Noname manuscript No.
(will be inserted by the editor)

Distributed On-Line Dynamic Task Assignment for Multi-Robot Patrolling

Alessandro Farinelli · Luca Iocchi · Daniele Nardi

Received: date / Accepted: date

Abstract Multi-Robot Patrolling is a key feature for various applications related to surveillance and security, and it has been studied from several different perspectives, ranging from techniques that devise optimal off-line strategies to implemented systems. However, still few approaches consider on-line decision techniques that can cope with uncertainty and non-determinism in robot behaviors.

In this article we address on-line coordination, by casting the multi-robot patrolling problem as a task assignment problem and proposing two solution techniques: DTA-Greedy, which is a baseline greedy approach, and DTAP, which is based on sequential single-item auctions. We evaluate the performance of our system in a realistic simulation environment (built with ROS and stage) as well as on real robotic platforms. In particular, in the simulated environment we compare our task assignment approaches with previous off-line and on-line methods. Our results confirm that on-line coordination approaches improve the performance of the multi-robot patrolling system in real environments, and that coordination approaches that employ more informed coordination protocols (e.g., DTAP) achieve better performances with respect to state-of-the-art online approaches (e.g., SEBS) in scenarios where interferences among robots are likely to occur. Moreover, the deployment on real platforms (three Turtlebots in an office environment) shows that our on-line approaches can successfully coordinate the robots achieving good patrolling behaviors when facing

Alessandro Farinelli
Department of Computer Science
University of Verona, Italy
E-mail: alessandro.farinelli@univr.it

Luca Iocchi
Dept. of Computer Control and Management Engineering
Sapienza University of Rome, Italy
E-mail: iocchi@dis.uniroma1.it

Daniele Nardi
Dept. of Computer Control and Management Engineering
Sapienza University of Rome, Italy
E-mail: nardi@dis.uniroma1.it

typical uncertainty and noise (e.g., localization and navigation errors) associated to real platforms.

Keywords Multi-robot patrolling · Distributed Multi-Robot Coordination · Dynamic Task Assignment

1 Introduction

Patrolling, that is the continuous monitoring of an environment, is an important activity for applications related to security and surveillance; using multiple robots for patrolling clearly helps improving task performance. Therefore, *Multi-robot patrolling* (MRP) is a problem that has been studied from many different perspectives in the last years, including formal representations of the problem, optimal solutions, theoretical analyses, implementation and experimental validation.

Although a relative large literature on this topic is available, experiments on real robots or on realistic simulators are rather limited. With some notable exceptions (see next section), most previous approaches focus on defining patrolling strategies according to static characteristics of the environment (for example, the layout of the map) and of the robots, without taking into account problems arising in actual deployment of real robots in a real environment.

On the other hand, perception noise and non-deterministic effects of action execution, that cannot be avoided by robots acting in the real environment, are not modeled, while they significantly affect patrolling performance. In many cases, optimal strategies obtained in ideal situations are suboptimal (or they may even fail) in a real situation. In other cases, no static solutions with good performance can be found: for example, in environments that do not contain loops (thus cyclic patrolling is not possible or inconvenient because of interferences among robots) and that cannot be partitioned in a balanced way (thus making the work of some robots unbalanced with respect to the others). The work in [14] analyzes some of these situations and advocates the use of on-line patrolling approaches strategies to address those issues.

However, to devise and execute effective on-line MRP strategies, robots must be able to coordinate their movements while patrolling, so to optimize their choices by considering up-to-date information and hence cope with the inevitable effects of uncertainty in perception and action execution. Moreover, the use of decentralized coordination approaches is a key issue for the deployment of practical MRP systems. In contrast to centralized solutions, decentralized approaches do not have a single point of failure and offer more flexible implementations (e.g., they do not require all robots to be connected to a base station at all time).

In this article, we investigate decentralized on-line coordination approaches for multi-robot patrolling, and show that dynamic task assignment techniques can be successfully employed to coordinate robot actions in presence of non-modeled characteristics of the real environment. In more detail, the present article makes the following contributions to the state of the art:

1. Formulation of the on-line coordination problem associated to MRP as a Dynamic Task Assignment (DTA) problem; this allows to use state-of-art solution approaches for DTA to perform on-line coordination for MRP.

2. Description of two dynamic, decentralized task assignment solution techniques for MRP: a greedy baseline approach (DTA-Greedy) and a market-based approach based on sequential single item auctions (DTAP); for the latter approach, we present a novel bidding strategy that departs from standard rules (such as min, avg or max path cost [27]) and is based on a dynamic partitioning of visit locations that are all close to each other, hence minimizing the possible interferences among robots.
3. Critical analysis of the performance metrics for measuring performance of on-line MRP solutions. In particular, we show that an evaluation of the methods based only on standard metrics, such as global idleness average, global idleness standard deviation and global maximum idleness, might not provide enough information to allow for a suitable comparison of different methods. Hence we propose novel forms of representing the results of a MRP task that allow for a more detailed analysis of the MRP performance.
4. Evaluation of the proposed techniques and comparison with previous methods in a realistic simulation environment for multi-robot patrolling, `patrolling_sim`[1], that is based on ROS and Stage. Moreover, an implementation of the developed algorithms has been also tested in a real environment with three Turtlebots.

The experiments show that on-line coordination is crucial to improve system performance with respect to static patrolling strategies and non-coordinated dynamic choices. Specifically, our decentralized coordination methods provide good patrolling solutions and adjust to unpredictable changes in the patrolling team (i.e., due to robot malfunctioning/failures). Moreover, we used `patrolling_sim` to perform a quantitative comparison of our proposed approaches with several state of the art techniques [19], considering both on-line and off-line approaches in several experimental settings (i.e., varying the number of patrolling robots and the floor maps). Our results, show that on-line coordination is needed to provide an effective patrolling system, and that the market-based coordination method we propose has many advantages over the other on-line coordination approaches we considered.

The rest of the paper is organized as follows: Section 2 puts our work in perspective with previous approaches for patrolling. Section 3 defines the on-line Multi-Robot Patrolling (MRP) problem we address. Section 4 provides a formulation of the on-line MRP problem as a Dynamic Task Assignment problem and proposes two DTA techniques: DTA-Greedy and DTAP. Section 5 describes the performance metrics and the experimental evaluation of the proposed approach. Finally, Section 6 concludes the article.

2 Related Work

The literature on MRP shows a significant body of work on the problem and provides a wide ranges of solution techniques. To better frame our approach with respect

[1] The simulator was originally developed by David Portugal and has been used to compare patrolling strategies in previous work (e.g., [22]). We modified the simulator to include our DTA approaches and to improve the behaviours of the robots for providing more realistic simulations. The current version of the simulator is available at `wiki.ros.org/patrolling_sim`

to the current literature, we consider three main dimensions that characterize most important algorithmic aspects of previous work in this area: i) adversarial vs. non-adversarial approaches; ii) continuous vs. discrete environment representation; iii) on-line vs. off-line solutions.

Adversarial approaches (also called strategic approaches), such as [1] or [6], consider and explicitly model the presence of an intruder trying to find the best strategy to enter the system and typically formulate the patrolling problem using game-theoretic concepts. On the other hand, non-adversarial approaches do not model the presence of an intruder and are typically used for applications such as environmental monitoring or rescue operations [26], or where no information about the intruder are available or no assumptions are made. Here we consider the non-adversarial application scenarios, hence we do not further discuss the literature pertaining to adversarial patrolling.

Several works focus on patrolling in continuous areas [11,4,3,16], where patrolling metrics (e.g., idleness) must be considered at every point of the environment. In contrast, other approaches such as [5,9,20,22] focus on a discrete representation of the environment that typically takes the form of a patrol graph, where nodes represent specific locations that the agents have to travel across and, typically, no assumptions are made on the properties of nodes or regions to be patrolled. In this work, we adopt the graph-based representation, since it allows to add a-priori information about which locations must be visited in an environment. Notice that, in the solutions using patrol graph, interferences between robots are typically considered only on the patrol nodes (i.e., when robots visit the same node of the graph). In contrast interferences that might occur on the edges (i.e., when robots are traveling between visit locations) are typically ignored. In our work, although using a graph-based representation, we provide an algorithm (DTAP) that aims at reducing also the number of interferences on the edges.

Off-line solutions [11,20] pre-compute paths that robots must follow before mission execution, while on-line solutions [19,22,25,4,18] compute (or modify) paths while robots are patrolling. On-line solutions can use most up-to-date information and hence compensate for non-modelled characteristics of the environment. On the other hand, off-line solutions typically provide optimal or near-optimal guarantees on solution quality.

Many off-line approaches model MRP through a patrol graph [15], thus allowing to apply several results from Graph Theory. Several strategies have been proposed to navigate the resulting path. Among them, graph partitioning (e.g., [20,26]) is often used as a basis for comparison. This comparison is performed using different metrics that have been suggested for determining the optimal solution: *idleness* ([9]), *frequency*[2] ([10,11]) or *exploration time* ([15]).

However, the theoretical analysis developed on graphs is typically based on rather unrealistic assumptions on the operational environments and on the behaviour of the robots. Consequently, recent research focuses on experiments with real robots or with realistic simulators, where both the features of the environment and the robot capabilities (e.g. localization, navigation, communication and perception) are taken into account. For example, Portugal and Rocha [21] and Iocchi et al. [14] use a realistic

[2] Frequency is the inverse of idleness, thus results based on idleness can be directly related to frequency.

simulator, based on a ROS implementation, while Agmon et al. [3] developed an interesting simulator that includes a model of environmental conditions (i.e., currents in a maritime scenario). Few experiments with real robots have also been performed: Elmaliach et al. [12] use a team of 3 modified vacuum cleaners to show the impact of velocity uncertainty in the motion model; Marino et al. [17] use a team of 3 Pioneer robots to patrol the perimeter of a predefined area. Recently, extending previous work, Portugal and Rocha [23] present an analysis on five different algorithms using a realistic simulator as well as experiments with five Pioneer robots.

The more realistic validations of MRP approaches show that the theoretical strategies need to be adapted to take into account the uncertainties and dynamics of the actual execution and, more generally, on-line coordination techniques are needed. Consequently, alternative approaches have been proposed, based on reinforcement learning [24], on multi-agent Markov decision processes [16], or using Bayesian learning [19].

In this paper, we develop an on-line task assignment approach to increase the robustness of MRP when dealing with the uncertainties arising from action execution in a real environment. Specifically, we propose to model on-line MRP as a dynamic task assignment problem, thus providing a general framework for on-line coordination of MRP. On-line MRP shares similarities with approaches for on-line multi-robot exploration, where robots cooperatively choose the next frontiers to visit aiming at optimizing the information acquisition process [13, 8]. However, in on-line MRP, robots must visit locations repeatedly minimizing the idleness rather than acquiring new information on the environment. This results in important differences in the employed coordination strategies. An early work considering on-line MRP is the one by Sempé and Drogoul [25], who address MRP by dynamically assigning tasks to robots. Task data are propagated through a centralized virtual world, and robots follow the gradient determined by the task strengths. Robustness to communication failures has been considered in [2], where authors empirically evaluate on real platforms different patrolling strategies with varying levels of coordination (no coordination, loose coordination and tight coordination). More closely related to our approach is the work by Ahmadi and Stone [4], where a negotiation is proposed to dynamically assign the frontier cells to be visited by the robot that has more time available. In fact, this work is very similar in spirit to our proposal. However, major differences with respect to our approach are that Ahmadi and Stone assume that robots start-off from a pre-computed partitioning of the cells, and that robots can negotiate only on frontier cells. In contrast, in our work, robots start-off with no pre-computed allocation and build a partitioning entirely through negotiation (which can involve any visit location). While using a pre-computed partitioning might be beneficial for some aspects (e.g., it would immediately provide the robots with a good allocation) such a solution might require more time to react to unexpected changes in the environment (e.g., when a robot exits/enters the system). Finally, the approach by Ahmadi and Stone is designed for a grid representation of the environment and should be adapted to work on a graph representation, that is more common in previous works. Recently, Pippin et al. [18] provide an approach to monitor robot performance in a multi-robot patrolling system (using a centralized computational unit) and to re-assign tasks when robots perform poorly. Task re-assignment is performed by using an auction-based

method where poor performing robots auction-off their tasks. We also consider on-line coordination and propose a market-based technique, but in our approach robots continuously negotiate over all available tasks. Moreover, we developed a completely decentralized system that can immediately react to unexpected changes (i.e., a robot that leaves or enters the system).

