Market-based Coordination in Dynamic Environments Based on Hoplites Framework

Zeynab Talebpour, Stefano Savarè and Alcherio Martinoli

Abstract— **This work focuses on multi-robot coordination based on the Hoplites framework for solving the multi-robot task allocation (MRTA) problem. Three variations of increasing complexity for the MRTA problem, spatial task allocation based on distance, spatial task allocation based on time and distance, and persistent coverage have been studied in this work. The Fast Marching Method (FMM) has been used for robot path planning and providing estimates of the plans that robots bid on, in the context of the market. The use of this framework for solving the persistent coverage problem provides interesting insights by taking a high-level approach that is different from the commonly used solutions to this problem such as computing robot trajectories to keep the desired coverage level. A high fidelity simulation tool, Webots, along with the Robotic Operating System (ROS) have been utilized to provide our simulations with similar complexity to the real world tests. Results confirm that this pipeline is a very effective tool for our evaluations given that our simulations closely follow the results in reality. By modifying the replanning to prevent having costly or invalid plans by means of priority planning and turn taking, and basing the coordination on maximum plan length as opposed to time, we have been able to make improvements and adapt the Hoplites framework to our applications. The proposed approach is able to solve the spatial task allocation and persistent coverage problems in general. However, there exist some limitations. Particularly, in the case of persistent coverage, this method is suitable for applications where moderate spatial resolutions are sufficient such as patrolling.**

I. Introduction

Coordination is of paramount importance in the development and deployment of Multi-Robot Systems (MRS). In this paper, we focus on one particular class of MRS coordination mechanism commonly known as Multi-Robot Task Allocation (MRTA) [1], [2]. MRTA algorithms vary in design and application, but their common objective is to find a mapping between robots in a team and a set of tasks that must be accomplished in order for the teams goal to be completed. The term ”task” can have different interpretations in robotics research but herein we assume it to represent a subgoal that is necessary for the overall goal to be achieved, and that can be achieved independently of other subgoals. This subgoal can be at a high abstraction level (e.g., behavioral) or at a lower level (e.g., motion planning). Multiple approaches to MRTA have been proposed. In this work, we are mainly interested in distributed approaches that can be executed by a team of robots without the explicit need for a centralized entity outside the team with perfect knowledge of the environment.

Market-based multi-robot coordination [3]–[6] is an example of such a MRTA approach. In these systems, robots act as agents trying to maximize their individual profits. Every time a task is auctioned, robots must pay a price to obtain it.

Zeynab Talebpour, Stefano Savarè and Alcherio Martinoli at Distributed Intelligent Systems and Algorithms Laboratory (DISAL), School of Architecture, Civil and Environmental Engineering, École Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland, firstname.lastname@epfl.ch

Once the task is completed, a payment is done to the robot which won the auction. The underlying assumption is that with every robot trying to maximize its individual profit, the overall team coordination and efficiency will be improved. Market-based systems can be said to have an intentional model of cooperation [7], where different tasks have to be accomplished and robots cooperate explicitly, often through communications, to correctly allocate resources to tasks.

As Gerkey states in [1], ”if the robots are deliberately cooperating with each other, then, intuitively, humans can deliberately cooperate with them, which is a long-term research goal of multi-robot research”. Moreover, using an intentional model of cooperation it is more likely that the resulting behavior of the system is easily understood and predictable by the humans present in the environment, a key feature for social robots. There exist a variety of different solutions to MRTA. On the centralized end of the spectrum, most approaches tend to treat MRTA as a combinatorial optimization problem and use standard algorithms to solve it [8]. The Broadcast of Local Eligibility approach [9] and the L-ALLIANCE architecture [7] are examples of solutions that consider MRTA as an Optimal Assignment Problem.

On the other end of the spectrum, the auction algorithm [10], [11] exhibits the distributed nature required for distributed systems. Further research on the auction algorithm [12] has shown that it can be implemented in a distributed fashion. Another distributed yet different approach from market-based coordination is the threshold-based allocation used by [13], [14]. A comparison between threshold-based and market-based approaches can be found in [15].

While most market-based approaches deployed on robots for path planning consider tasks as final locations, a task in the Hoplites framework [16] is composed of a sets of locations or waypoints. This framework allows for coordinating plans instead of tasks. This is the key feature differentiating Hoplites with other methods mentioned so far, except for [12], where planning for sequences of locations is done in a distributed manner. Hoplites framework consists of two concurrent coordination mechanisms: passive and active. A passive coordination quickly produces locally-developed solutions while active coordination produces complex team solutions via negotiation among teammates. Active coordination makes the distinction between this approach and the consensus-based bundle algorithm (CBBA) proposed in [12]. The active coordination scheme allows for more flexibility in Hoplites, since robots can modify their plans after having been allocated a task or even in action, upon request of other team members.

This work is a first step towards adopting MRTA to dynamic human-populated social environments. In such problems, the number of robots are often limited and the number of tasks are usually moderate. The main difficulty for MRTA in such highly dynamic and noisy environments, is that plans are likely to change or to be rendered invalid, particularly if the robots are planning for longer periods of time. Additionally, the robots are required to perform in a socially acceptable manner in terms of navigation and interaction with people and other team members. This adds additional constraints to the planning problem. MRTA approaches are required to have enough flexibility and furthermore dedicated solutions for performing efficiently under such circumstances. Hoplites is an interesting approach that we believe is suitable for our long term goal of deploying teams of robots in social environments. Therein, passive coordination can produce local decisions for robots at small distances while active coordination can produce joint plans in terms of task allocation for all the robots in the team.

