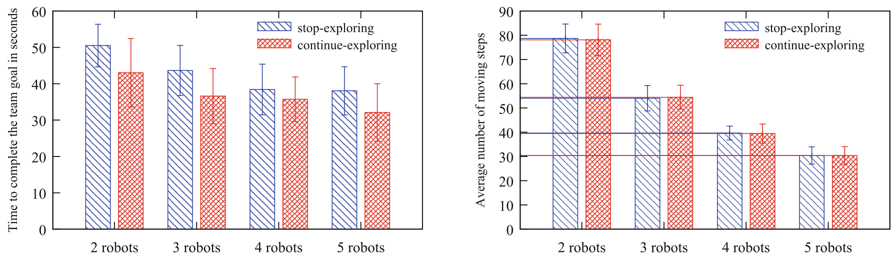
4.2 Experimental Setup

In the experimental study, we use our proposed approach to investigate whether the robots should stop exploring unvisited rooms when all blocks that are needed have been located, i.e., the *stop-exploring* and *continue-exploring* heuristics. We use robot teams from 2 robots to 5 robots, which start from more or less the same location near the drop-zone. We use a standard map with 12 rooms (see Fig. 4), and the team goal of the robots is to collect 10 colored blocks from the environment, which will be set randomly in each simulation. In order to allow for the possibility that robots find other blocks that can also contribute to the team goal in unvisited rooms even when all blocks that are needed have been located, we initialized the environment to randomly generate a total of 20 blocks in each simulation run. In our experiments, we have measured *completion time* and the *travel costs* which provide an indication of consumed energy. Each condition has been run for 50 times to reduce variance and to filter out random effects in our experiments.

4.3 Results

Figure 5 shows the performance of the robots that use the proposed approach to resolve the foraging problem in the BW4T simulator. Figure 5(a) depicts the completion time for various robot teams in a foraging mission, and Fig. 5(b) shows the average number of moving steps of each robot in a team. Figure 5(a) shows that more robots can indeed reduce completion time, but the speed up obtained by using more robots is sub-linear, which is consistent with the results reported in [23,24]. For example, using the *stop-exploring* heuristic, two robots take 50.51 s on average to complete the task, whereas four robots only take 38.42 s and thus yields a 23.94 % gain. In addition, more robots can share the workload more evenly, as shown in Fig. 5(b). The results, therefore, demonstrate that our proposed auction-inspired approach provides an efficient solution for a general foraging problem.

With regard to the issue about whether the robots should immediately stop exploring when required blocks have been located by the team, we can see in Fig. 5(a) that robots can indeed get benefit from continuing to explore unvisited rooms. This means that it would be worth the effort to explore the remaining unvisited rooms until the team goal has been achieve. For example, it can yield a 14.81 % gain for a team with two robots if doing so. Moreover, as shown in Fig. 5(b), we also find that, statistically, such a strategy does not increase average travel costs for each robot. Because we have simulated with more blocks available in the environment than needed, we can conclude from our results that if there is a sufficiently large chance to find more than one target object that satisfies an



(a) Completion time (b) Travel costs

Fig. 5. Experimental results using the BW4T simulator.

indexed type in the goal sequence, then it is more efficient to continue exploring than to stop exploring.

It is an interesting observation that continuing exploration may not increase the average travel costs. To explain this point, we first need to understand why a robot can have a chance to retrieve another block ahead of a teammate to contribute to the team goal by exploring unvisited room. We know that task allocation and task execution in our approach run in parallel. This means that each robot may have multiple allocated tasks to execute at any moment, but it needs to execute those allocated tasks one after the other. It happens that, for example, robot 1 is busy retrieving two allocated blocks (first a blue one and then a yellow one), whereas robot 2 has no block to retrieve but still has two rooms to explore. Although robot 1 can accomplish the team goal without any help from robot 2, it is possible that robot 2 can find the other yellow block when exploring its remaining allocated unvisited rooms. In this case, robot 2 has a chance to retrieve the newly found yellow block to contribute the team goal because robot 1 first has to achieve the blue color. As a consequence, robot 1 does not need to retrieve its allocated yellow one because their team goal can be finished with the active help from robot 2. In this case, indeed robot 2 increases its travel costs, but it reduces the travel costs of robot 1, which is why this strategy may not increase the average travel costs for each robot. Therefore, we can conclude that it is worth the effort to continue exploring rooms when the required blocks have been located in the foraging domain. To some extent, offering active help to teammates is a kind of altruistic cooperation, which requires an individual robot to do more but may enhance their teamwork.

5 Conclusions and Future Work

In this paper, we discussed *dynamic task allocation in a foraging problem with ordering constraints* and presented an auction-inspired approach to the problem. We performed an experimental study in the BW4T simulator to evaluate our proposed approach and investigated whether robots should stop exploring when all the required blocks have been located. Our experimental results show that the proposed approach provides an efficient approach to a general foraging problem and allow robots to share the workload in a team more evenly. In addition, a key insight into the foraging task allocation problem is that even if all the required targets have been located, robots still can reduce the time to complete their team goal by continuing to explore unvisited rooms until the team goal has been achieved. The experimental results, somewhat surprisingly, also show that such a strategy will not increase the average travel costs for each robot.