3 Problem Definition

In this section we define the MRP problem considered in this article. A set of robots $R = \{r_1, ..., r_n\}$ act in a known environment with the objective of visiting all the relevant places as often as possible. The environment in which the robots act is represented as a *patrol graph* $PG =< P, E, c >$, where P is a set of *patrol nodes*, i.e. poses (position and orientation) in the environment that the robots have to reach to take some observation, $E \subseteq P \times P$ is a set of edges connecting the nodes, c is a function denoting the expected travel time for each edge.

The MRP problem can be solved *off-line*, by defining the paths π_i before the MRP task starts, or *on-line*, by building such paths during the MRP task execution. On-line solutions to the MRP problem dynamically choose a path $\pi_i = \langle p_1, ..., p_t \rangle$, for each robot r_i so to maximize the MRP performance. These solutions are characterized by the fact that each robot updates its path, while patrolling, by considering information on the current performance of the system. Coordinated on-line solutions are also characterized by explicit negotiation among robots.

Standard performance measures for MRP are based on the *idleness* of the nodes [15]. The *instantaneous idleness* $I^p(t)$ for a node p at time t is the elapsed time since the last visit from any robot in the team. Let $\langle t_0, t_1, ..., t_k \rangle$ be the time frames in which any robot of the team visits p, then we can collect the idlenesses of node p as $\langle I^p(t_1), ..., I^p(t_k) \rangle$ (i.e., $I^p(t_j) = t_j - t_{j-1}$). From these values we can calculate the *average idleness* of a node I^p_{avg} and its *standard deviation* I^p_{stddev}. Finally, three global measures can be computed by determining average, standard deviation and maximum of all the values $I^p(t)$ for every time t and every $p \in P$. We refer to these measures as *global idleness average* I^G_{avg}, *global idleness standard deviation* I^G_{stddev}, and *global maximum idleness* I^G_{max}, respectively. I^G_{stddev} actually measures how balnaced are the visits to the nodes: low values for I^G_{stddev} mean that all the patrol nodes are visited approximately with the same frequency. While I^G_{max} represents a worst case analysis.

From these definitions, different performance metrics may be considered for evaluating the performance of a MRP system. A typical choice is to minimize the global idleness average, which is in general a good measure. However, in some cases, a good average can be obtained with a large standard deviation, meaning that some nodes are visited very often and some other nodes are visited less often. This may be unacceptable for some surveillance applications. On the other hand, it is possible to minimize the global idleness standard deviation, thus guaranteeing a more uniform visit of all the nodes, but this may be in contrast with minimizing the global average idleness. Thus, a trade-off between global idleness average and global idleness standard deviation should be considered, as it may better characterize the overal system performance. Moreover, specific needs of the application scenario may lead to prefer

some performance metric over the others. Section 5.3 provides more details about the analysis of MRP performance.

In the following, we will use the term *MRP performance* to refer to any performance metric based on the concept of idleness that is defined by the designer of the surveillance application.

4 Dynamic Task Assignment for on-line MRP

In this section we describe how to employ Dynamic Task Assignment (DTA) for developing on-line solutions for the MRP problem.

The MRP problem, as defined in the previous section, is characterized by a set of robots $R = \{r_1, ..., r_n\}$, a patrol graph $PG =< P, E, c >$, and some MRP performance metrics. The goal of the multi-robot system is to choose a path $\pi_i = \langle p_1, ..., p_t \rangle$, for each robot r_i, so to maximize such MRP performance metrics.

The DTA problem associated to MRP consists of a set of tasks $\mathscr{T} = \{\tau_1, \cdots, \tau_m\}$ a set of robots $R = \{r_1, \cdots, r_n\}$ and a reward matrix $\mathbf{V} = \{v_{ij}\}$, where each v_{ij} indicates the reward the system achieves when robot r_i executes task τ_j. An allocation matrix $\mathbf{A} = \{a_{ij}\}$ defines the allocation of robots to tasks, with $a_{ij} \in \{0, 1\}$ and $a_{ij} = 1$ if and only if robot r_i is allocated to task τ_j. The goal of the system is then to find the best assignment of tasks to robots with respect to the given reward, i.e.

$$\mathbf{A}^* = \arg\max_{\mathbf{A}} \sum_{i=1}^{|R|} \sum_{j=1}^{|P|} v_{ij} a_{ij}$$

Moreover, a set of constraints $\mathscr{C}$ usually describes valid allocations of robots to tasks (e.g., one task per agent) and hence the above optimization must be performed subject to $\mathscr{C}$.

In our patrolling problem, tasks are locations to be visited, i.e., a set of patrol nodes $P = \{p_1, \cdots, p_m\}$ and rewards depend on the average idleness of a node and on the travel cost that a robot incurs to visit such node. Specifically, we have that $v_{ij} = U(r_i, p_j, t)$, where $U(r_i, p_j, t)$ is a utility function that encodes how good is for the system to allocate robot r_i to node p_j at current time t. An example of such a utility function may be

$$U(r_i, p_j, t) = \theta_1 I^{p_j}(t) + \theta_2 Tc(r_i, p_j, t)$$

where $I^{p_j}(t)$ is the idleness of p_j at time t, $Tc(r_i, p_j, t)$ is the travel cost for robot r_i to reach p_j considering the robot position at time t, and θ_1,θ_2 are parameters that balance travel cost and idleness. Moreover, we enforce the constraint that, at any time t, only one robot should be allocated to a specific location (i.e., $\forall t, j \sum_i a_{ij} \leq 1$) to maintain a similar visit frequency across the locations.

Notice that, in our DTA problem the costs and the allocation matrices depend on time; however, as it is frequently the case in the task assignment literature, we actually solve a one shot allocation and then iterate the decision process over time.

Furthermore, notice that, our formalization of the DTA problem does not explicitly represent paths for the robots (i.e., a task is one patrol node and not a sequence of

such nodes), hence at each time step only a subset of the tasks might be allocated (this depends on the solution approach as described below). However, over time robots effectively build paths that visit patrol locations based on the current value of the idleness.

In what follows, we detail two methods for Dynamic Task Assignment: DTA-Greedy and DTAP. Both methods provide a form of distributed coordination based on explicit communication. The difference between these two methods lies in the coordination protocol and on the length of the path that robots consider. In particular, DTA-Greedy can be considered as a baseline, while DTAP is a more informed method for solving on-line MRP with DTA. In fact, in DTAP robots negotiate over all patrol nodes to build a partition of visit locations over the robots, while in DTA-Greedy they negotiate only their next patrol node and aim at avoiding conflicts (e.g., two robots heading towards the same target). In other words, DTAP allocates a subset of patrol nodes to each robot (i.e., $\forall i \sum_j a_{ij} \geq 1$), hence considering the paths that robots will follow, while DTA-Greedy always allocates one patrol node to each robot (i.e., $\forall i \sum_j a_{ij} = 1$). By considering and exchanging more information, DTAP takes better decisions and, hence has better performance (see Section 5).

4.1 Shared representation of information

The algorithms described in this section are totally distributed and thus each robot has a local representation of the information, that is an estimate of the unknown global state of the world.

Each robot r_k maintains a local copy of the instantaneous idlenesses of all the nodes, denoted by $I^P(t)^{(r_k)} = \langle I^{p_1}(t), ..., I^{p_n}(t) \rangle^{(r_k)}$. As mentioned above, these idleness values do not correspond in general to the actual idleness of the MRP task and they may be different for each robot (unless communication is broadcast and instantaneous).

The robots periodically exchange and integrate information about their status and the information sent by every robot include its local estimates of the idlenesses.

The algorithms are based on events and a new decision about the patrol path is taken when one of these two events occur: (1) the robot reaches its current target node, (2) the robot receives a message from another robot. Note that, after deciding the initial target with any method (e.g., the closest one), the first event will eventually occur. The following notation is used: r_k is the robot running the algorithm; P is the set of patrol nodes, $I^P(t)^{(r_k)}$ is a vector of instantaneous idleness of all the nodes for robot r_k, with $I^P(0)^{(r_k)} = \langle 0, ..., 0 \rangle$; p_k is the current target node for robot r_k; t_k is the last update time of $I^P(t_k)^{(r_k)}$; t is the current time when the algorithm is running; $I^P(t)^{(r_j)}$ and p_j are the instantaneous idlenesses and the current target node of robot r_j as received by robot r_k; finally, p^* is the next target node for r_k.

The algorithms presented in this section assume normal operations of the robots (i.e., no major failures occurring). This means, for example, that if a robot aims at reaching a target, it will eventually reach it. Dealing with major failures is not explicitly considered by the following algorithms. Some of these major failures (e.g., a

dead robot) are implicitly considered by the fact that the solution is on-line, and thus remaining robots will eventually reconfigure to take the tasks of the dead one.

4.2 Basic Dynamic Task Assignment (DTA-Greedy)

The first algorithm presented here is a basic DTA that performs a greedy maximization of the utility function $U(r_k, p', t)$. Decisions are taken by considering the local instantaneous idlenesses and the instantaneous idlenesses received from the other robots. The process is described in more detail in Algorithm 1, which is split in two parts corresponding to the two events to be handled.

Algorithm 1: DTA-Greedy

1 **Event** node p_k reached:
2 $I^{p_k}(t)^{(r_k)} \leftarrow 0$;
3 **foreach** $p' \in P, p' \neq p_k$ **do**
4 $\quad I^{p'}(t)^{(r_k)} \leftarrow I^{p'}(t_k)^{(r_k)} + (t - t_k)$;
5 $p^* \leftarrow \texttt{argmax}_{p'} U(r_k, p', t)$;
6 $I^{p^*}(t)^{(r_k)} \leftarrow 0$;
7 $send(< I^P(t)^{(r_k)}, p^* >)$;
8 $t_k \leftarrow t$;
9 $gotoTarget(p^*)$;
10
11 **Event** $< I^P(t)^{(r_j)}, p_j >$ received from robot r_j:
12 **foreach** $p' \in P$ **do**
13 $\quad I^{p'}(t_k)^{(r_k)} \leftarrow \texttt{min}(I^{p'}(t)^{(r_k)} + (t - t_k), I^{p'}(t)^{(r_j)})$;
14 $t_k \leftarrow t$;
15 **if** $p_j = p^*$ **then**
16 $\quad p^* \leftarrow \texttt{argmax}_{p'} U(r_k, p', t)$;
17 $\quad gotoTarget(p^*)$;
18 $\quad send(< I^P(t)^{(r_k)}, p^* >)$;

When a target node is reached [lines 1–9], $I^P(t)$ is updated accordingly and then the next target node p^* is chosen by maximizing the utility function. The value of $I^{p^*}(t)$ is also set to 0, so that the utility function for this node will decrease for the other robots, thus trying to avoid conflicts in accessing the same node.