As the first step towards such a collaboration scheme, a variation of the Hoplites framework is proposed and tested in three different scenarios. Firstly. the performance of the proposed method is studied in comparison with other state of the art MRTA methods using a realistic high fidelity simulator (Webots) with similar complexity to the real world. Additionally, the problems of spatial task allocation and persistent coverage with real robots are investigated. We note that modifications are made to the original framework in order to both improve and adapt this method for solving MRTA problems. Namely, the replanning has been changed to include turn taking and priority planning for avoiding and resolving conflicts respectively. Moreover, instead of estimating the plan time, that is very sensitive to the unpredictable changes in a dynamic environment, we have opted to take the plan length into account.

II. Market-based Multi-robot Coordination

Complex cooperations between robots that require high computational and communication costs are not a necessity at all times. Making local decisions can be sufficient in a number of situations. However, more complex situations require advanced forms of cooperation where at the expense of higher costs, a considerably better overall solution can be found. This is the key idea behind the Hoplites framework described in the next section.

A. Hoplites Theory

Hoplites [16] is not bound to any particular planning method or in general problem-specific feature and only deals with robot cooperation. Hence, it's a powerful framework to be used in different applications as we aim to show in this paper. It allows for coordinating plans instead of tasks. A *plan* is a sequence of tasks and computing the cost and the revenue of a plan depends on the problem. This framework consists of two main concurrent coordination mechanisms: *passive coordination* and *active coordination*.

In **passive coordination**, each robot chooses its most profitable plan and broadcasts it to other teammates without any attempt to modify their plans. This information is then used by other robots to reevaluate the expected profitability of their current plans, update and broadcast the changes.

Since robots can affect the actions of one another and change their plans at any point of time, they cannot be very confident about their estimate of the profitability of their actions. Sometimes a robot's best plan can only be marginally profitable and a team plan could result in a higher profit. This indicates modifying the plans of the robot's teammates. This implies that the requesting robot asks its collaborators for compensation price quotes and persuades them into cooperation. This process is ruled by a market-based approach and constitutes the **active coordination**.

The decision of switching to the active coordination mode (explained in Sec.III-A) is based on the evaluation of a **balance function**. For robot r_j and a given plan P_k the balance function is generally defined as follows:

$$B_{j,k} = R_{j,k} - C_{j,k} - Penalty_{j,k} \tag{1}$$

where R is a generic revenue function, C a generic cost function and $Penalty$ the penalty for constraint violations. Note that this is a *local* balance function: the costs, revenues and penalties are related to a single robot. The local balance function is strongly problem-dependent and can contribute to reaching the globally optimal solution on the team level if chosen correctly. Additionally, a problem-dependent *global* balance function is also required for team-level evaluations.

B. Proposed Method

Since real social environments are noisy and dynamic, it is required to ensure the validity of plans. Invalid plans can be the result of changes in the environment or other robots changing or stopping their current plans. Therefore, a turn taking mechanism is introduced into the Hoplites framework to avoid computing costly invalid plans. This implies only one robot planning at a given time.

To reduce the drawbacks of this choice, two improvements have been made. Firstly, the robots can choose to immediately replan for avoiding conflicts or some particular situations whenever the need arises. This is done by requesting priority from other robots in a distributed manner. Secondly, robots do not wait for their turn to start replannig but rather compute and store a new plan without following it during the other robots' turn. At the start of a robot's turn, if no conflicts were detected, the stored plan is used. Otherwise, the robot will replan again. This leads to speeding up the team performance. Assigning and communicating the turns can be done in a centralized manner using a supervisor or in a distributed fashion by reaching a consensus among robots.

Another difference between our method and Hoplites is basing the planned coordination on the maximum number of tasks rather than a predefined time. This abstraction of time, allows for a more robust handling of uncertainties and makes the approach less sensitive to the unpredicted changes that might affect the estimated time for the plans. Although this aspect directly affects the planner, it's a design choice that is far more suitable for dynamic social environments where an accurate estimate of the time to accomplish a plan is not guaranteed. While this work has not been tested in such environments, it constitutes a baseline that is intended to be used in environments shared with people in the future. Therefore, it is essential to opt for a robust feature that is not very sensitive to uncertainties and noise.

The main drawback of this choice is that robots may decide to take tasks irrespective of their costs in terms of time and, in the case of spatial task assignment, favor further away tasks. However, distance and therefore an estimate of time are accounted for while computing the balance of a plan. Ergo, by tuning the weights of the balance function appropriately the desired behavior can be achieved. Moreover, through active coordination an initially suboptimal assignment of a task can be modified by selling it to a robot that is closer if available. It is important to note that this balance can be computed locally by the robots with partial information or in a centralized manner. We have chosen the first approach in this work.

To describe the method in detail, consider a set of n_r robots and n_t tasks, where each task can be assigned to only one robot. The cost of completing a task $\{t_i, i = 1, \dots, n_t\}$ for robot $\{r_j, j = 1, \dots, n_r\}$ is denoted by c_{j,t_i} and the corresponding revenue is r_{t_i}.

Using a revenue function R, and a cost function C, the robots, who are self-interested agents in pursuit of individual profit, can evaluate each available task and decide whether to take it or sell/buy it from another robot. Details of the described method can be found in Algorithms 1, 2, and 3. Note that each robot finds a plan using the coordination mechanisms in a distributed manner. Similar to Hoplites, it is assumed that a group of robots will only accept to cooperate if the requesting robot is able to pay all of them the compensation price.

