Advanced Robotics, 2013
Vol. 27, No. 5, 325–336, http://dx.doi.org/10.1080/01691864.2013.763722

FULL PAPER

Multi-robot patrolling algorithms: examining performance and scalability

David Portugal[a]* and Rui P. Rocha

Institute of Systems and Robotics, University of Coimbra, 3030-290 Coimbra, Portugal

(*Received 28 March 2012; accepted 6 May 2012*)

In this paper the problem of patrolling an environment with a dynamic team of robots is targeted. Lately, the interest of the research community has been focused in the development of patrol strategies; however there is a deficit of studies comparing such strategies, namely in terms of their performance and team scalability in different environments. For this reason, an evaluation of five representative patrol approaches is presented in this article. Aiming to analyze the performance, ability to scale and the behavior resulting from interactions between teammates, extensive realistic simulation using ROS together with Stage was conducted. The metric used to compare the performance is the average idleness of the topological environment (i.e. graph), that represents the area to patrol. The results presented help to identify which strategies enable enhanced team scalability and which are the most suitable approaches given any environment, supporting future research directions in the field.

Keywords: multi-robot systems; patrolling; security; scalability and performance

1. Introduction

In this work, the focus is laid on surveillance tasks using multiple robots, which involve frequent visits to important places of the environment. Consequently, the word 'Patrolling' is implicitly used in this sense.

The major motivation for studying this issue relates to its spectrum of applicability in the context of security systems and the potential to replace or assist human operators in dangerous real-life scenarios, like mine clearing, rescue operations or surveillance, easing arduous and time-consuming tasks and offering the possibility to relieve human beings, enabling them to be occupied in nobler tasks like monitoring the system from a safe location.

Cooperation among robots is one of the most decisive issues in this context; since robots must efficiently work together in order to improve the performance of the system as a whole. In addition, multi-robot patrol is a challenging problem, because agents must navigate autonomously, coordinate their actions, be distributed in space and must be independent of the number of robots and the environment's dimension.

This work presents a comparative study of five different state-of-the-art patrolling strategies using distinct topological environments and different team sizes, in order to analyze the performance and scalability of each approach. The results herein presented extend previous work [1] by adding an analysis of variance (ANOVA) and an analysis of the algorithms' convergence time. Conclusions drawn in this field of research may support the development of future approaches not only in this domain but also in other multi-robot applications.

The rest of this document is organized as follows. A brief survey of the work previously done in the area is presented in the next section. The problem is defined in the following section and section 4 presents the patrolling strategies evaluated and compared in this work. The subsequent section presents the motivation for the maps and simulator used. In section 6, experimental results are presented and discussed. Finally, the article ends with conclusions and future work directions.

2. Related work

The existing algorithms in the literature for patrolling an environment with multiple mobile agents present many differences in terms of strategy, communication paradigm, cooperation scheme, performance evaluation, and other features. They can be divided into pioneer methods;[2,3] graph theory methods;[4–6] and alternative coordination methods. [7–9]

Pioneer strategies include simple architectures with agents with different capabilities that move in the environment mostly looking for locations that have not been visited for some time, aiming to maintain a high frequency of visits in every place of the area.

*Corresponding author. Email: davidbsp@isr.uc.pt

Graph theory strategies look for solutions of classical problems like finding Hamilton cycles, graph partitioning and others to assign efficient routes for the robot's patrolling missions. These strategies typically rely on a centralized coordinator to calculate those routes.

Recently, many alternative coordination methods have also been presented, aiming to solve the problem through the usage of approaches that have presented good results in multi-robot systems in general, like task allocation, reinforcement learning, negotiation mechanisms, and swarm-based strategies.

A pioneer work was presented in [2], where several architectures for multi-agent patrol were proposed. These architectures have distinct agents' behavior, perception, communication paradigms, and decision-making. Additionally, they have contributed with criteria to evaluate the performance of the approaches based on the average and maximum idleness of the vertices of the graph that represents the topological environment.