When idlenesses of another robot are received [lines 11–18], the local values of $I^P(t)^{(r_k)}$ are updated with the minimum between the received value $I^{p'}(t)^{(r_j)}$ and the current estimates as known by r_k, $I^{p'}(t_k)^{(r_k)} + (t - t_k)$. Notice that this update rule assumes (lacking any knowledge about it) that all the other robots have not visited any other nodes meanwhile. Furthermore, notice that the algorithm updates the idleness of all the patrol nodes in P, even though the message received from robot r_j relates to a specific node p_j. Now, under the assumption of perfect communication and no robot failure this is redundant, as each robot could update only the idleness of node p_j. However, when messages can be lost, or when robots can enter the system during

the patrolling mission, such redundancy helps robots to quickly reach a common estimate for the idleness of all patrol nodes. Next, if a conflict is detected (i.e., the current target node for robot r_k equals the target node of robot r_j), a new target p^* is computed by maximizing the utility function. This means that a robot can change its destination during its path towards a node. This can happen when a realistic or a real model for communication is used (as in the simulation experiments described in this article and with real robots), in which non-instantanous messages[3] may result in two robots choosing to go to the same target node.

In DTA-Greedy algorithm, we have used the following utility function

$$U(r_k, p', t) = \theta_1 I^{p'}(t) + \theta_2 Tc(r_k, p', t) + \theta_3 d_0(r_k, p')$$

where, in addition to the terms $I^{p_j}(t)$ and $Tc(r_i, p_j, t)$ defined before, we also consider the distance $d_0(r_k, p')$ between the node p' to be reached and the initial node for robot r_k (i.e., the very first node in which r_k was when the MRP task started). This special heuristic is suitable when robots start the task in a distributed fashion (i.e., from distant locations each other) and this term of the utility function tends to keep the robots apart from each other to avoid interferences. More in general, heuristics to spread out robots have been frequently employed in multi-robot coverage [7] and exploration [8].

Of course other heuristic functions can be defined, for example, functions that are specific of the environment. The definition of a proper heuristic is not the focus of this article, since DTA-Greedy is only shown here as a baseline of a simple method based on DTA. Furthermore, good performance of this utility function depends also on a proper set-up of the parameters θ_i. In this work, no optimization of these parameters has been performed and the same set of parameters has been used in all the experiments.

4.3 DTA based on Sequential Single Item Auctions (DTAP)

DTAP is inspired by auction based task allocation: the basic idea is that robots announce their destinations to everyone and then collect "bids" from their team-mates. Such bids encode how well each robot fits to a given destination. In more detail, each robot selects the next visit node as the one that maximizes a utility function. Then the robot broadcasts its node selection, announcing its corresponding bid, to all team mates. After collecting all bids the robot checks whether it is the one with the best bid for the selected node. If this is the case, the robot visits the selected node, otherwise it selects the next best visit node and iterates the selection process.

Our market based allocation scheme takes inspiration from Sequential Single Item auctions, where robots allocate one task at the time, and when they compute their bids, they consider previous allocated tasks. This allows to take into account important synergies between tasks (i.e., patrol nodes that are close to each other). In our approach, such bid computation considers the number of tasks a robot is responsible for and the distance of the target node to the *central node*, which is the node

[3] In the experiments we used a delay of 0.2 sec for each message.

Algorithm 2: DTAP

Initialization:
$CurrentTasks \leftarrow P$; // Initialize CurrentTasks with all patrol nodes
$MyTasks \leftarrow \emptyset$; // Initialize MyTasks with the empty set

Event node p_k reached:
/* Update Idleness of all patrol nodes */
$I^{p_k}(t)^{(r_k)} \leftarrow 0$;
foreach $p' \in P, p' \neq p_k$ **do**
    $I^{p'}(t)^{(r_k)} \leftarrow I^{p'}(t_k)^{(r_k)} + (t - t_k)$;
$t_k \leftarrow t$; // update last visit time with current time
$p^* \leftarrow \texttt{argmax}_{p' \in CurrentTasks} U(r_k, p', t)$; // select best node among $CurrentTasks$
$CurrentTasks \leftarrow CurrentTasks \setminus p^*$; // remove selected node from $CurrentTasks$
$bid \leftarrow computeBid(p^*)$; // compute bid for p^*
$forceBid(p^*, bid, r_i)$; // assume this robot is the best one to patrol p^*
$sendTarget(p^*, bid)$; // send target and bid to other robots
$collectAllBids()$; // wait for bids from other robots (this inlcudes a timeout)
/* check whether this robot has the lowest bid */
if $bestBid(p^*)$ **then**
    $CurrentTasks \leftarrow P$; // reset $CurrentTasks$
    $MyTasks \leftarrow p^*$; // allocate this robot to p^*
    $p_{next} \leftarrow chooseNextNode(MyTasks)$; // choose the best task to patrol
    $gotoTarget(p_{next})$; // go to p_{next}
/* check if there are still tasks the robot should consider */
else if $CurrentTasks == \emptyset$ **then**
    $CurrentTasks \leftarrow P$; // reset $CurrentTasks$

Event msg received from robot r_j:
if $mgs.type == IP$ **then**
    /* Update Idleness of all patrol nodes */
    foreach $p' \in P$ **do**
        $I^{p'}(t_k)^{(r_k)} \leftarrow \texttt{min}(I^{p'}(t)^{(r_k)} + (t - t_k), I^{p'}(t)^{(r_j)})$;
    $t_k \leftarrow t$; // update last visit time with current time
else if $msg.type == Bid$ **then**
    /* Maintain best bid for each patrol node, update MyTasks so to
       include nodes for which this robot has lowest bid */
    $updateBids(msg.dst, msg.value, msg.sender)$;
    $updateMyTasks()$
else if $msg.type == Target$ **then**
    /* Handle task request */
    $bid \leftarrow computeBid(msg.dst)$; // compute bid for the target
    $updateBids(msg.dst, msg.value, msg.sender)$;
    $updateMyTasks()$;
    /* Check whether the bid of this robot is better */
    if $msg.value > bid$ **then**
        $sendBid(msg.dst, bid)$; // send bid

at minimum path distance from all other nodes (see below for a more detailed explanation). This is different with respect to standard bid computation rules employed in sequential single item auctions for task allocation (e.g., [27]), that typically consider an aggregation of the path cost to cover all allocated tasks (e.g., the sum, max or average of the path cost). The rational behind this choice is twofold: i) by considering the number of nodes, we foster a balanced workload among the robots and ii) by considering the distance to the *central node*, we aim at creating a partition of the patrol nodes that tries to minimise path crossing among the robots, hence resulting in less interferences for navigation. Moreover, we do not consider marginal costs for bid computation, as this could result in unbalanced allocations (as stated in [18]), where some robots might have significantly more visit nodes than others. This would be problematic for the MRP strategy as it would increase the standard deviation of the global idleness.

Our DTAP approach is described in Algorithm 2, which is again split in two parts corresponding to the two events to be handled. Whenever a robot completes a task (i.e., it reaches its current destination) [lines 5–22], it selects the next destination (p^*) as the one that maximises the utility function. Next, the robot computes its fitness for this destination (*bid*), and sends a target request ($sendTarget(p^*, bid)$) to all the robots.

After announcing its next target, the robot waits to receive the bids from the other members for that target ($collectAllBids()$); this function is blocking and a time out is used to avoid deadlocks due to possible robot failures. Notice that, waiting for a specified amount of time whenever a decision must be made, can significantly slow down the patrolling operations, hence resulting in a higher average idleness. However, this can be avoided by reasoning in advance on the future destination. In particular, robots can select the next destination, while travelling to the current target. Hence the waiting time for bids is absorbed by the travelling time to the current destination. Our current implementation employs this scheme, however, for ease of explanation, the pseudocode reported here assumes that each robot always chooses its next target after the current goal is completed.

After collecting all the bids, the robot checks whether its own bid is the best one for node p^*, if this is the case, it selects the best destination among its tasks ($chooseNextNode(MyTasks)$) and goes towards this destination. Moreover, it resets the task list, as appropriate [lines 16–20]. The choice of the next position (p_{next}) should try to maximize the system performance considering all the nodes that the robot should patrol, this requires to solve a problem that in general is not tractable as it reduces to a TSP. However, in our implementation we observed that a good and simple heuristic is to always choose the patrol node with the highest utility among the robot's tasks, hence in practice this amounts to simply set $p_{next} = p^*$.

When a message is received by another robot [lines 24–37], first the local values of $I^{p'}(t_k)^{(r_k)}$ are updated as explained for DTA-Greedy .

If the message is a *bid* (i.e., a robot is responding to a previous task request issued by this robot), the bids are updated ($updateBids(msg.dst, msg.value, msg.sender)$) so to always maintain the lowest bid for each patrol node. Once bids are updated the robot updates its tasks ($updateMyTasks()$) by modifying the structure *MyTasks* so to include nodes for which the robot holds the lowest bid.

If the message is a task request (i.e., this robot must respond to a task request of the sender), the bid of the current robot for the target is computed and sent to the message sender. Notice that, the robot communicates its bid only if it is *useful* for the sender, i.e., if the computed bid is lower than the one communicated (this reduces the number of messages for the coordination).

In order to effectively deploy sequential auctions a number of additional issues need to be suitably addressed and they are discussed below. First, the possibility that robots disappear during mission execution must be considered: if the best robot to perform a task t_i is a robot r_j and such robot disappears, the task t_i will never be executed unless someone else becomes best suited for that task, which might never happen. This is addressed when choosing the next target position after computing the best candidate node, by the $forceBid(dst, value, sender)$ function, which forces the current bid as the best one. Moreover, the next node to reach is selected independently of the tasks that the robot is responsible for [see line 10], hence if a robot disappears the task it was responsible for will eventually be considered by the others.

Another critical issue is the computation of a bid b ($computeBid(dst)$). As mentioned before, our approach is based on the concept of *central node*, which is the node that has minimum travel cost from all other nodes the robot is currently responsible for. In more detail, we compute the central node as follows:

$$cn = min_{p \in MyTasks} \sum_{p' \in MyTasks} Tc(p, p')$$

where $Tc(p, p')$ is the travel cost from location p to location p'. In our implementation, the travel cost is comuputed as the length (i.e., the number of edges) of the shortest path between the two nodes. The central node is updated each time the *MyTasks* structure changes (i.e., when the robot acquires or loses a task). Next, we compute the bid for a destination *dst* by multiplying the number of tasks the robot is responsible for by the travel cost from the central node to the destination: $b = |MyTasks| * Tc(cn, dst)$. This computation of the bid helps balancing the workload, as it penalizes robots that are responsible for too many tasks, and it considers synergies among tasks, by penalizing robots that have a central node which is far from the current destination. Notice that, a standard metric, such as for example the sum of travel cost for a tour that covers all tasks, would also achieve workload balance; however, since the bid is not related to a central node, this metric would not facilitate space partitioning. As a result, it can produce several path intersections that can increase difficuty in robot navigation.

5 Experimental Setting and Evaluation

The goal of the experiments described here is to assess the performance of the proposed MRP algorithms in realistic scenarios. As already mentioned, in order to compare the performance of the different algorithms, we have used `patrolling_sim` simulator. The simulator has been modified for improving the behaviors of the robots and for providing more realistic simulations. In the simulations reported in this article, typical odometry and laser range finder noises are used, robots use the standard

ROS navigation stack (`amcl` for localization and `move_base` for navigation), and execute their tasks at a nominal maximum speed of 1.0 m/s. Moreover, as communication among robots is implemented through the use of ROS messages, but by introducing a fixed amount of delay of 0.2 sec in the exchange of each message. This simulates a typical delay when sending TCP packets on a wireless network as experienced in the experiments with real robots. Finally, we added the implementation of the DTA methods described in this paper, thus allowing for a full replicability of all the results reported in this section.