Algorithm 1 Passive Coordination

```
1: procedure PassiveCoordination($Tasks, r_j$)
2:     $T_{avail} \leftarrow \emptyset$                          ▷ Set of available tasks
3:     ▷ Tasks contains all tasks regardless of their availability
4:     for $t_i \in Tasks$ do
5:         ▷ Checks if the task is unassigned
6:         if IsAvail($t_i$) then
7:             $T_{avail} \leftarrow$ Add($t_i$)
8:     $Plan \leftarrow$ Planner($T_{avail}$)
   return $Plan$
```

Algorithm 2 Active Coordination

```
1: procedure ActiveCoordination($Tasks, r_j$)
2:     $T_{avail} \leftarrow Tasks$                      ▷ Set of available tasks
3:     $R_c \leftarrow \emptyset$                        ▷ Set of robots in conflict with $r_j$
4:     $Plan \leftarrow$ Planner($T_{avail}$)
5:     ▷ Find conflicting robots
6:     $R_c \leftarrow$ RobotInConflict($Plan$)
7:     ▷ Find the tasks of a given plan
8:     $T_j \leftarrow$ GetTasks($Plan$)
9:     $T_{avail} \leftarrow T_{avail} - T_j$
10:    for $r_k \in R_c$ do                  ▷ Reset the plans of other robots
11:        $Plans_k \leftarrow \emptyset$
12:    ▷ Take the most profitable plan and start replanning for other robots
13:    for $r_k \in R_c$ do
14:        ▷ Plans for the robots in conflict
15:        $Plans_k \leftarrow$ PassiveCoordination($T_{avail}, r_k$)
16:        $T_k \leftarrow$ GetTasks($Plan_k$)
17:        $T_{avail} \leftarrow T_{avail} - T_k$
   return $Plan, R_c, Plans$
```

III. Test Cases

To evaluate the effectiveness of the proposed method, we have tested three variations of the MRTA problem with increasing complexity. Initially, the proposed method was tested against a sequential greedy algorithm (SGA) [12] and the consensus-based auction algorithm (CBAA) [12] in simulation, to provide insight about the performance of the methods in a realistic noisy environment. SGA is a single task centralized method that assigns the task with the minimum distance to each available robot. CBAA is another single task method that is shown to provide similar solutions to SGA through the auction and consensus mechanisms, in a decentralized manner.

Algorithm 3 Market-based Coordination Alg. for robot r_j

```
1: ▷ This is the procedure running on each robot
2: procedure MarketBasedCoordination
3:     $currentPlan \leftarrow \emptyset$
4:     $storedPlan \leftarrow \emptyset$
5:     $coopAccepted \leftarrow 0$
6:     ▷ While there is a task to be assigned
7:     while NotEmpty(GetTasks()) do
8:         if myTurn($r_j$) then
9:             ▷ Check if the stored plan is valid
10:            if IsValid($storedPlan$) then
11:                $currentPlan \leftarrow storedPlan$
12:            else
13:                $newPlan \leftarrow \emptyset$
14:                ▷ Get all tasks regardless of their availability
15:                $Tasks \leftarrow$ GetTasks()
16:                $newPlan \leftarrow$ PassiveCoordination($Tasks, r_j$)
17:                ▷ Compute the revenue
18:                $R \leftarrow$ GetRevenue($newPlan$)
19:                if $R \leq Threshold$ then
20:                    $activePlan, R_c, Plans \leftarrow$ ActiveCoordination($Tasks, r_j$)
21:                    $coopAccepted \leftarrow$ AskForCoop($R_c, Plans$)
22:                    if ($coopAccepted$) then
23:                        $newPlan \leftarrow activePlan$
24:                $currentPlan \leftarrow newPlan$
25:        else
26:            $storedPlan \leftarrow$ PassiveCoordination($Tasks, r_j$)
27:            if ReceivedCooperationProposal() then
28:                EvaluateProposal()
29:                BroadcastAnswer()
```

Additionally, to show the flexibility of the approach, we tested the proposed method for the problems of spatial task allocation and persistent coverage, in simulation and reality. The two test cases are inherently similar but in the case of persistent coverage, an additional dimension of time is added, making the problem continuously recurring. This section describes the planner that is commonly used in all cases and later on explains the details of each test case.

A. The Coordination Planner

Since the focus of this work is not optimizing the plans but rather investigating the capabilities of this approach, a simple and computationally inexpensive planner based on single step optimization has been used. This choice is motivated by the nature of dynamic environments and the fact that a plan can be rendered invalid or suboptimal at any time. Therefore, optimizing the immediate decision is of higher importance and step-by-step optimization is justified.

This planner is used for both types of coordination. However, for active coordination it should additionally consider team plans. Active coordination occurs when the revenue of the current plan is too small and there exists a task for which the *expected added revenue* is higher for the replanning robot compared to the currently assigned robot. The expected added revenue is the difference between the balance of the robot *with* the task and the balance of the robot *without* the task.

The planning horizon should be chosen to accommodate to the dynamicity of the environment. In highly dynamic environments long planning horizons are no longer effective and incur unnecessary computational costs without contributing much to a better plan compared to shorter planning horizons. As an example, for a robot that relies on its path planning for computing bids, the changes made to the environment by the movement of people, can largely modify the plans. This is exacerbated, for longer plans which have a higher probably of change. This is the main reason of not including the consensus-based bundle algorithm (CBAA) [12] in our comparisons. CBBA is most effective in problems with long planning horizons where it can optimize the plan of each robot and the global plan of the team by finding the best sequence of task to be performed by each robot in a decentralized manner. If a single task or very short sequences of tasks are to be considered, the superiority of CBBA to single task methods maybe overshadowed by its higher computation cost.

B. Spatial Task Allocation Based on Distance

Given the formulation in Sec.II-B, a team of robots decides how to efficiently subdivide a set of tasks that will induce optimizing a global criterion. This global criterion can be a function of time, distance travelled, etc. In this case, the tasks are moving to a specific location in the environment. These tasks can be identified locally by the robots through on-board perception or can be broadcasted to all robots by an external source. Many applications such as patrolling, attending service requests, etc., can benefit from this functionality.