In [3], the architectures proposed by Machado were enhanced with advanced decision-making, based on both the instantaneous idleness of vertices and the distance to them, as well as with advanced path finding, which considers distance (or cost) of the edges and the idleness of vertices in the path towards a goal. Also, the tests were run on more and distinct environment topologies, which was a strong limitation of Machado's work. Nonetheless, Almeida's work contains several simplifications that are overcome in this article by using realistic simulations, as in [10], that consider both the dynamic of robots and the actual time to measure performance instead of using iterative simulation cycles.

Approaches based on graph theory and operational research (O.R.) commonly address the patrolling problem by computing minimal-cost cycles that visit all points in the target area. The agents are employed uniformly along the path and follow the same patrol route over and over again. For example, [4] presented an area patrol algorithm based on the computation of Hamilton cycles that guarantees that each point in the target area is covered at the same optimal frequency. On the other hand, [5] described a cyclic algorithm based on heuristics that approximates a travelling salesman problem (TSP) cycle on top of the topological representation of the environment.

These strategies are robust, being independent of the number of robots and are recognized by the good results that they achieve in terms of visit frequency. However, they have a deterministic nature, which means that an intelligent intruder that apprehends the patrolling scheme may take advantage of the idle time between passages of robots in some points of the area.

In [6], the multilevel subgraph patrolling (MSP) Algorithm is described. In this approach, balanced graph partitioning is projected in order to assign different patrolling regions (subgraphs) of the environment for each mobile agent. The algorithm subsequently computes effective paths for every robot using classical graph theory approaches. Results confirmed the flexible and high-performance nature of the approach, which benefits from being non-redundant and not needing inter-agent communication.

Some alternative methods have been presented throughout the years. In [7], patrolling is addressed in a task allocation perspective, where each robot is assigned a different region to visit. Robots send their current state to a centralized system running on a remote computer, through wireless communication, to compute the task strength and drive the robot through propagated data.

Moreover, reinforcement learning was used in [8] to solve the patrolling problem by automatically adapting the agents' strategies to the topology of the environment. Additionally, an approach based on a negotiation mechanism has also been proposed in [9]. In this work, agents exchange vertices of the environment graph to patrol, using an auction-based system. The agents will naturally aim to obtain a set of vertices in the same region of the graph. Despite the results obtained, reinforcement learning and negotiation mechanisms prove to be extremely complex when compared to pioneer strategies, with nearly no communication ability, which achieve similar results.[11]

In the literature, other alternative solutions based on Markov decision processes,[12] game theory,[13] artificial forces,[14] neural networks [15] and even swarm robotics [16] can be found. For a more thorough survey of multi-robot patrolling architectures, one should refer to [17].

3. Problem formulation

As mentioned before, it is common to represent the area to patrol by an undirected connected graph G = (V, E) with $v_i \in$ V vertices and $e_{i,j} \in$ E edges. Therefore, G corresponds to the topological map for the patrolling mission and is assumed to be known a priori. Also, in these maps, vertices correspond to important places or landmarks, connected by edges that represent the paths between them.

In order to address and compare the performance of different patrolling algorithms, it is important to establish an evaluation metric.

The instantaneous idleness (Idl_{t_k}) of a vertex $v_i \in$ V in time t_k, with t = {0, ..., t_k}, is defined as:

$$Idl_{t_k}(v_i) = t_k - t_{last\_visit}, \tag{1}$$

where t_{last_visit} corresponds to the last time instant when the vertex v_i was visited by any robot of the team. Consequently, the average idleness (Idl_m) of a vertex $v_i \in$ V in a total time T can be defined as:

$$Idl_m(v_i) = \frac{\sum_{k=0}^{T} Idl_{t_k}(v_i)}{T} \tag{2}$$

Figure 1. Properties of the two planning approaches to solve the patrolling problem with teams of multiple mobile robots.

Finally, in order to obtain a generalized measure, the average idleness of the graph G (Idl_G) is defined as:

$$Idl_G = \frac{\sum_{i=0}^{|V|} Idl_m(v_i)}{|V|}, \tag{3}$$

where $|V|$ represents the cardinality of the set V.