The use of the ROS framework allows for an easy porting of the developed methods on actual robots, as described at the end of this section.

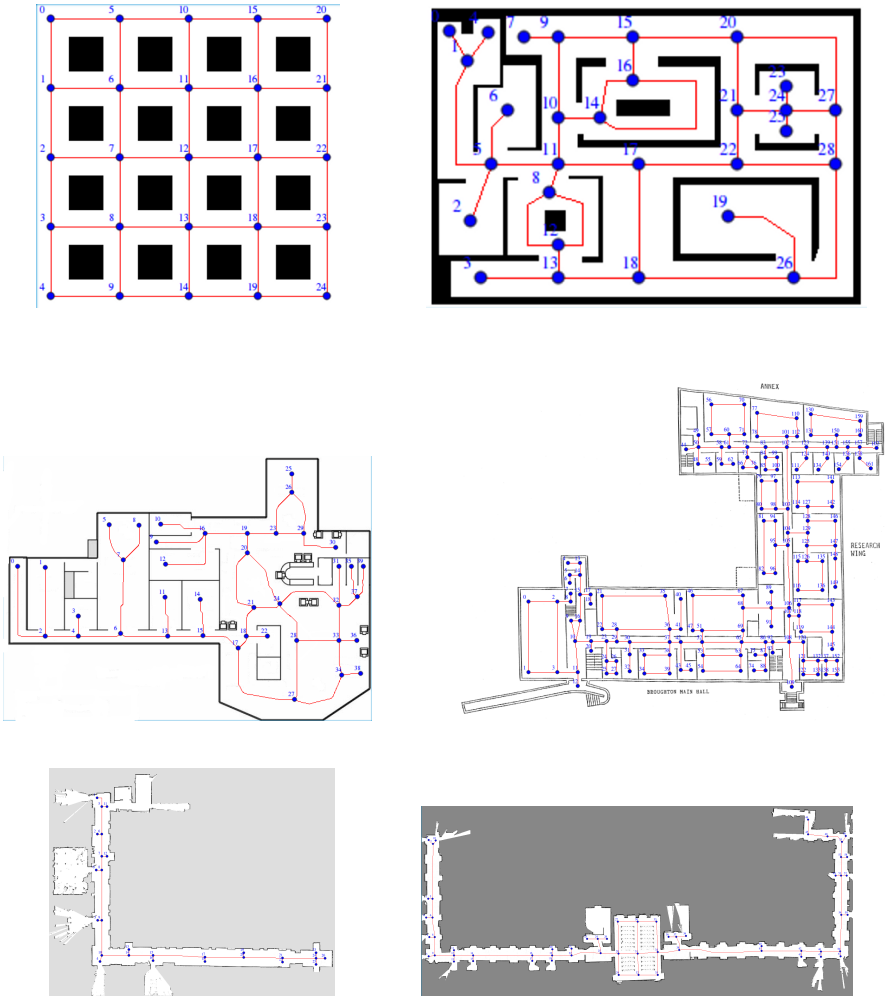

Fig. 1 Maps used in the experiments (not in scale): grid, example, cumberland, broughton, DIAG_labs, DIAG_floor1.

Map	$\|P\|$	$\|E\|$	**Size [m]**
grid	25	40	26×26
example	29	36	47×33
cumberland	40	44	52×37
broughton	163	186	100×80
DIAG_labs	27	26	50×40
DIAG_floor1	60	63	114×46

Table 1 Size of the patrol graphs used in the experiments.

5.1 Evaluation scenarios and compared algorithms

Several different evaluation scenarios have been created in order to compare different algorithms in different experimental conditions. Each scenario is defined by a map (see Figure 1) and the number of robots. Other variables of an experiment are discussed in Section 5.2.

The six maps used in the experiments are shown in Figure 1 and present different size and characteristics of the patrol graphs. Details on the number of nodes $|P|$, the number of edges $|E|$ and the overall size of the area are shown in Table 5.1.

In the experimental evaluation reported in this article, the following algorithms have been compared:

- RAND - Random algorithm chooses the next vertex randomly among the adjacent ones.
- CR - Conscentious Reactive algorithm [15] selects the next node among the adjacent ones, based on its local estimate of the idleness.
- HCR - Heuristic Conscientious Reactive [5] is an extension of CR considering also the distance of the adjacent nodes from the current one.
- HPCC - Heuristic Pathfinder Cognitive Coordinated [5] considers path of length > 1 in the graph, instead of only the next adjacent node.
- CGG - Cyclic algorithm for generic graphs [9] is an off-line method computing Hamiltonian cycles (through a fast heuristic), when they exist, or long paths and non-Hamiltonian cycles.
- GBS - Greedy Bayesian Strategy [22] selects the next vertex to reach based on a Bayesian formulation of the problem for choosing the best direction. It uses information coming from other robots about their arrival time at a given node, so that a global idleness can be estimated.
- SEBS - State Exchange Bayesian Strategy [22] is an extension of GBS in which also intentions of other robots are collected and used for determining (again with a Bayesian method) the best direction.
- DTAG - DTA-Greedy algorithm described in this paper.
- DTAP - DTAP algorithm described in this paper.

The characteristics of the compared algorithms are also summarized in Table 2. On-line vs. off-line decision making is a first classification dimension. As already mentioned, on-line methods are more robust to non-modeled effects of the environment, while off-line methods aim at optimizing the behavior of the robots given the prior knowledge about the environment. The path length determines how many nodes

Algorithm	Decision	Path length	Communication
RAND	on-line	1	-
CR	on-line	1	-
HCR	on-line	1	-
HPCC	on-line	> 1	-
CGG	off-line	> 1	-
GBS	on-line	1	arrived
SEBS	on-line	1	arrived + intention
DTAG	on-line	1	utility values
DTAP	on-line	> 1	two-steps protocol

Table 2 Summary of characteristics of the compared algorithms.

are planned to be visited at each decision. This is particularly interesting, of course, for on-line methods. In previous work, methods with path length 1 are called Reactive, while the others are called Cognitive. Finally, the table highlights the communication among robots, from no communication for non-coordinated methods, to minimum communication for methods with implicit coordination (GBS and SEBS) to explicit communication for methods based on a coordination protocol (DTAG and DTAP).

It is important to notice that the last four algorithms (GBS, SEBS, DTAG and DTAP) present some form of explicit coordination, based on communication of some information among the robots that are used to decide which path to select. In GBS, SEBS and DTAG, a single message is sent from a robot to all the others containing information that will be used to take decision. The content of the messages exchanged in SEBS and DTAG include some information about the intention of a robot to visit the next node. This feature (as shown by experimental results) allows for a significant reduction of the interferences. On the other hand, DTAP implements a two-steps coordination protocol, in which robots exchange information and negotiate their paths. Moreover, DTAP decision involves not only the next node to visit, but a subset of nodes. Therefore, DTAP implements a more informed coordination protocol with respect to the other methods.

For CR, HCR, HPCC, CGG, GBS and SEBS we have used the implementations available in `patrolling_sim` and we have added the implementation of RAND, DTAG and DTAP to `patrolling_sim`. Other algorithms could be easily integrated and compared using this simulator in the future. Previous comparisons among these algorithms (except for DTA ones) are reported in [23] and [22]. Notice that values are different in this article since we are using a different setting for the robots (e.g., 1.0 m/s of maximum speed in the experiments reported in this article vs. 0.2 m/s used in previous experiments).

5.2 Variability of the experiments

The results of the experiments are mostly based on the concept of idleness, which depends on the time needed to reach nodes in the patrol graph and thus depends on the underlying navigation module. Due to the choice of using a realistic simulator mod-

eling typical forms of uncertainty in mobile robots, some *external factors* influence the results of the experiments, but are not part of the MRP algorithms. We believe that it is important to consider these factors in the experiments and not to hide them (by using for example a more abstract simulator or less noisy modules), since they are actually present in the real world.

In the following, we briefly describe and comment the most important external factors that affect the performance of MRP task.

1. **Localization**. Localization introduces some noise in the navigation component, but in our experience this noise does not affect the overall results of multi-robot patrolling in a significant way. Indeed, we consider robots equipped with a typical laser range finder navigating in an indoor environment with a known map. In this situation, standard localization methods based on particle filters (we use ROS `amcl`) are known to be very robust and precise.
2. **Path planning and trajectory following**. The definition of the path to reach a target node and its execution also introduce noise, but also in this case standard implementations (we use ROS `move_base`) are reliable enough and do not introduce high variability in the time to reach a target goal, except when there are obstacles (see next item).
3. **Obstacle avoidance**. The feature that introduces the highest variability in the experimental results is the obstacle avoidance behavior. When a robot encounters an obstacle (which is another robot in our case, since the environment is static except for the robots moving in it), an obstacle avoidance behavior is executed (in our case by `move_base`). In some cases, the robots just slightly change their trajectory and thus the overall time to reach a node is just minimally affected by this maneuver. In other cases, however, the procedure is more complicated, specially if the avoidance occurs in a narrow passage. So the time to solve the situation may change significantly, depending on the position of the map in which the situation occurs and on the performance of the obstacle avoidance module. While `move_base` is generally good at solving these situations, in some cases it requires a significant amount of time and in some other cases a deadlock occurs and the robots remain stuck. It is important to notice that obstacle avoidance is activated only in case of interferences and that an ideal MRP system should reduce interferences as much as possible.
4. **Communication delay**. For MRP algorithms that rely on communication among robots, the communication model is a relevant factor that affects performance. In this article, we assume that a wireless network infrastructure is available to the robots and that they use TCP communication. From our experience in real settings, wireless communication introduces a delay of messages of up to 0.2 seconds. As already mentioned, this delay has been considered in the experiments in order to provide more realistic results.
5. **Initial poses of the robots**. For the initial poses of the robots we have considered two situations: 1) robots start in predefined positions (the same in all the experiments for a given map and a given number of robots and the same used in previous experiments by other authors) that are scattered through the environment; 2) robots start from a cluster of positions close each other; this set-up is a

more realistic one in which we assume robots are all started from a given starting area. As shown in a set of specific experiments, when most effective algorithms are considered (SEBS and DTAP in our case), the initial position of the robot does not affect significantly the overall performance.

In summary, the main factor that increases variability of the experimental results is the obstacle avoidance behavior, that is activated to solve interferences. As already mentioned, we have added the interferences as a performance metric. Moreover, we have implemented a mechanism to detect when results are strongly affected by the obstacle avoidance procedure. More specifically, we have stopped all the experiments in which a robot is not able to reach a target location within 5 minutes. These experiments are not considered in the results, since the performance based on the idleness is not suitable for a fair comparison with another run in which this situation did not occur. Moreover, as explained in the next section, we have discarded all the experiments that have low correlation with the majority.

5.3 Performance metrics

In order to assess the performance of a MRP system, several performance metrics and evaluation procedures have been proposed in previous work. Most of them are based on the concept of idleness that measures time distance between two consecutive visits of a node by any robot in the team. Therefore, as already mentioned, *global idleness average* I^G_{avg}, *global idleness standard deviation* I^G_{stddev}, and *global maximum idleness* I^G_{max} are typical performance metrics used to compare different systems. However, it is important to notice that very often a trade-off among these metrics allows for a better comparison of different algorithms.