In this test case, three metrics of individual robot contribution, total time and total distance of the assignment problem have been considered to evaluate the performance and the behavior of the MRS. On the local level, each robot tries to maximize a **local balance function** that is inversely proportional to the length of the path planned by FMM to a given task location.

C. Spatial Task Allocation Based on Distance and Time

Similar to the previous case, a number of tasks are associated to specific locations in the environment and the robots should find the most appropriate assignment for optimizing the global balance function. The main difference between this test case and the previous one, is incorporating time as a factor in the local balance function. The consequence of adding this factor is to encourage robots to reach tasks as early as possible, leading to increased parallelism in the team. The **global balance function** for this problem is defined as the completion time of the team objective. The **local balance function** for robot r_j, given a plan P consisting of tasks t_i, is computed in the following. Note that this function is also problem-specific. It is an instance of Eq. 1 without the penalty term. This is due to constraint violation being prevented on a higher level by replanning and in the lower level by the collision avoidance modules.

$$B_{j,P} = \sum_{t_i \in P} (r_{t_i} - \text{dist}(pos_{t_{i-1}}, pos_{t_i})) \tag{2}$$

r_{t_i} is the revenue of task t_i, pos_{t_i} is the position of t_i and pos_{t_0} is the position of the robot when starting the plan. This function includes a revenue r_{t_i} that is decreasing with time as shown below.

$$r_{t_i}(t) = \max(0, R_{max}(1 - \frac{t - t_{a,i}}{T}) \tag{3}$$

where t is the time in which t_i is reached, R_{max} is the maximum revenue for the task, $t_{a,i}$ is the allocation time of t_i and T is the time after which the positive revenue becomes zero. As mentioned previously, only an estimate of the time to reach a task can be computed in real noisy environments. However, since all robots are faced with the same limitations, this does not affect the team performance considerably.

D. Persistent Coverage

This problem consists of continuously covering an area with a group of robots. It has many applications such as cleaning, heating, etc. This is a more challenging problem compared to the previous test case. The same approach used in Sec.III-C is used here. The robots need to reach designated points in the environment with the purpose of maintaining a desired coverage level over time. Persistent coverage entails a continuous assignment of locations to robots. The coverage level of a point is maximized upon a robot reaching the point.

The **global balance function** in this case is based on the coverage function of [17] where a set of points discretizing the 2D space is considered. Each point is assigned an initial coverage value and at every time step its coverage level is decayed with a predefined rate of $\delta < 1$. If a point is sufficiently close to a robot its coverage level is increased as shown in Eq.4:

$$v(p_{i,t}) = \delta v(p_{i,t-1}) + K \sum_{j=1}^{n_r} f(r_j, p_{i,t}) \tag{4}$$

where $v(p_{i,t})$ is the coverage level associated to the point p_i at time t and f is defined as:

$$f(r_j, p_{i,t}) = \begin{cases} 1 & \text{if } d = 0 \\ \frac{R_f\ - d}{R_f} & \text{if } 0 < d \leq R_f \\ 0 & \text{if } d > R_f \end{cases} \tag{5}$$

where $d = \text{dist}(pos_{r_j}, p_{i,t})$, and pos_{r_j} is the position of robot r_j, and the radius that the robot can cover is denoted by R_f. f modulates the increase of coverage at any point. It assumes the maximum value in the center of the robot and then decreases linearly.

The **local balance function** is composed of two terms: the revenue and the cost. Considering a plan P consisting of tasks t_i, the balance function for robot r_j is defined as:

$$B_{j,P} = \sum_{t_i \in P} \left(R_{t_i} - KC_{t_i} \right) \qquad K > \frac{rate_r}{v_{max}} \tag{6}$$

where

$$\begin{cases} C_{t_i} = i \ \text{dist}(pos_{t_{i-1}}, pos_{t_i}) \\ R_{t_i}(t) = \min\left(R_{max}, R_{min} + (R_{max} - R_{min}) \frac{t - t_{a,i}}{T} \right) \end{cases} \tag{7}$$

pos_{t_0} is the position of the robot r_j at the start of the plan, t is the time when r_j reaches the task, T is the time after which the revenue is R_{max} and $t_{a,i}$ is the allocation time of t_i. Note that the cost is dependent on the distance travelled. The revenue is similar to the previous case, but it is increasing with time rather than decreasing. This is because points that have remained uncovered for longer will contribute more to reaching the desired coverage level. Hence, they should have a higher priority.

If the revenue increases with time, the robots will favor longer paths to a given point, to paths that are more optimal in terms of distance and time. This is the reason K is introduced. With a sufficiently large K the increased travelling cost will be higher than the increased revenue for longer paths. Given the maximum speed of the robot v_{max} and the increasing rate per second of the revenue $rate_r$, each unit of distance must cost more than $\frac{rate_r}{v_{max}}$ for the robot to move directly to a task rather than taking a longer trip.

Additionally, a multiplier factor i is used in the cost function to penalize the tasks of the later steps. This is necessary for avoiding some counter-intuitive side effects i.e., increasing the revenue causes the robots to give a larger revenue to tasks that are reached later in the plan, since more time has passed. This leads to situations such as a robot taking a task as its second task while it is better to have that task assigned to another robot as the first task. This is an undesired situation since we want to minimize the time for reaching a desired coverage level for the team.

IV. Navigation Framework

In this section we will briefly explain the underlying navigation used for the robots. The navigation system is that of the MOnarCH project [18], detailed in [19]. As input, it uses the pose estimates provided by a standard AMCL self-localization system, given odometry, laser range finder readings, and a static map. The navigation system is based on FMM for motion planning, together with a Dynamic Window Approach (DWA) algorithm for guidance and obstacle avoidance. FMM and DWA run asynchronously. FMM is activated when a new goal position is given, and DWA is running in a closed loop with a fixed rate of 20 *Hz* in our experiments, using the last updated potential field from the FMM.