Considering a patrolling path as an ordered array of vertices of G, the multi-robot patrolling problem can thus be described as the problem of finding a set of paths **x** which visit all vertices $v_i \in$ V of the graph G, using an arbitrary team of R robots, with the overall team goal of minimizing Idl_G:

$$f = \underset{\mathbf{x}}{\operatorname{argmin}}\, Idl_G \tag{4}$$

By finding:

$$\mathbf{x} = [x_1, ..., x_r, ..., x_R] \tag{5}$$

Such that:

$$x_r = \{v_{r,1}, v_{r,2}, ..., v_{r,N}\} \tag{6}$$

$$v_{r,n} \in V$$

$$1 \leq r \leq R, R \in \mathbb{N}$$

$$1 \leq n \leq N, N \in \mathbb{N}$$

Subject to:

$$\forall v_i \in V, \exists x_r \in \mathbf{x} : v_i \in x_r \tag{7}$$

Note that x_r represents the patrolling path of robot r that can either be calculated a *priori*, which is typically done by centralized algorithms, or online to consider and incorporate the dynamics of the system in a given time step, which is usual in distributed approaches, as shown in Figure 1.

4. Evaluated patrolling algorithms

Having analyzed the literature, five representative algorithms were implemented. These algorithms were chosen from among all previous research works based on the good performance results that they have obtained and the different properties assumed like agent perception, decision-making, and planning, as it is shown in Table 1. In this section, those algorithms are examined in detail. No alternative approaches were implemented in this work mainly due to the fact that the complexity of their implementation does not lead to better results when compared to simpler approaches, as previously concluded by Almeida *et al.* [11].

Besides the analysis of the performance of the diverse algorithms, the scalability of the approaches studied is also addressed in this work. In the context of multi-robot systems, scalability is related to how well a given strategy performs as the dimension of the team grows [18] and how the individual productivity of each robot is influenced by the increase of several number of agents in the team. Having this in mind, the interference between robots is measured in every experiment as the number of times that different agents share nearby areas, having to avoid each other.

4.1. *Conscientious reactive*

Ranked one of the top algorithms in the study of Machado *et al*. [2], Conscientious Reactive (CR) is a simple pioneer approach, in which agents decide locally which vertex they should move to in the next step, taking only into consideration the instantaneous idleness of the open neighborhood of the current vertex, $N_G(v)$[1], where the robot is located at the moment. Algorithm 1 presents the pseudo-code of the approach.

Algorithm 1: Conscientious Reactive

```
1  r ← load(robot_id);
2  G ← load(topological_map);
3  v_{r,n} ← load(initial_vertex);
4  add(v_{r,n} to x_r);
5  init(Idl_{t_k}[V]);
6  while true do
7      v_{r,n+1} ← argmax(Idl_{t_k}[N_G(v_{r,n})]);  /* Next vertex
       is the neighbor of the current vertex with
       highest idleness. */
8      move_robot(v_{r,n+1});
9      v_{r,n} ← v_{r,n+1};
10     add(v_{r,n} to x_r);
11     update(Idl_{t_k}[V]);
```

4.2. *Heuristic Conscientious Reactive*

Heuristic Conscientious Reactive (HCR) is an algorithm presented by Almeida in [3]. It is similar to CR with an important modification on the decision-making process, where the authors calculate a decision value that considers not only the instantaneous idleness of the neighbors of the current vertex as well as the distance to them. Algorithm 2 presents the pseudo-code of the approach.

Table 1. Comparative table of the analyzed architectures.

Algorithm	Complexity	Perception	Decision-making	Planning
CR	*	Reactive	Local idleness-based	Online
HCR	**	Reactive	Local heuristic-based	Online
HPCC	***	Cognitive	Global heuristic-based	Online
CGG	****	Cognitive	Cycle computation	Offline
MSP	****	Cognitive	O.R. inside each region	Offline