In this section, we discuss the results of a first set of experiments (named **Experiment Set 1**) aiming at presenting and discussing the different performance metrics and their visualization. The experiments have been performed in the following settings: Cumberland map with 4 robots, 1 hour of duration, and three algorithms compared: a random method (RAND) in which each robot selects the next target node randomly among the adjacent ones, SEBS algorithm [22], and DTAP algorithm described in this article. For each algorithm, three runs have been executed. In the following, we present the results of this experiment in different forms in order to discuss them in details.

A first representation of the results is given in Table 3, which reports the values I^G_{avg},I^G_{stddev}, and I^G_{max} for each run. This is the typical way in which MRP results have been presented in previous papers.

From this table, it is possible to notice that the difference among the average values I^G_{avg} in the different runs in some cases is not statistically significant, given the high standard deviation. Moreover, the RAND algorithm has even generally better performance if only the average idleness is considered, but this is obtained at the cost of a substantially higher standard deviation and maximum value. Both SEBS and DTAP have significant lower standard deviation and maximum values. The differences between SEBS and DTAP are further analyzed in the next sections. This

Algorithm	I^G_{avg}	I^G_{stddev}	I^G_{max}	If_{rate}
RAND_1	117.2	216.2	2066.0	6.857
RAND_2	**99.3**	199.8	2429.9	2.663
RAND_3	102.6	201.6	2044.8	2.848
SEBS_1	114.3	100.3	567.1	0.100
SEBS_2	113.8	102.8	575.2	0.216
SEBS_3	112.8	102.4	687.4	0.283
DTAP_1	142.6	57.1	371.3	0.167
DTAP_2	140.2	55.4	**306.9**	**0.083**
DTAP_3	135.7	**54.2**	320.5	0.117

Table 3 Exp. ES1: cumberland, 4 robots, RAND, SEBS, and DTAP algorithms. Best values in bold.

table shows also the number of interferences per minute (last column) that is a metric explained later in this section.

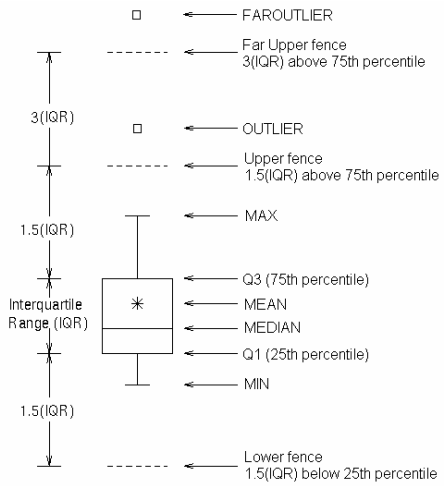

Fig. 2 Meaninf of boxplot representation.

A different format for representing the results of a MRP task is given by *boxplots*, whose definition is presented in Figure 2. Figure 3 shows the same results of this experiment using boxplots. From this view, the difference between the algorithms appears more clearly. RAND algorithm achieves a low average thanks to many small values and a few high ones, while SEBS and DTAP provide for a more balanced set of values having smaller standard deviation and few outliers and high values.

Another view of the results is given in the form of a temporal plot of idleness values over the time of an experimental run. Figure 4 shows a plot for one run for each of the algorithms considered in this experiment. From these plots, we can analyze the distribution of the idleness values and the temporal evolution of the performance metrics. We can see that the majority of values in RAND are low values, while in

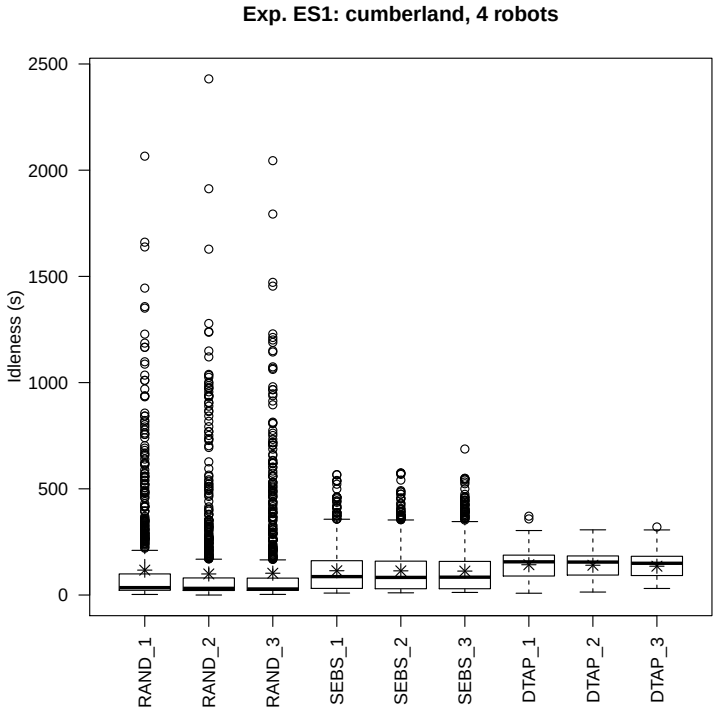

Fig. 3 Boxplots of comparison among the three algorithms.

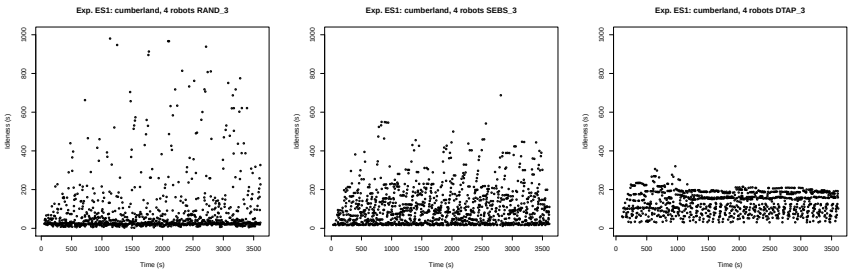

Fig. 4 Temporal plots of comparison among the three algorithms: RAND, SEBS, DTAP.

DTAP the majority of values are around 200 seconds. Notice also that some values in RAND are not shown in the leftmost figure, since the idleness scale (Y axis) has been limited to 1000 s, while RAND generates values also above this limit. Finally, temporal plots are useful to see if an algorithm has some transient behavior. For example, in DTAP_3 outliers are present only in the first phase of the experiment. A

more detailed analysis of temporal evolution of performance is presented in Section 5.8.

In addition to these metrics based on the idleness, we also measure the number of *interferences* over time. An interference is defined as a situation when two robots get closer so that the obstacle avoidance module is used to avoid each other. In the simulated experiments, the interferences are detected when two robots are within 2 meters and they are counted by an external module accessing the ground truth positions of the robots. The interferences are important, since they affect the time to reach a goal position and thus all the performance metrics based on idleness. Moreover, they cannot just be disregarded, since minimizing conflicts on space is an important feature of a multi-robot patrolling system. In the experiments reported in this article, we use the rate of interferences (If_{rate}), expressed in number of interferences per minute, as an additional performance metric of an MRP system. The results for If_{rate} obtained in the experiments reported in this section are illustrated in the last column of Table 3. As expected, SEBS and DTAP algorithms present much less interferences with respect to the RAND algorithm.

From this analysis, we can conclude that using only the average value I^G_{avg} would not allow for a suitable comparison of different methods. The use of boxplots and temporal plots provides for a detailed analysis of the performance of the algorithm. However, in these cases, it is difficult to summarize the result of an experiment in a single value and thus it is not always possible to definitely compare two sets of results. The use of the maximum value I^G_{max} is a good choice as worst case analysis, but from the experiments done during this work we have experienced a high variability of this value (since it depends on noise and uncertainty in the execution of the task by the robots, which is not modeled in the MRP problem). Finally, the rate of interferences If_{rate} is important to assess the ability of the algorithm to prevent space conflicts.

In the following, we mainly present the results of experiments with boxplots of idleness values. In some cases, we also show temporal plots, in order to evaluate temporal evolution of the idleness, and tables with the aggregate results based on the idleness and the rate of interferences.

5.4 Length and repetitions of the experiments

Given the external factors mentioned in the previous section, it is important to determine how many experiments in the same setting are needed in order to guarantee significant results and how long an experiment must last. To this end, we made the experiments reported below (named **Experiment Set 2**).

Because of the above mentioned external factors that affect the results of a MRP test, performing a single run of an experiment in a given setting is not enough to guarantee significant results. The procedure we have developed in this article is based on performing a statistical test (in particular, the one-way ANOVA test) to the obtained results in order to determine the ones that are significant. More specifically, we perform many runs of each configuration until we are able to collect at least 3 tests that have significant correlation. Significant correlation is measured with the ANOVA test applied to all the values of the idlenesses collected during an experiment. When the

ANOVA test returns a high p value (e.g., > 0.1) then we can conclude that there is high probability that the two sets of values (i.e., idlenesses of the two experiments) come from the same distribution.

Therefore, by collecting three experimental runs for which the ANOVA test returns mutual correlation, we can guarantee statistical significance of the experimental results. On the other hand, experimental runs for which the ANOVA test return a small p value (i.e., low correlation) are discarded, since in these cases it is likely that some of the external factors have determined a non-significant result.

Another important control parameter for the experiments is length. In order to determine this parameter, we have conducted some experiments and applied the statistical test described above.

A first experiment has been done to measure the evolution over time of the two metrics: global idleness average (I^G_{avg}) and global idleness standard deviation (I^G_{stddev}).

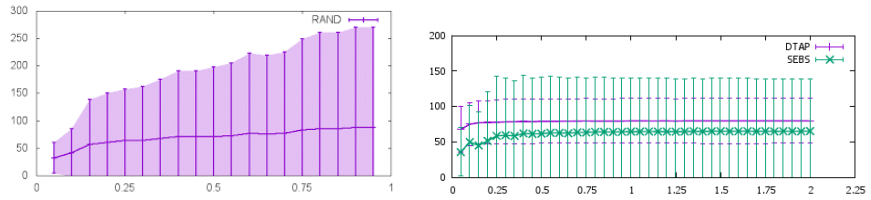

Fig. 5 Performance metrics ((I^G_{avg} and I^G_{stddev}) over time (hours): a) RAND, b) SEBS and DTAP.

In Figure 5 the evolution over time of the algorithms RAND (left side), SEBS and DTAP (right side) is reported. On the left plot, it is clear that average and standard deviation of the idleness when using the RAND algorithm increases over time, while the right plot shows that SEBS and DTAP get stable results between 0.5 and 1 hour.

This test highlights that stability of the results over time depends also on the algorithm. However, most of the algorithms, and in particular the ones mostly addressed in this article, have stable results within 1 hour.

In order to further confirm this result, we have executed another experiment in which we applied the statistical analysis described above. In this second experiment, we have collected three statistically correlated runs for each of the following lengths of the experiments: 30 minutes, 1 hour, and 2 hours.

Statistical analysis has been performed by computing the p values for each pair of such experiments and the results are reported in Table 4. The table shows that there is high correlation between all the results obtained. Consequently, all the experiments describer in the next sections have been performed for 1 hour, which, as discussed before, guarantees significant results.

5.5 Comparison among all the algorithms

Comparison among all the algorithms has been the subject of the next set of experiments (named **Experiment Set 3**). A limited set of scenarios have been considered

	0.5 h	0.5 h	0.5 h	1 h	1 h	1 h	2 h	2 h	2 h
0.5 h		0.848	0.731	0.915	0.783	0.959	0.649	0.807	0.578
0.5 h			0.571	0.900	0.610	0.774	0.475	0.620	0.425
0.5 h				0.592	0.906	0.715	0.990	0.845	0.880
1 h					0.624	0.839	0.444	0.631	0.369
1 h						0.772	0.852	0.933	0.723
1 h							0.596	0.802	0.497
2 h								0.739	0.820
2 h									0.587
2 h									

Table 4 Correlation of experiments with SEBS algorithm in grib map with 4 robots.

in this set, since it was clear that two algorithms outperform the others and thus we decided to provide a more detailed analysis of these two algorithms only in a next set of experiments (reported in next section).