In our approach, each robot may have multiple allocated tasks to execute at any moment, and in order to distribute the workload more evenly, each robot needs to consider previously allocated tasks when bidding on a new task. In our experiments, we noticed that this may not be the most efficient task allocation strategy. For example, in the BW4T environment, 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 than the robot that found the block because it already has a heavy workload. This may happen, for example, if a robot has already been allocated many rooms to explore, which causes its bid (based on total time needed) to be rather high. In future work, we are planning to do a follow-up study in which the robots will coordinate with each other through direct communication to allocate the foraging tasks without using auctions. We aim to develop a fully decentralized coordination mechanism for the foraging task allocation, in which a robot is only allocated a new task when its previously allocated task has been completed.

References

1. Kaminka, G.A.: Autonomous agents research in robotics: a report from the trenches. In: 2012 AAAI Spring Symposium Series (2012)
2. Cao, Y.U., Fukunaga, A.S., Kahng, A.B., Meng, F.: Cooperative mobile robotics: antecedents and directions. Autono. Robots **4**, 1–23 (1997)
3. Campo, A., Dorigo, M.: Efficient multi-foraging in swarm robotics. In: Almeida e Costa, F., Rocha, L.M., Costa, E., Harvey, I., Coutinho, A. (eds.) ECAL 2007. LNCS (LNAI), vol. 4648, pp. 696–705. Springer, Heidelberg (2007)
4. Krannich, S., Maehle, E.: Analysis of spatially limited local communication for multi-robot foraging. In: Kim, J.-H., et al. (eds.) Progress in Robotics. CCIS, vol. 44, pp. 322–331. Springer, Heidelberg (2009)
5. Winfield, A.: Foraging robots. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 3682–3700. Springer, New York (2009)
6. Davids, A.: Urban search and rescue robots: from tragedy to technology. IEEE Intell. Syst. **17**, 81–83 (2002)
7. Yuh, J.: Design and control of autonomous underwater robots: a survey. Auton. Robots **8**, 7–24 (2000)
8. Hompel, M.T., Schmidt, T.: Warehouse Management: Automation and Organisation of Warehouse and Order Picking Systems (Intralogistik). Springer-Verlag New York, Inc., Secaucus (2006)
9. Parker, L.E.: Distributed intelligence: overview of the field and its application in multi-robot systems. J. Phys. Agents **2**, 5–14 (2008)
10. Koenig, S., Keskinocak, P., Tovey, C.A.: Progress on agent coordination with cooperative auctions. In: AAAI (2010)

11. Lagoudakis, M.G., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., Jain, S.: Auction-based multi-robot routing. In: Proceedings of Robotics: Science and Systems, pp. 343–350 (2005)
12. Zheng, X., Koenig, S.: 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. (2009)
13. Zlot, R., Stentz, A.: Market-based multirobot coordination for complex tasks. Int. J. Robot. Res. **25**, 73–101 (2006)
14. Dasgupta, P.: Multi-robot task allocation for performing cooperative foraging tasks in an initially unknown environment. In: Jain, L.C., Aidman, E.V., Abeynayake, C. (eds.) Innovations in Defence Support Systems -2. SCI, vol. 338, pp. 5–20. Springer, Heidelberg (2011)
15. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Robot. Trans. **21**, 376–386 (2005)
16. Tovey, C., Lagoudakis, M.G., Jain, S., Koenig, S.: The generation of bidding rules for auction-based robot coordination. In: Parker, L.E., Schneider, F.E., Schultz, A.C. (eds.) Multi-Robot Systems. From Swarms to Intelligent Automata, vol. III, pp. 3–14. Springer, The Netherlands (2005)
17. Koenig, S., Zheng, X., Tovey, C., Borie, R., Kilby, P., Markakis, V., Keskinocak, P.: Agent coordination with regret clearing. In: Proceedings of the 23rd National Conference on Artificial Intelligence, AAAI 2008, vol. 1, pp. 101–107. AAAI Press (2008)
18. Nanjanath, M., Gini, M.: Performance evaluation of repeated auctions for robust task execution. In: Carpin, S., Noda, I., Pagello, E., Reggiani, M., von Stryk, O. (eds.) SIMPAR 2008. LNCS (LNAI), vol. 5325, pp. 317–327. Springer, Heidelberg (2008)
19. Koenig, S., Tovey, C., Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., Kleywegt, A., Meyerson, A., Jain, S.: The power of sequential single-item auctions for agent coordination. In: Proceedings of the National Conference on Artificial Intelligence, vol. 21, p. 1625. AAAI/MIT, Melano Park, Cambridge (1999, 2006)
20. Schoenig, A., Pagnucco, M.: Evaluating sequential single-item auctions for dynamic task allocation. In: Li, J. (ed.) AI 2010. LNCS, vol. 6464, pp. 506–515. Springer, Heidelberg (2010)
21. Johnson, M., Jonker, C., van Riemsdijk, B., Feltovich, P.J., Bradshaw, J.M.: Joint activity testbed: blocks world for teams (BW4T). In: Aldewereld, H., Dignum, V., Picard, G. (eds.) ESAW 2009. LNCS, vol. 5881, pp. 254–256. Springer, Heidelberg (2009)
22. Hindriks, K.: The goal agent programming language (2013). http://ii.tudelft.nl/trac/goal
23. Wei, C., Hindriks, K., Jonker, C.M.: 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 (2014)
24. Balch, T., Arkin, R.C.: Communication in reactive multiagent robotic systems. Auton. Robots **1**, 27–52 (1994)