Algorithm 2: Heuristic Conscientious Reactive

```
1  r ← load(robot_id);
2  G ← load(topological_map);
3  v_{r,n} ← load(initial_vertex);
4  add(v_{r,n} to x_r);
5  init(Idl_{t_k}[V]);
6  while true do
7      HIdl ← max(Idl_{t_k}[N_G(v_{r,n})]);
8      MaxDist ← max(e_cost(v_{r,n}, N_G(v_{r,n})));
9      forall the v_i ∈ N_G(v_{r,n}) do
10         NormIdl[v_i] ← Idl_{t_k}[v_i] / HIdl ;
11         NormDist[v_i] ← (MaxDist − e_cost(v_{r,n}, v_i)) / MaxDist ;
12         Decision[v_i] ← NormIdl[v_i] + NormDist[v_i];
13     v_{r,n+1} ← argmax(Decision[N_G(v_{r,n})]);   /* Next vertex is the neighbor of the current vertex with highest heuristic decision value. */
14     move_robot(v_{r,n+1});
15     v_{r,n} ← v_{r,n+1};
16     add(v_{r,n} to x_r);
17     update(Idl_{t_k}[V]);
```

4.3. *Heuristic pathfinder conscientious cognitive*

Unlike the two previous approaches, which use reactive agents that move only to close by vertices, Heuristic Pathfinder Conscientious Cognitive (HPCC) plans on the global graph to decide which vertex to move to subsequently. HPCC was also presented by Almeida [3] as a modified version of an approach called 'Conscientious Cognitive' previously described in [2].

Agents use a similar decision-making process as in HCR. However, instead of only moving to vertices in their neighborhood, they can move to any vertex of the graph. In addition, the algorithm takes into account the vertices on the way from the current one to the calculated destination. The chosen path depends on the instantaneous idleness and the distance of the vertices along the way. This is possible by computing new edge costs and running a Dijkstra shortest path algorithm, as seen in line 21 of Algorithm 3, which presents the pseudo-code of the approach.

Algorithm 3: Heuristic Pathfinder Conscientious Cognitive

```
1  r ← load(robot_id);
2  G ← load(topological_map);
3  v_{r,n} ← load(initial_vertex);
4  MaxCost ← max(e_cost(e_{i,j} ∈ E));
5  MinCost ← min(e_cost(e_{i,j} ∈ E));
6  add(v_{r,n} to x_r);
7  init(Idl_{t_k}[V]);
8  while true do
9      HIdl ← max(Idl_{t_k}[v_i ∈ V]);
10     MaxDist ← max(dijkstra(from v_{r,n} to all v_i ∈ V));
       // Heuristic Decision:
11     forall the v_i ≠ v_{r,n} ∈ V do
12         dist ← dijkstra(from v_{r,n} to v_i);
13         NormIdl[v_i] ← Idl_{t_k}[v_i] / HIdl ;
14         NormDist[v_i] ← (MaxDist − dist) / MaxDist ;
15         Decision[v_i] ← NormIdl[v_i] + NormDist[v_i];
16     v_{r,n+1} ← argmax(Decision[v_i ≠ v_{r,n} ∈ V]);
       // Path-Finding (Compute new edge costs):
17     forall the e_{i,j} ∈ E do
18         IdleCost ← (HIdl − Idl_{t_k}[v_j]) / HIdl ;
19         DistCost ← (e_cost(e_{i,j}) − MinCost) / (MaxCost − MinCost) ;
20         NewEdgeCost[i, j] ← IdleCost + DistCost;
21     path ← dijkstra_path(from v_{r,n} to v_{r,n+1});
           // Using NewEdgeCosts.
22     move_robot(path);
23     add(path to x_r); // Without 1st vertex (v_{r,n}).
24     v_{r,n} ← v_{r,n+1};
25     update(Idl_{t_k}[V]);
```

4.4. *Cyclic algorithm for generic graphs (CGG)*

Previous studies like [2] and [5] identify cyclic strategies based on the TSP as a method of guaranteeing low average idleness values, as long as these cycles can be computed for a given graph. However, no hints on the computation of such cycles, as well as on how to proceed in graphs with no cycles, are given. Solving TSP is NP-hard and is typically calculated using complete graphs[2]. The problem becomes even more complex when using generic non-complete graphs, like topological maps.

Consequently, inspired on the work of Elmaliach *et al.*, [4] a Cyclic Algorithm was implemented and used in [6]. It is essentially an offline graph theory-based method which

looks for Hamilton cycles or paths in the graph in order to visit all vertices. Due to the NP-Completeness of finding an Hamilton cycle, a fast heuristic algorithm proposed in [21] was used. When no such cyles or paths exist, the method looks for long paths and non-hamiltonian cycles as an alternative and computes detours to unvisited vertices, as seen in the pseudo-code of Algorithm 4. In this work, each robot is endowed with the ability of computing the final cycle. Hence the algorithm is run in a totally distributed fashion like the three previous ones.