The six scenarios considered in this Experiment Set are the following ones, indicated as a pair (map, number of robots): (grid,4), (example,4), (cumberland,4), (cumberland,6), (DIAG_floor1,6), (broughton, 8). In all these experiments, we used the predefined scattered initial poses of the robots. As for the duration of the experiments, we set the maximum time to 60 minutes. However, some runs were interrupted before this time out, when a robot did not reach a target after 5 minutes.

The results obtained are shown as boxplots in Figure 6 and in Tables 5–10. According to the discussion about performance metrics in Section 5.3, performance assessment can be done by considering low standard deviation (i.e., small size of the box), small number and lower values of outliers (i.e., less and lower "high" values of the idleness), and small number of interferences. From this analysis, it appears evident (although not associated to a single result value) that SEBS and DTAP generally outperform all the other algorithms. These results also confirm the trend reported in previous experiments on MRP (e.g., [23,22]).

5.6 Detailed comparison between SEBS and DTAP

In the next phase of the experiments (named **Experiment Set 4**), we focused the analysis only on the two most effective algorithms, SEBS and DTAP, in order to make a more detailed comparison in more scenarios and conditions. More specifically, we consider scenarios with a variable ratio between the size of the patrol graph and the number of robots. The results reported in this section generally show that DTAP is more effective when this ratio is high, that is large graph with a few robots, while when this ratio gets lower DTAP and SEBS tend to have similar performance or SEBS seems to be more effective. Therefore, the most appropriate algorithm could be chosen according to the specific application, although the case in which a few robots have to patrol a large space is more realistic.

More specifically, three runs for each algorithm and for each scenario are shown as boxplots in Figures 7 and 8. In Figure 7, six scenarios are considered with a ratio

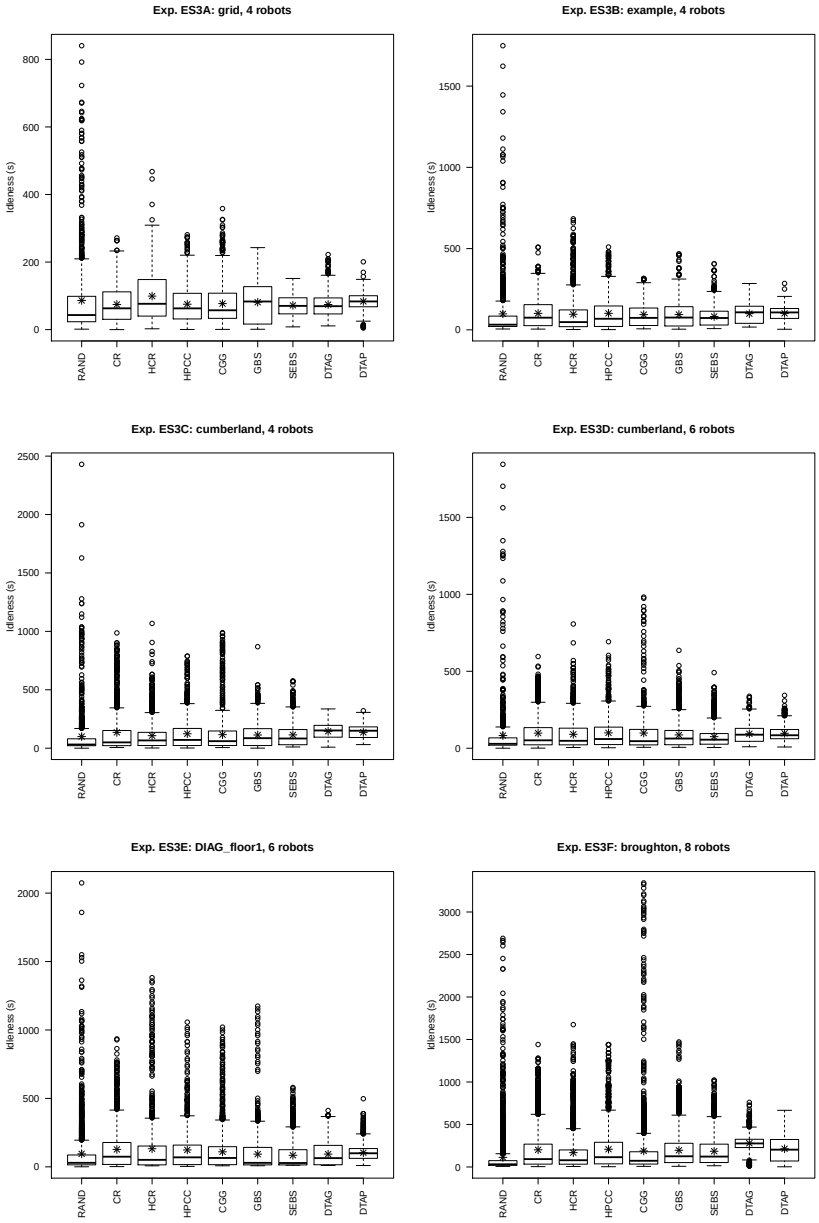

Fig. 6 Comparison among all the algorithms in six scenarios.

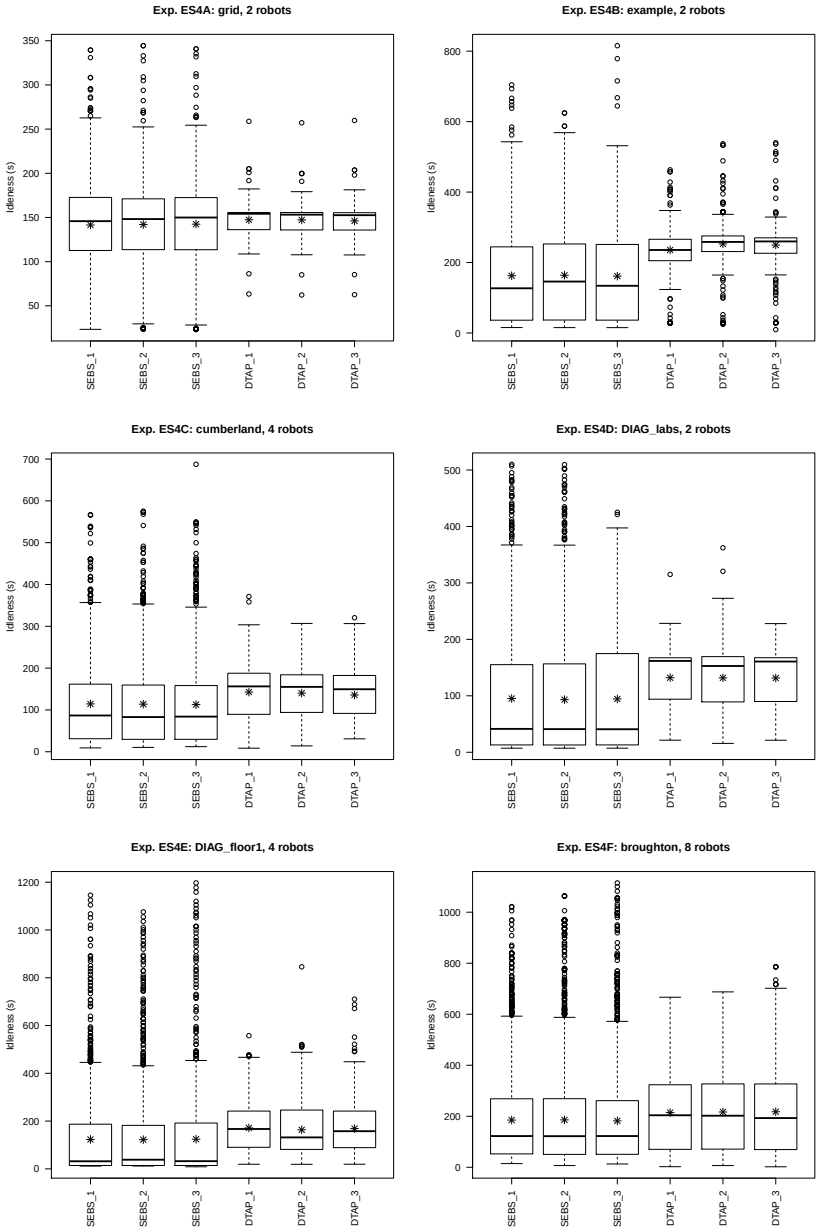

Fig. 7 Comparison among all SEBS and DTAP algorithms in six scenarios with ratio size of the graph / number of robots ≥ 10.

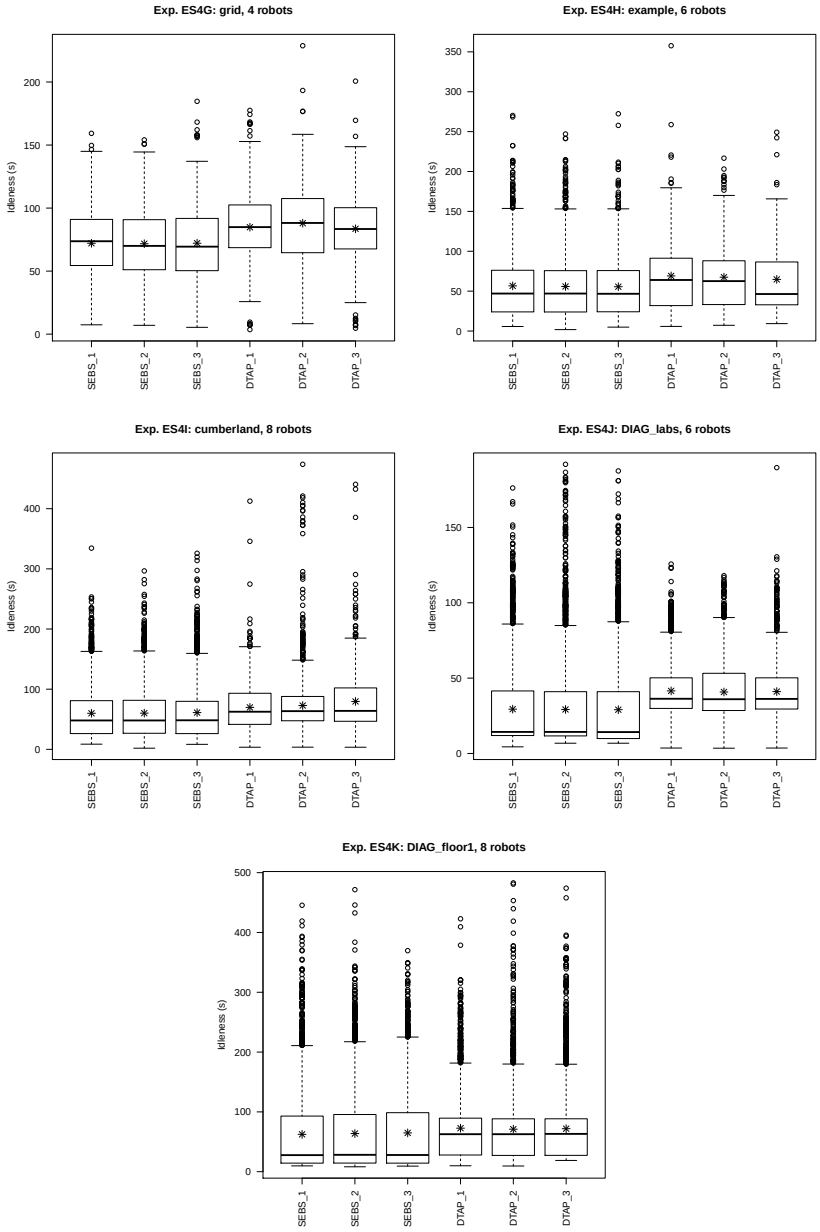

Fig. 8 Comparison among all SEBS and DTAP algorithms in five scenarios with ratio size of the graph / number of robots < 8.