V. Experiments

In this section we will describe the robots used for our experiments, the simulation tool and the experimental setup.

A. Robot

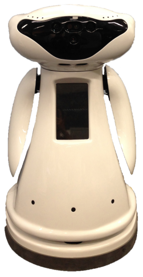
The robotic platform used in our experiments is shown in Fig. 1a. This robot is called MBot [20] and has been developed within the FP7 European project MOnarCH[1]. It is an omni-directional drive robot with an approximately round footprint of 0.65 *m* in diameter and a height of 0.98 *m*. It is endowed with two laser range finders, on both the front and the back for providing 360° coverage.

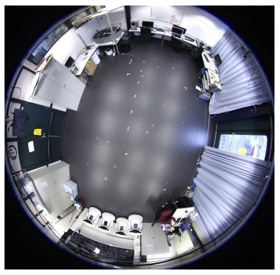
B. Simulations

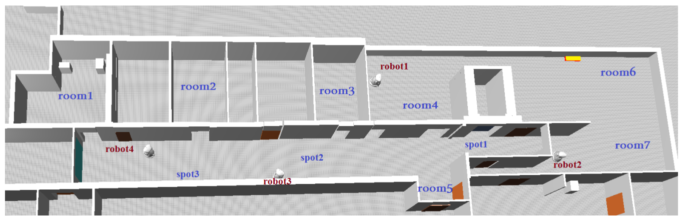
The use of high-fidelity simulators such as Webots [21] is fundamental, especially when considering multi-robot systems. Such a simulator provides a tool to perform repeated experiments under controlled conditions and also perform long-term experiments that are very expensive to have in reality. In this work, the implementation is designed based on ROS. Webots is used as a realistic simulator that allows for perfect integration with ROS and a smooth migration from simulation to reality using the exact same code.

We have developed models of the environments (see Fig.1c) that we plan to use for our experiments and a model of the MBot to match as much as possible the shape and the geometry of the real robot, namely weight, height, center of mass, position of the sensors with respect to the base frame of the robot and the position of the wheels. Additionally, calibrated models of the laser range finders (LRF), mecanum

[1]http://monarch-fp7.eu/


(a)


(b)



(c)

Fig. 1: a) The MBot robotic platform. b) Snapshot of the robotic arena in the experimental setup. c) Screenshot of a the environment of test case one, with four MBots and 10 tasks in a Webots world.

wheels and motors have been added. These data were gathered directly from the real robot. The mentioned models were all designed in the scope of MOnarCH project. Much effort in terms of optimization of computational costs has been dedicated to dealing with the challenges of faithful and real-time simulations of this rich set of sensors in complex environments, particularly in the presence of multiple robots. As a result, our simulations have similar complexity to the real world experiments. This is key, for evaluating the performance of different methods for a MRS targeting social, dynamic and noisy environments.

C. Experimental Setup

We have used a suite of experiments both in simulation and reality, for performance evaluation. Initially, we tested the performance of the proposed method with planning horizon of one (H1) and planning horizon of two (H2) in comparison with SGA and CBAA in simulation. The test consists of 10 repeated experiments, with four robots and 10 tasks. The simulation environment is a realistic model of the pediatric ward of the IPOL hospital in Lisbon where the MBots have been deployed and tested in the context of the MOnarCH project. Fig.1c shows the environment and the initial placement of the robots and the location of tasks. This dynamic noisy environment is the type of environment that we are targeting in our research. In this case, the noise is the result of the localization error that can be different for different parts of the map. Moreover, since the navigation is realistic, the movement of any robot can also modify the trajectories of the others due to obstacle avoidance.

The two other test cases have been tested in a laboratory environment of 5 $m \times 7$ m shown in Fig. 1b, with up to three robots in simulation and two robots in reality. The results shown for these two testcases are obtained from five runs. The planning horizon for both cases has been chosen to be two tasks for the reasons previously mentioned. The global balance function is used as the metric for evaluating the

Description	μ	σ
Total Distance SGA	64.83	3.70
Total Distance CBAA	68.03	2.05
Total Distance H1	51.80	4.40
Total Distance H2	52.87	4.93
Total Time SGA	75.10	6.69
Total Time CBAA	89.34	5.67
Total Time H1	62.10	5.82
Total Time H2	59.92	5.10
Robot Contribution SGA	16.21	6.26
Robot Contribution CBAA	17.00	8.65
Robot Contribution H1	12.95	3.89
Robot Contribution H2	13.22	1.75

TABLE I: Evaluation metrics for 10 simulation runs of the problem depicted in of Fig.1c. μ is the mean and σ stands for the standard deviation. The distance and robot contribution are measured in meters and the time is measured in seconds.

performance of the proposed method in each case. The tasks have been created and broadcasted using a supervisor node that is running external to the robots. The distance computed by the robots is given by the FMM planner, but for persistent coverage with high resolution we used a heuristic of Euclidean distance to decrease to computation time introduced by FMM. The ground truth for robot positions is given by AMCL.

VI. Results

In this section we show the results of our tests and quantify the differences in performance between simulation and real robot experiments. Note that the experiments are subject to noise and the robots have an average self-localization accuracy in the order of 0.2 m.