Algorithm 4: Cyclic algorithm for generic graphs

```
1  build_x_r(G) {
2      main_path ← hamilton(G);
3      if main_path = ∅ then
4        cycle ← non_hamilton_cycle(G) ;
5        if size(cycle) > V/2 then
6          main_path ← cycle;
7        else
8              main_path ← longest_path(G) ;
9        end
10     end
11     final_path ← main_path + detours(all e_{i,j} ∈ E) ;
12     if main_path = hamilton_path ∨ longest_path then
13       final_path ← add_inverse_path() ;
14     end
15     return x_r;
16 }
17 r ← load(robot_id);
18 G ← load(topological_map);
19 x_r ← build_x_r(G);
20 v_{r,n} ← load(initial_vertex);
21 k ← 0 ;
22 while true do
23 v_{r,n+1} ← x_r [k + 1] ;
24 move_robot(v_{r,n+1});
25 k ++ ;
26 if k = size(x_r) − 1 then k ← 0;
27 end
```

4.5. *Generalized MSP algorithm (MSP)*

The MSP Algorithm [6] is an offline graph-theory based method, which partitions the graph into regions, where agents perform the patrol task. In the first phase of the algorithm, the graph is partitioned by a centralized entity, which then assigns regions to different robots. In a second phase, robots patrol their independent areas in a cyclic way, using a similar approach to the CGG method in their own subgraphs. The word 'Generalized' was added since the algorithm has the ability to partition the graph up to a given number of regions, being limited to the point where the graph can no longer be partitioned. Algorithm 5 presents the pseudo-code of the approach. The performance of the algorithm strongly depends on how balanced the partitioning of the graph is.

Algorithm 5: Generalized MSP.

```
1  r ← load(robot_id);
2  SG ← load(topological_region); // Assumes a
     previous centralized partitioning
     phase.
3  x_r ← build_x_r(SG); // Build path in the
     assigned subgraph.
4  v_{r,n} ← load(initial_vertex);
5  k ← 0 ;
6  while true do
7  | v_{r,n+1} ← x_r [k + 1] ;
8  | move_robot(v_{r,n+1});
9  | k ++ ;
10 | if k = size(x_r) − 1 then k ← 0;
```

5. Setting up the experiments

The performance and scalability of the five algorithms were compared using three topological maps chosen due to their different connectivity and complexity. To address the connectivity of the graph, a well-known metric of the graph was analyzed: the Fiedler value or algebraic connectivity.[22] In order to remove its dependency on the number of vertices in the spectrum of the Laplacian matrix, the Normalized Laplacian $\mathscr{L}$ [23] was adopted to obtain the Fiedler value of each graph.

All eigenvalues of $\mathscr{L}$ are non-negative and $\lambda_0 = 0$. For non-complete connected graphs (as is our case), the Fiedler Value λ_1 is the smallest non-zero eigenvalue of $\mathscr{L}$ and:

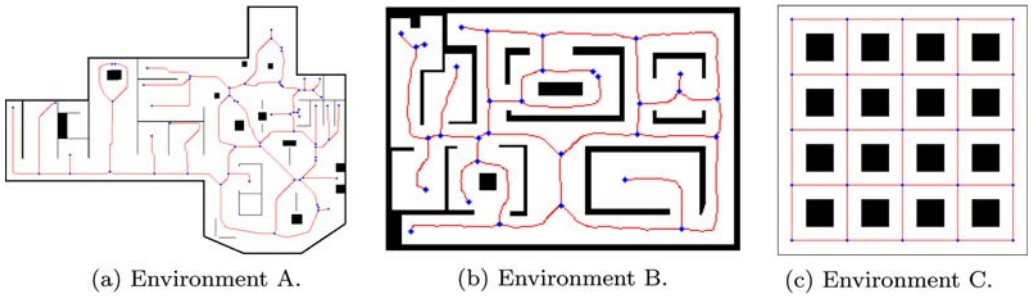
$$0 < \lambda_1 \leq 1 \tag{8}$$

Table 2 presents the connectivity properties of the graphs chosen for the experiments. Beyond the Fiedler Value, the Graph Density (D) was also calculated. This value represents a ratio between the number of edges and all possible edges if it were a complete graph:

$$D = \frac{2|E|}{|V|(|V|-1)} \tag{9}$$

All three graphs and the respective environments are presented in Figure 2. In addition, it was necessary to resort to a simulator since it would not be possible to obtain the extent of results presented in the next section, within reasonable time limits, if teams of real robots had been used. Therefore, a recognized simulator with realistic modeling was chosen: the Stage 2D multi-robot simulator.[24]

Stage considers the robot's dynamics and, together with ROS,[25] a framework for robot software development, was chosen to implement the experiments. The graph information of a given environment is loaded by every robot in the beginning of each simulation, which then runs one of the five algorithms described. Robots navigate safely in the environment by heading towards their goals and avoiding collisions with walls and other robots through the use of ROS's navigation stack [26] and a probabilistic localization system, more specifically the adaptive Monte Carlo local-



(a) Environment A. (b) Environment B. (c) Environment C.

Figure 2. Environments used in the experiments with respective topological map.

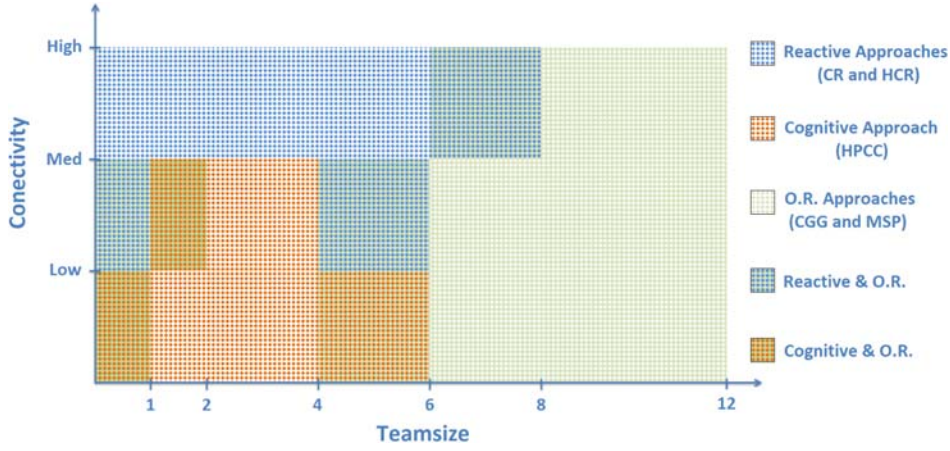


Figure 3. General simulation results. In this chart, the best strategies for a given map connectivity and team size are shown. Note that the figure presents some intersections of design solutions.

Table 2. Connectivity properties of the graphs used in the experiments.

Topological map	Environment area	$\|V\|$	Graph density (D)	Fiedler value (λ_1)
A	1357.17 m^2	66	0.0308	0.0080
B	1542.30 m^2	32	0.0746	0.0317
C	665.64 m^2	25	0.1333	0.1313

ization approach,[27] which uses a particle filter to track the pose of a robot against a known map. Note that this dynamic is implicit in all algorithms, when robots move. In addition, robots are non-holonomic and have a maximum velocity of 0.2 m/s.

6. Results and discussion

The simulation process involved running the five described patrolling strategies with six different team sizes (1, 2, 4, 6, 8, and 12 robots) in all three environments. Robots had the same starting positions for all algorithms when using the same team size and environment. Every trial was repeated three times, in a total of 264 simulations[3], which lasted around 345 hours with a cluster of four processors that were used due to the powerful computation demands of simulations, mainly those with higher team sizes. Simulations were when the value of the average graph idleness (Idl_G) after each patrolling cycle, i.e. every vertex visited, converges with 2.5% of tolerance. This resulted into an average simulation time of 1h18 m, which led to accurate and similar results between different trials; hence, there was no need to repeat the trials several times as testified by the overall average standard deviation of the results: $\overline{\sigma} = 4.42\%$.