Alg.	I^G_{avg}	I^G_{stddev}	I^G_{max}	If_{rate}
RAND	86.3	118.0	840.5	4.308
CR	74.4	51.8	271.4	3.164
HCR	99.2	75.5	467.9	4.396
HPCC	75.2	54.7	280.8	3.213
CGG	77.1	58.7	358.3	3.808
GBS	80.9	56.7	242.7	5.092
SEBS	**71.7**	29.0	**151.4**	**0.033**
DTAG	74.3	36.0	222.2	0.383
DTAP	83.5	**23.2**	200.7	0.466

Table 5 Exp. ES3A: grid, 4 robots

Alg.	I^G_{avg}	I^G_{stddev}	I^G_{max}	If_{rate}
RAND	98.1	183.1	1748.5	5.645
CR	101.2	88.1	509.8	5.358
HCR	95.8	117.7	683.4	5.462
HPCC	101.5	101.9	509.8	5.108
CGG	93.6	78.1	316.3	1.498
GBS	93.9	80.1	467.4	2.995
SEBS	**81.9**	63.8	406.7	**0.067**
DTAG	99.5	58.2	**285.2**	0.582
DTAP	101.6	**43.3**	285.7	0.133

Table 6 Exp. ES3B: example, 4 robots

Alg.	I^G_{avg}	I^G_{stddev}	I^G_{max}	If_{rate}
RAND	**99.3**	199.8	2429.9	2.663
CR	135.5	191.8	986.8	5.662
HCR	107.2	125.2	1067.8	2.946
HPCC	123.3	139.1	788.0	4.762
CGG	115.9	154.3	989.5	2.165
GBS	112.7	106.1	868.7	3.096
SEBS	113.8	102.8	575.2	0.216
DTAG	144.3	72.3	336.1	0.483
DTAP	135.7	**54.2**	**320.5**	**0.117**

Table 7 Exp. ES3C: cumberland, 4 robots

Alg.	I^G_{avg}	I^G_{stddev}	I^G_{max}	If_{rate}
RAND	81.3	181.5	1844.9	11.172
CR	98.3	106.9	596.0	12.585
HCR	90.6	97.3	807.4	10.170
HPCC	99.4	109.0	692.1	12.134
CGG	98.4	143.3	982.3	12.787
GBS	86.4	86.7	636.0	7.523
SEBS	**75.2**	64.7	491.4	0.433
DTAG	93.9	58.6	337.6	2.981
DTAP	97.0	**47.1**	**343.9**	**0.033**

Table 8 Exp. ES3D: cumberland, 6 robots

Alg.	I^G_{avg}	I^G_{stddev}	I^G_{max}	If_{rate}
RAND	95.0	182.9	2074.6	8.291
CR	127.4	149.0	934.9	11.849
HCR	133.5	213.2	1382.3	14.447
HPCC	123.9	162.0	1057.4	9.673
CGG	109.8	137.0	1020.7	7.958
GBS	93.0	124.2	1174.2	4.314
SEBS	**84.5**	97.9	579.7	0.083
DTAG	93.2	84.6	**411.6**	1.199
DTAP	103.4	**57.7**	497.7	**0.033**

Table 9 Exp. ES3E: DIAG_floor1, 6 robots

Alg.	I^G_{avg}	I^G_{stddev}	I^G_{max}	If_{rate}
RAND	**112.9**	259.9	2690.2	6.063
CR	201.9	247.0	1440.9	8.959
HCR	168.6	221.2	1675.2	8.844
HPCC	208.8	242.4	1440.4	6.113
CGG	188.6	358.0	3342.4	2.996
GBS	196.4	201.3	1471.4	2.048
SEBS	185.1	174.8	1022.1	**0.000**
DTAG	279.8	**80.9**	760.2	2.781
DTAP	214.2	152.0	**666.6**	2.496

Table 10 Exp. ES3F: broughton, 8 robots

between size of the graph and number of robots ≥ 10. In these situations, DTAP provides better performance than SEBS, since the benefit of a stronger coordination is advantageous given the large number of coordination possibilities. While in Figure 8, the results in five scenarios with a lower ratio between size of the graph and number of robots (< 8) show situations in which a strong coordination is less beneficial. In these settings, the performance are similar or, in some cases, SEBS has better performance.

These experiments show the need of a more sophisticated form of coordination when there are many possible choices for the team, corresponding to high number of tasks with respect to the available robots.

5.7 Impact of different initial positions of the robots

In these experiments (named **Experiment Set 5**), we have evaluated the impact of the initial poses of the robots in the environment in the overall performance. We have

considered three scenarios and additional initial poses with respect to the "default" ones used in previous experiments. While in all the default initial poses the robots are scattered through the environment far away each other, in the new sets of initial poses considered here the robots starts close each other in a limited area. The new sets of initial poses are indicated below with a letter ('a', 'b', 'c'), while the letter is missing when the default poses are used.

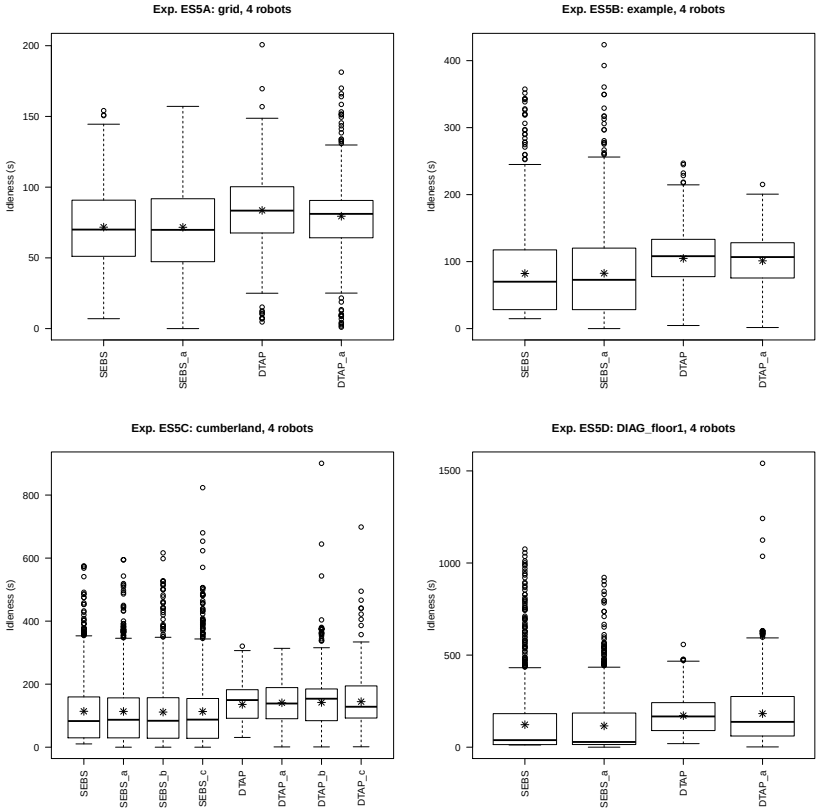

Fig. 9 Comparison among all SEBS and DTAP algorithms in three scenarios with different intial poses.

Figure 9 shows the results obtained in the four scenarios for SEBS and DTAP algorithms. In these figures, labels 'SEBS' and 'DTAP' refers to default initial poses, while SEBS_α and DTAP_α refer to the new sets of initial poses when the robots are grouped in a limited area. As shown in the figure, we have defined one additional set of initial poses ('a') for grid, example, and DIAG_floor1 maps and three additional sets of initial poses ('a', 'b' , 'c') for cumberland map.

The results confirm that in general the initial poses of the robots do not substantially affect the overall performance in the long term. The runs 'DTAP_b' and 'DTAP_c' in Experiment ES5C and DTAP_a in Experiment ES5D, however, show

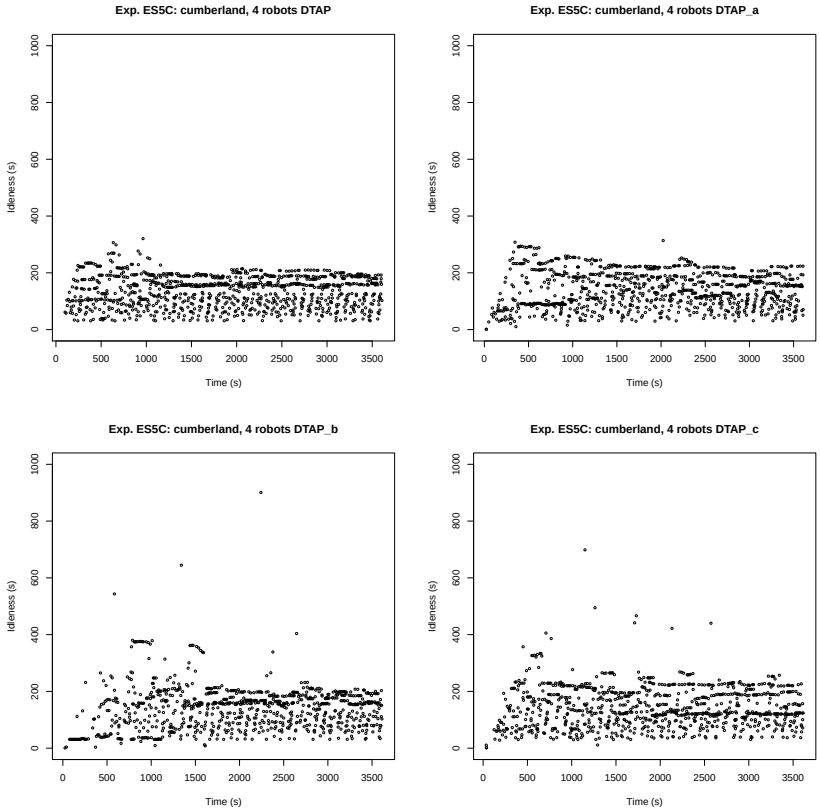

Fig. 10 Time plots of runs with different initial poses.

higher values of the idleness. By looking at the timeplots for some of these runs (illustrated in Figure 10), a similar trend of the values of the idleness can be observed with the only exception of a few outliers, due to particular situations in which task allocation was not optimal.

We can conclude that in most cases the initial poses of the robots do not significantly affect the overall performance, while DTAP seems to produce a few isolated outliers.

5.8 Temporal analysis and variable number of robots

Another important evaluation is related to analyze temporal evaluation of the performance metrics. This is also related to situations in which the number of robots in the team changes over time, for example in a long-term situation in which robots have to temporary leave the team for battery recharge operations.

In these experiments (named **Experiment Set 6**), we have analyzed how the performance metrics vary over time, to assess the ability of an algorithm to adapt to changing conditions over time also in presence of a variable number of robots.