A. Spatial Task Allocation Based on Distance

Fig.2 shows four different assignment solutions from a sample run for each of the SGA, CBAA, H1 and H2 methods. For the sake of fair comparison and given that short planning horizons are more suited for our target environment, we have set the planning horizon of the proposed method to one task in H1. However, to evaluate the result of having a longer planning horizon we have also included H2 where the planning horizon is set to two. Table I contains the mean and standard deviation of the total time, total distance, and the individual robot contribution in terms of the travelled distance. It can be seen that the proposed method manages to find shorter solutions which take less time compared to both SGA and CBAA. However, SGA and CBAA exhibit a more consistent assignment in terms of the travelled distance. Despite the centralized SGA suffering the least from communication delays, since a shorter solution is found by H1 and H2, the total time of the assignment is also less for those methods. There is no significant difference between the travelled distance and the time for H1 and H2 in this problem. But H2 finishes the assignment in less time and spends less time in the idle state between tasks, as the result of two step planning.

The robot contribution is more balanced and stable (smaller μ and σ) for the proposed method, due to robots using active coordination to improve individual and team level plans. However, this will be true for the cases in which the tasks are distributed more or less uniformly in the environment. If all tasks were located in a given region close to only one robot, the proposed method would have a less balanced robot

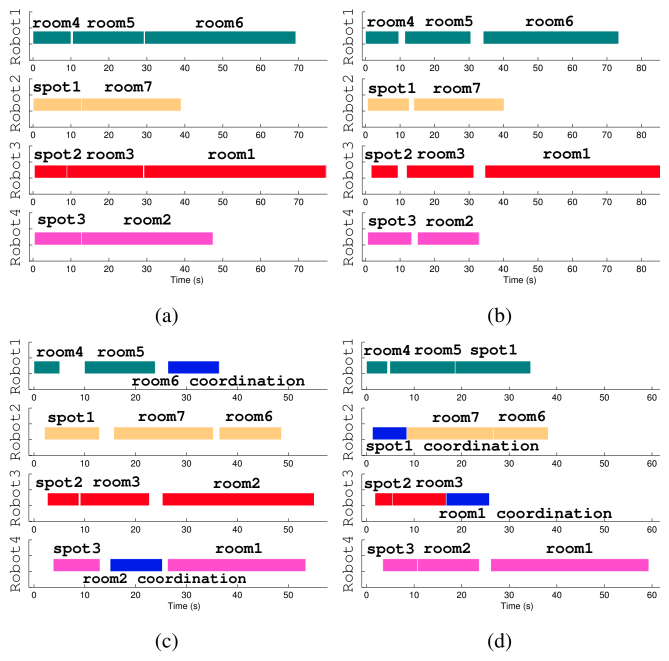


Fig. 2: Task assignment per robot over time for a sample run of the first test case, given the following methods: a) SGA b) CBAA c) H1 and d) H2.

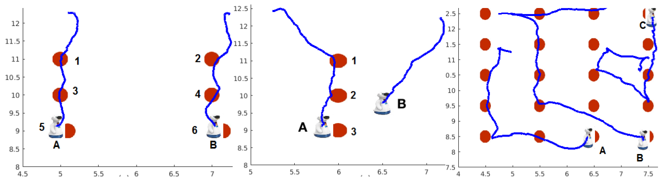
contribution compared to SGA or CBAA which will engage all the available robots.

CBAA performs similarly to SGA in terms of the total distance but takes longer due to the communication required for decentralized task allocation between the robots. These delays seem to compensate for the assignment variability caused by the localization error since CBAA has smaller σ for the total travelled distance and time. The main reason for variability of the solutions for the robots, in SGA and CBAA is the localization error and the time in which a robot finishes a task and becomes available again. Each task is allocated to the closest available robot and if robotA takes longer to reach a task due to some effort lost in improving its localization, another robot could become available before robotA and take a task that would otherwise be assigned to robotA. This is less problematic for the proposed method since the active coordination mechanism negotiates the plans with all other active and inactive robots and if the currently available robot results in a better team performance upon taking a task, the currently active robot delegates the task to it.

As an example, in Fig.2.c, Robot1 is moving toward room6 at time 28 but it receives a collaboration request (shown in blue blocks) from robot2 upon the completion of task7 and stops moving and transfers task6 to Robot2. This collaboration can also take place for longer plans, as depicted in Fig.2.d for spot1. By delegating spot1 to Robot1, Robot2 can find a better two step plan while allowing the next two step plan of Robot1 to have a larger local and global balance contribution. The gaps between robot movements relate to the communication delays in Fig.2a and 2b, and for Fig.2c and 2d they correspond to the communication delays as well as the time spent for negotiation.

-	#Robots	μ_T	σ_T	μ_D	σ_D
Simulator	2	44.36	3.45	9.78	1.44
Real robots	2	43.12	5.31	8.47	2.6
Simulator	3	43.57	9.1	6.9	2.07

TABLE II: Results of the Spatial Task Assignment problem. μ is the mean, σ the standard deviation, T the time to completion is seconds and D the total distance traveled by the robots in meters.



(a) (b) (c)

Fig. 3: a) Using passive coordination A would take tasks (1,3) and B (2,4), then A would take (5,6). However, active coordination leads to distributing the last pair of tasks rather than assigning it to one robot. b) Passive coordination would assigns (1,2) to A and (3) to B. However, (3) is more suited for A and is later assigned to it by active coordination once A is free. c) Robot trajectories and task distribution for the case of three robots. Robots are shown in map coordinates with meter scale.

B. Spatial Task Assignment Based on Distance and Time

Fig.3a-b show a case in reality where active coordination leads to achieving better results and corrects the decision of passive coordination. Fig.3c shows the robot trajectories for a set of 20 tasks with three robots. The tasks can be added to the task list dynamically. However, we chose this configuration for the ease of presentation.

Table III shows that the simulation results follow the real robot test results closely in terms of time. This similarity highlights the strength of our simulation tools. The tests with three robots were only conducted in simulation due to limitations of the available robots.