The chart in Figure 3 represents environment connectivity vs. team size and depicts some general insights about the most suited solutions in different regions of the design space, providing a graphical overview of the results

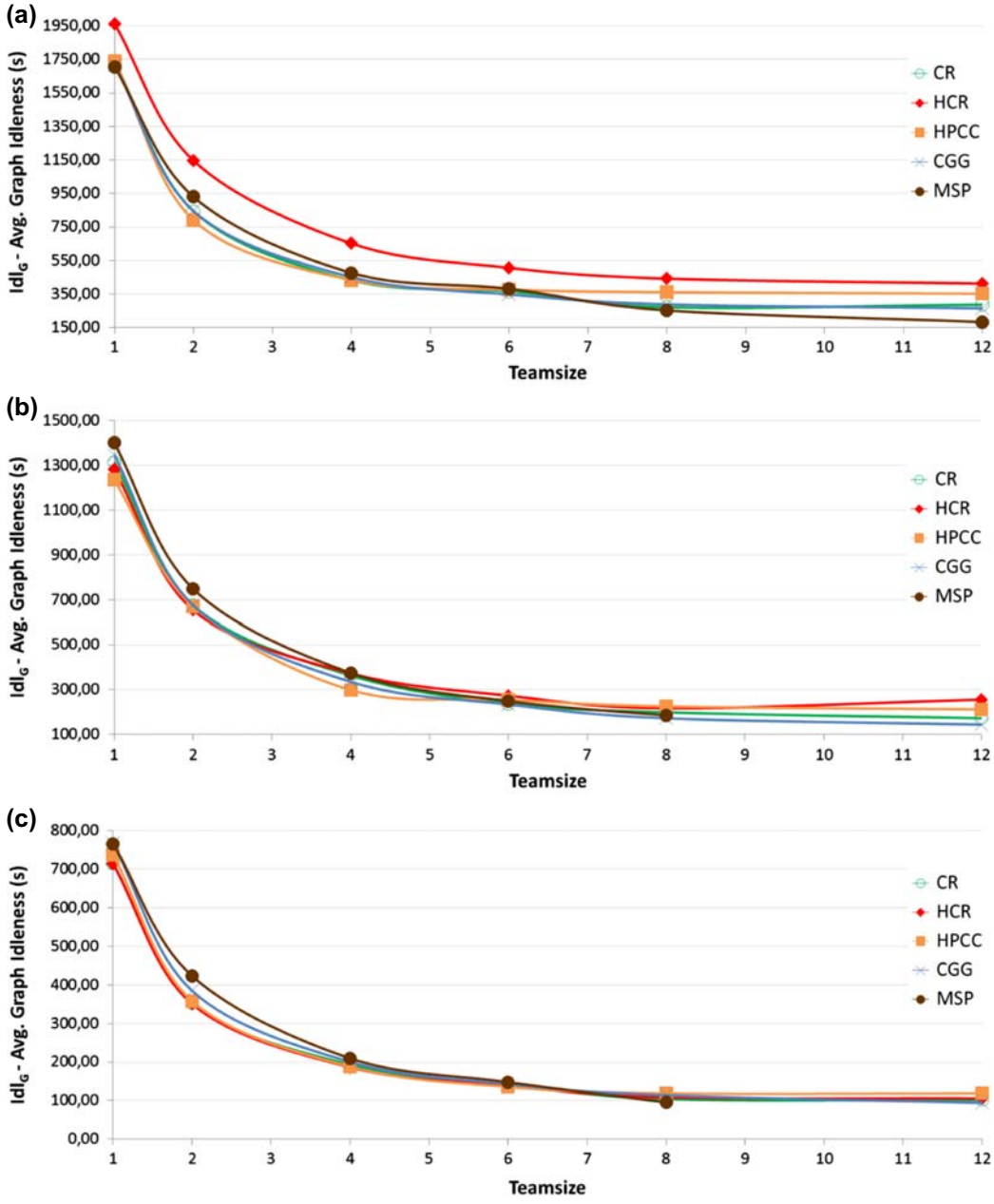


Figure 4. Simulation results: Idl_G performance curves. (a) Results for Environment A; (b) results for Environment B; (c) results for Environment C.

obtained. It is possible to verify that offline planning strategies (MSP and CGG) perform better in weakly connected environments than in strongly connected ones. This occurs because one can take better advantage of offline planning in such environments, while there are more path alternatives in strongly connected environments, where online planning performs adequately.