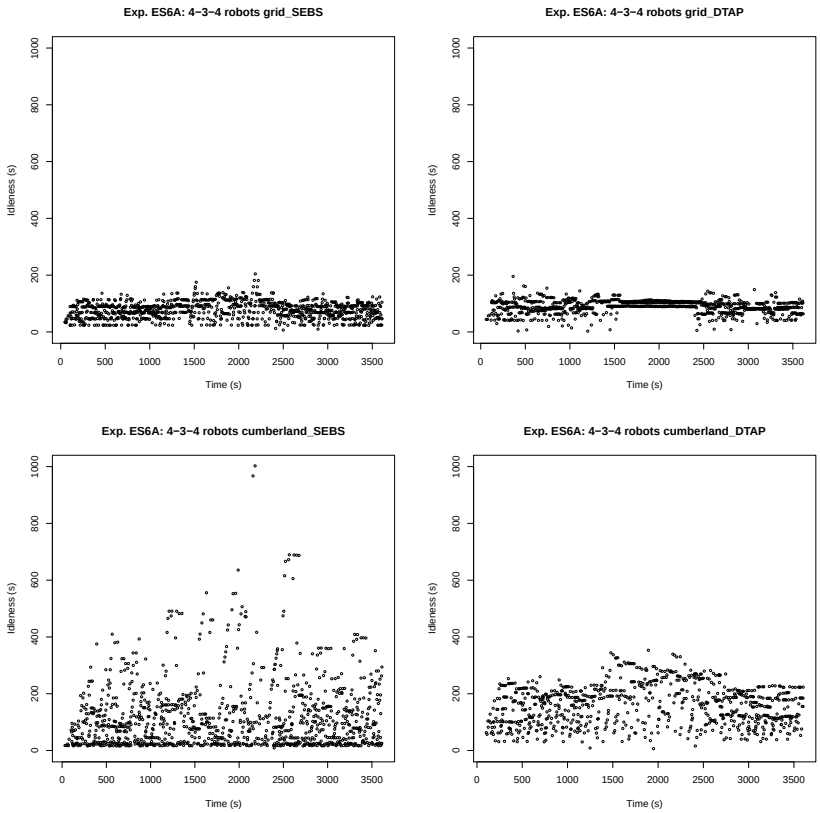

Fig. 11 Time plots of runs with different number of robots (4,3,4).

Figure 11 shows the timeplots of four runs of Experiment ES6A in which the two algorithms SEBS and DTAP have been compared in two scenarios: grid and cumberland. In these runs, 4 robots are used at the beginning of the task, then after 20 minutes one robot is removed from the team, finally after additional 20 minutes the robot is put again in the team. The timeplots in the Figure show the evolution of the idleness values. It is interesting to notice that in the grid map, when three robots are in the team (from time 1200 to time 2400), SEBS presents just a slight increase of idleness values, while DTAP finds an even better partition of the nodes among the three robots, as demonstrated by the almost continuous flat lines in the plot. After time 2400 when the four robots are again operating, it is possible to observe similar performance with respect to the first time period (0-1200), so showing adaptability of the algorithms to the number of robots. In the cumberland map, the difference

between the period with three robots and the periods with four robots is more evident. Also in this case, both the algorithms adapt to this changing condition and DTAP has general better performance, as confirmed in previous experiments.

It is interesting to observe that, even though the algorithms developed so far show some degree of adaptability to changing situations, it is not yet the case that they show "learning abilities". We believe that introducing learning in MRP can bring to a novel family of algorithms that can further exploit adaptiveness to changing situations.

5.9 Porting to real robots

Fig. 12 Real robot scenario: DIAG_labs map, 3 robots and 8 patrol nodes.

The developed algorithms have been also validated in a real environment with three robots. Figure 12 shows the real environment of DIAG_labs map, which is the corridor of our lab. A portion of this map, considered in the simulated scenarios, has been used to test the execution of the developed algorithms with real robots. In this environment, a patrol graph with 8 nodes and a team of 3 robots have been considered. The tests with real robots have been performed by using the same implementation of the algorithms described before. ROS infrastructure indeed allows for an easy porting from a Stage-based simulated application to a real one. In particular, Turtlebots in the real environment and robots in the Stage simulator use the same map of the environment, the same configuration of parameters for localization and navigation, and the same implementations of the MRP algorithms.

The experiments performed with the real robots have the objective of demonstrating the portability of the software in a real environment. Experimental runs with variable initial poses and variable number of robots during the task have been performed, confirming the trends in the simulated experiments. Details on these experiments and videos are available in this web site: `https://sites.google.com/a/dis.uniroma1.it/mrp-dta/`.

6 Conclusions

This article proposes an on-line coordination approach to multi-robot patrolling based on dynamic task assignment. Specifically, we propose two dynamic task assignment techniques: a baseline greedy approach (DTA-Greedy) and a market based technique based on sequential single item auctions (DTAP). We evaluate the performance of such approach in a realistic simulation environment (built with ROS and stage) as well as on real robotic platforms.

In particular, in the simulated environments we compare our task assignment approaches with previous off-line and on-line methods. Our results confirm that on-line coordination approaches improve the performance of the multi-robot patrolling system in real environments, by coping with the unpredictable and inevitable dynamic elements that are due to noise in perception and uncertain action execution, typical of robotic systems. Moreover, our results show that coordination approaches that employ more informed coordination protocols (i.e., DTAP) achieve better performance with respect to state-of-the-art on-line approaches (i.e., SEBS). This is particularly true, in scenarios where there are many possible tasks (i.e., large environment) with respect to the number of robots.

The experimental evaluation reported in this article has analyzed the different performance metrics used in previous work and some forms of presentation of the results. This analysis showed that a more detailed analysis of these metrics is needed in order to assess MRP performance. The algorithms and the simulator used for the experiments is fully available, the results reported in this article are fully reproducible and this simulator can be thus further extended to include other algorithms and to improve performance evaluation and benchmarking of MRP systems.

Finally, the experiments performed with real robots (a team of three Turtlebot platforms in an office environment) show that, when deployed on a real system, our approach is able to successfully coordinate the robots achieving good patrolling behaviors in face of uncertainty and noise (e.g., localization and navigation error) associated to real platforms.

In conclusion, on-line distributed task assignment techniques allow to cope with the dynamics arising from the robot behaviors, hence increasing the performance of the patrolling system.

Among the several future research directions in Multi-Robot Patrolling, we believe that an interesting direction refers to extending on-line coordination algorithms for MRP in settings where strategic reasoning comes into play and where robots must jointly act to perform more complex surveillance operations in each location of the environment. For example, heterogeneous teams of robots with different abilities are used to monitor an environment in which each node must be visited simultaneously by more robots with different abilities.

References

1. Agmon, N.: On events in multi-robot patrol in adversarial environments. In: AAMAS (2010)

2. Agmon, N., Fok, C.L., Emaliah, Y., Stone, P., Julien, C., Vishwanath, S.: On coordination in practical multi-robot patrol. In: IEEE International Conference on Robotics and Automation (ICRA) (2012). URL http://www.cs.utexas.edu/users/ai-lab/?ICRA12-agmon
3. Agmon, N., Urieli, D., Stone, P.: Multiagent patrol generalized to complex environmental conditions. In: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (2011). URL http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=127070
4. Ahmadi, M., Stone, P.: A multi-robot system for continuous area sweeping tasks. In: Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1724–1729 (2006).
5. Almeida, A., Ramalho, G., Santana, H., Tedesco, P.A., Menezes, T., Corruble, V., Chevaleyre, Y.: Recent advances on multi-agent patrolling. In: SBIA (2004)
6. Basilico, N., Gatti, N., Villa, F.: Asynchronous multi-robot patrolling against intrusion in arbitrary topologies. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI), pp. 1124–1229. Atlanta, USA (2010)
7. Batalin, M.A., Sukhatme, G.S.: Distributed Autonomous Robotic Systems 5, chap. Spreading Out: A Local Approach to Multi-robot Coverage, pp. 373–382. Springer Japan (2002)
8. Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Transactions on Robotics **21**(3), 376–386 (2005)
9. Chevaleyre, Y.: Theoretical analysis of the multi-agent patrolling problem. Intelligent Agent Technology, IEEE / WIC / ACM International Conference on (2004)
10. Elmaliach, Y., Agmon, N., Kaminka, G.: Multi-robot area patrol under frequency constraint. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) (2007)
11. Elmaliach, Y., Agmon, N., Kaminka, G.: Multi-robot area patrol under frequency constraints. Annals of Mathematics and Artificial Intelligence (2009)
12. Elmaliach, Y., Shiloni, A., Kaminka, G.A.: A realistic model of frequency-based multi-robot polyline patrolling. In: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems - Volume 1 (2008)
13. Holz, D., Basilico, N., Amigoni, F., Behnke, S.: A comparative evaluation of exploration strategies and heuristics to improve them. In: Proceedings of the Fourth European Conference on Mobile Robots (ECMR 2011), pp. 25–30 (2011)
14. Iocchi, L., Marchetti, L., Nardi, D.: Multi-robot patrolling with coordinated behaviours in realistic environments. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS), pp. 2796–2801 (2011). DOI http://dx.doi.org/10.1109/IROS.2011.6094844
15. Machado, A., Ramalho, G., Zucker, J.D., Drogoul, A.: Multi-agent patrolling: an empirical analysis of alternative architectures. In: Proceedings of the 3rd international conference on Multi-agent-based simulation II. Berlin, Heidelberg (2003)
16. Marier, J.S., Besse, C., Chaib-draa, B.: Solving the continuous time multiagent patrol problem. In: Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 941–946 (2010)
17. Marino, A., Parker, L., Antonelli, G., Caccavale, F.: Behavioral control for multi-robot perimeter patrol: a finite state automata approach. In: Proceedings of the 2009 IEEE international conference on Robotics and Automation (2009)
18. Pippin, C., Christensen, H., Weiss, L.: Performance based task assignment in multi-robot patrolling. In: Proceedings of the 28th Annual ACM Symposium on Applied Computing, SAC '13, pp. 70–76. ACM, New York, NY, USA (2013). DOI 10.1145/2480362.2480378. URL http://doi.acm.org/10.1145/2480362.2480378
19. Portugal, D., Couceiro, M.S., Rocha, R.P.: Applying bayesian learning to multi-robot patrol. In: Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on, pp. 1–6 (2013). DOI 10.1109/SSRR.2013.6719325
20. Portugal, D., Rocha, R.: Msp algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In: Proceedings of the 2010 ACM Symposium on Applied Computing (2010)
21. Portugal, D., Rocha, R.P.: On the performance and scalability of multi-robot patrolling algorithms. In: Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on, pp. 50 –55 (2011). DOI 10.1109/SSRR.2011.6106761
22. Portugal, D., Rocha, R.P.: Distributed multi-robot patrol: a scalable and fault-tolerant framework. Robotics and Autonomous Systems **61**, 1572–1587 (2013)
23. Portugal, D., Rocha, R.P.: Multi-robot patrolling algorithms: examining performance and scalability. Advanced Robotics **27**(5), 325–336 (2013)

24. Santana, H., Ramalho, G., Corruble, V., Ratitch, B.: Multi-agent patrolling with reinforcement learning. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 3, AAMAS '04, pp. 1122–1129 (2004)
25. Semp, F., Drogoul, A.: Adaptive patrol for a group of robots. In: IROS, pp. 2865–2869. IEEE (2003)
26. Stranders, R., de Cote, E.M., Rogers, A., Jennings, N.: Near-optimal continuous patrolling with teams of mobile information gathering agents. Artificial Intelligence **195**(0), 63 – 105 (2013). DOI http://dx.doi.org/10.1016/j.artint.2012.10.006. URL `http://www.sciencedirect.com/science/article/pii/S0004370212001282`
27. Tovey, C., Lagoudakis, M., Jain, S., Koenig, S.: The generation of bidding rules for auction-based robot coordination. In: F.S. L. Parker, A.S. (editor) (eds.) Multi-Robot Systems: From Swarms to Intelligent Automata, vol. 3. Springer (2005)