The time gain when adding the third robot is very little. Nonetheless, the mean of the travelled distance is shown to have slightly improved. The σ has increased largely for both distance and time. This could be due to the fact that more robots cause more complex situations and more complicated coordination. We can see an even distribution of number of tasks between the robots (see Fig.3c). This confirms that the time is spent in coordination rather than moving.

C. Persistent Coverage

Two sets of tests have been conducted for investigating the performance of the proposed method for the persistent coverage problem with two robots. For the sake of conciseness only real robot results are reported. Robots operate in the same environment of the previous experiment and tasks are created by means of a spatial grid. The list of parameters used in our implementation can be found in Table IV.

To understand the effect of R_f two values representing the radius of the robot footprint and its double are tested. Clearly for some applications such as cleaning, R_f should be the radius of the robot, but for some other cases e.g., heating it could be sufficient to assume a larger radius. Fig.4a shows the mean coverage level for different R_f values.

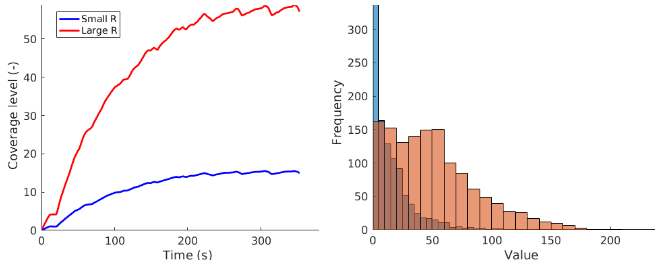
Three scenarios have been tested with varying number of tasks. Mean and variance of the coverage function along with the histogram of coverage levels are reported. The steady state coverage levels are shown in Fig.4a. The numerical results of all the cases are presented in the following table. μ indicates the mean, σ the standard deviation, and c_v is an indicator of variation defined as: $c_v = \frac{\sigma}{\mu}$

As the number of points are increased so does the resolution of the coverage and this gives a better result from the variance and mean point of view. We observe a large increase of σ when increasing R_f, despite seeing a much better coverage in terms of μ also visible in Fig.5d-f. This is the reason for introducing c_v because looking at the variance alone can be misleading here. c_v captures the mutual effect of both factors and is preferred to be smaller.

The reason for the increased variance can be seen from a different angle by looking at Fig.4b. The histogram of large R_f is more distributed and has larger values for the majority of points compared to the histogram of small R_f. The distribution of coverage values for large R_f has a higher mean but also a higher variance since the values are farther apart. For small R_f it can be observed that most points have low coverage levels and the distribution is pushed towards the lower end.

Parameter	δ	K	Small R_f	Large R_f
Value	0.99	20	0.325 (m)	0.65 (m)

TABLE IV: Parameters used in the Persistent Coverage problem



(a) (b)

Fig. 4: a) Mean coverage level over time for small R_f and large R_f. b) Histogram of the coverage values. Blue indicates small R_f and orange indicates large R_f.

VII. Conclusion and Future Work

In this work, we have proposed a method based on the Hoplites framework for solving the MRTA problem. We have been able to make improvements to Hoplites, by

-	#Robots	μ_T	σ_T	μ_D	σ_D
Simulator	2	44.36	3.45	9.78	1.44
Real robots	2	43.12	5.31	8.47	2.6
Simulator	3	43.57	9.1	6.9	2.07

TABLE III: Results of the Spatial Task Assignment problem. μ is the mean, σ the standard deviation, T the time to completion is seconds and D the total distance traveled by the robots in meters.

#Tasks	R_f	μ	σ	c_v
12	Small	14.53	25.1	1.72
20	Small	14.93	18.41	1.23
48	Small	16.01	16.01	1.43
12	Large	55.25	48.72	0.88
20	Large	57.13	35.98	0.63
48	Large	63.5	39.39	0.62

TABLE V: Coverage levels in the 6 cases studied in the Persistent Coverage problem. μ is the mean, σ the variance and c_v the coefficient of variation.

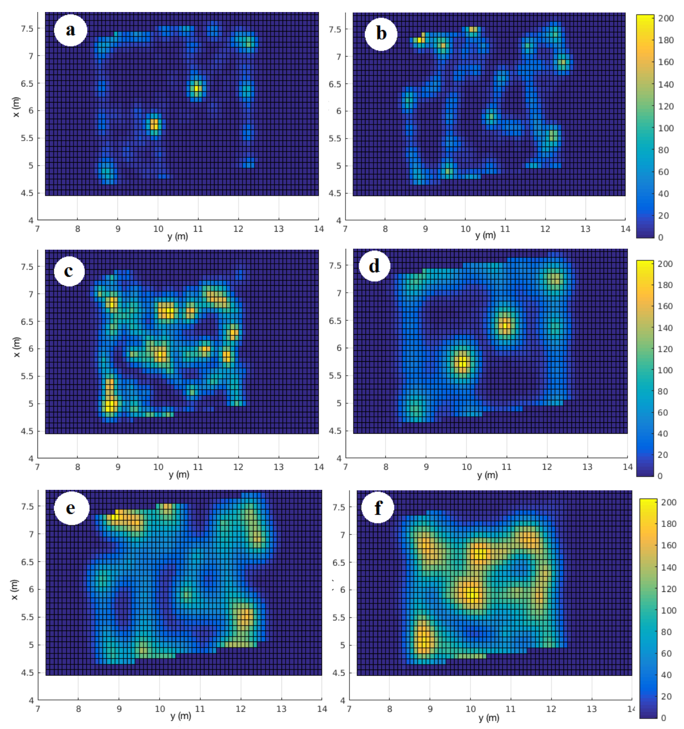


Fig. 5: Coverage values for small R_f and a) 12 b) 20 and c) 48 tasks. Steady state coverage values for large R_f and d) 12 e) 20 and f) 48 tasks.

modifying how the replanning is done and basing the planned coordination on the maximum plan length as opposed to time. The main motivation behind these changes is that in dynamic and noisy environments, plans can easily be rendered invalid and accurate estimates of time are not feasible. We have been able to apply this approach to different scenarios and demonstrated the flexibility of this coordination mechanism in solving different MRTA problems. Comparisons with SGA and CBAA methods show that the proposed method can achieve assignments that require less travelled distance and therefore take less time as well as creating a well-balanced load for the team, when tasks are uniformly distributed in the environment. The results in the task assignment problem were promising. Hoplites performs as expected in this case and the results of the simulator and the real robots show no significant differences.

For persistent coverage tests, results show that the proposed method does not achieve a homogeneous coverage level at all locations in the environment. However, it is able to reach an average steady state coverage level that can be adequate for applications that do not require a very small variation over all locations. As the covering radius R_f increases we can clearly see an improvement in the results. It is shown that the average coverage level is four times larger for a two times increase in R_f. Although more statistically significant tests are required, doubling the radius has quadruplicated the effect of the f function based on our results. Hence, this method can be appropriate for applications such as patrolling or keeping a desired temperature levels in a building, etc., where the attention is more focused on being present in a particular area rather than a specific position. The limitations seen in this case are because of the algorithm chosen to solve the coverage problem, not Hoplites.

For future work, improvements in the planning algorithm, estimation of the bids, and defining locations of the tasks to better represent the problem, will be considered. Additionally, more experiments in more complex environments i.e., social environments should be carried out. Therefore, accounting for social factors and the presence of humans in our method is another essential next step.

References

[1] B. P. Gerkey and M. J. Matarić, “A formal analysis and taxonomy of task allocation in multi-robot systems,” *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.
[2] G. A. Korsah, A. Stentz, and M. B. Dias, “A comprehensive taxonomy for multi-robot task allocation,” *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.
[3] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, “Market-based multirobot coordination: A survey and analysis,” *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.
[4] L. Xu and A. Stentz, “Market-based coordination of coupled robot systems,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2784–2789, IEEE, 2011.
[5] G. P. Das, T. M. McGinnity, S. A. Coleman, and L. Behera, “A distributed task allocation algorithm for a multi-robot system in healthcare facilities,” *Journal of Intelligent & Robotic Systems*, vol. 80, no. 1, pp. 33–58, 2015.
[6] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, “Auction-based multi-robot routing.,” in *Robotics: Science and Systems*, vol. 5, pp. 343–350, Rome, Italy, 2005.
[7] L. E. Parker, “Alliance: An architecture for fault tolerant multirobot cooperation,” *IEEE transactions on robotics and automation*, vol. 14, no. 2, pp. 220–240, 1998.
[8] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, “Network flows: theory, algorithms, and applications,” 1993.
[9] B. B. Werger and M. J. Matarić, “Broadcast of local eligibility for multi-target observation,” in *Distributed autonomous robotic systems 4*, pp. 347–356, Springer, 2000.
[10] D. P. Bertsekas, “The auction algorithm: A distributed relaxation method for the assignment problem,” *Annals of operations research*, vol. 14, no. 1, pp. 105–123, 1988.
[11] D. P. Bertsekas and D. A. Castañon, “Parallel synchronous and asynchronous implementations of the auction algorithm,” *Parallel Computing*, vol. 17, no. 6, pp. 707–732, 1991.
[12] H.-L. Choi, L. Brunet, and J. P. How, “Consensus-based decentralized auctions for robust task allocation,” *IEEE transactions on robotics*, vol. 25, no. 4, pp. 912–926, 2009.
[13] M. J. Krieger, J.-B. Billeter, and L. Keller, “Ant-like task allocation and recruitment in cooperative robots,” *Nature*, vol. 406, no. 6799, pp. 992–995, 2000.
[14] W. Agassounon and A. Martinoli, “Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems,” in *Proceedings of Autonomous agents and multiagent systems: part 3*, pp. 1090–1097, ACM, 2002.
[15] N. Kalra and A. Martinoli, “Comparative study of market-based and threshold-based task allocation,” in *Distributed autonomous robotic systems 7*, pp. 91–101, Springer, 2006.
[16] N. Kalra, D. Ferguson, and A. Stentz, “Hoplites: A market-based framework for planned tight coordination in multirobot teams,” in *Proceedings of the 2005 IEEE international conference on robotics and automation*, pp. 1170–1177, IEEE, 2005.
[17] J. M. Palacios-Gasos, E. Montijano, C. Sagues, and S. Llorente, “Distributed coverage estimation for multi-robot persistent tasks,” in *European Control Conference (ECC),*, pp. 3681–3686, IEEE, 2015.
[18] J. Sequeira, P. Lima, A. Saffiotti, V. Gonzalez-Pacheco, and M. Salichs, “Monarch: Multi-robot cognitive systems operating in hospitals,” in *ICRA workshop on many robot systems*, 2013.
[19] R. Ventura and A. Ahmad, “Towards optimal robot navigation in domestic spaces,” in *RoboCup 2014: Robot World Cup XVIII*, pp. 318–331, Springer, 2015.
[20] J. Messias, R. Ventura, P. Lima, J. Sequeira, P. Alvito, C. Marques, and P. Carriço, “A robotic platform for edutainment activities in a pediatric hospital,” in *IEEE International Conference on Autonomous Robot Systems and Competitions*, pp. 193–198, 2014.
[21] O. Michel, “Webots: Symbiosis between virtual and real mobile robots,” in *Virtual Worlds*, pp. 254–263, Springer, 1998.

Preprint submitted to 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received March 1, 2017.