Generally, MSP is the algorithm with the best Idl_G values for larger teams, up to the point where the algorithm can no longer partition the graph. The method is not able to partition the topologies B and C in the 12 regions case, which happens due to limitations of the partition stage of the algorithm which is based on a fast multi-level approach for partitioning irregular graphs presented by Karypis and Kumar [28]. Nevertheless, these good results can be explained by low interference between agents when compared to other strategies, because each robot operates in a specific section of the environment. For smaller teams, the approach is not usually worth to employ, because it is more complex than simple reactive approaches and it does not lead to enhanced performance, mostly when the partitioning in regions is not as balanced as it would be desirable.

Moreover, CGG is the most regular algorithm, achieving fairly good results for all cases, especially in weakly

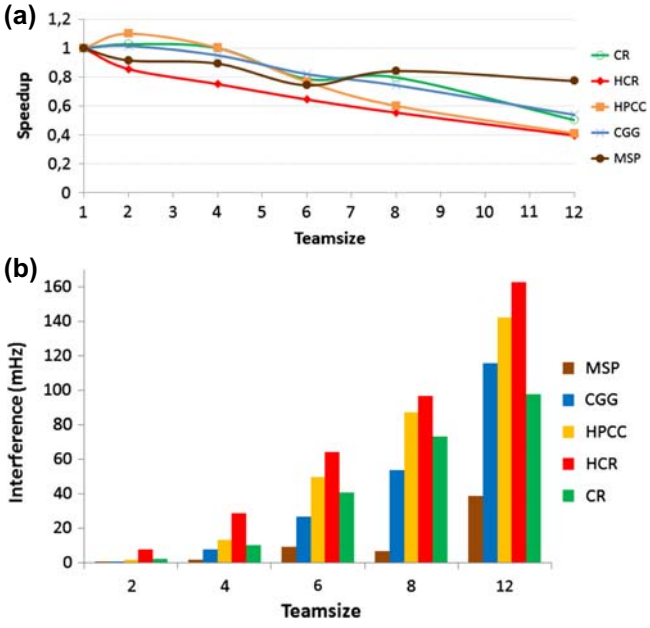


Figure 5. Speedup and interference in environment A.

connected environments or using larger teams, similarly to the MSP. However, it does not scale as well as the MSP as seen in Figure 4(a); e.g. note the 12 robots case.

On the other hand, HPCC proves to be an algorithm with good performance mostly for smaller teams, independently of the graph connectivity, given that, although it plans its decisions online, the entire graph is considered (unlike HCR and CR). Also, for the same reason, its performance drops for larger teams, because all robots wander and plan in the whole environment, which raises the probability of encounters between them. Results also show that this approach is the one that converges sooner to an Idl_G value, as the number of robots is increased, which indicates reduced scalability.

Moving on to reactive algorithms, it is interesting to observe that HCR does not present evident improvements when compared to CR. According to its authors, HCR can eventually be tuned to give different weights to the vertices' distance and the instantaneous idleness of neighbors during decision-making. In this work, the same weight for both parameters of the decision process was used and it was verified that for weakly connected environments, HCR was the algorithm with the worst performance (Figure 4(a) and Table 3). This happens because robots tend to stay longer in regions with close vertices, causing high interference between robots, which compete to reach those vertices, reducing overall performance dramatically. As for the CR algorithm, it scales better than HCR and HPCC, only staying behind the MSP and CGG for large teams. Reactive algorithms have good performance especially in strongly connected environments, as seen in Figure 4(c) and Table 5, where agents have alternatives to decide at the very moment, which vertex to move next to, taking into consideration the state of the system. Nevertheless, even in less connected environments, at some point when increasing the team size, the CR algorithm obtains better performance than the HPCC, since it scales better than the latter one.

Tables 3–5 show in detail a summary of all numerical results obtained in the simulation experiments. These were used to build the curves in Figure 4 completely clarifying and assisting the comparison between approaches, which is not always evident when the curves are too close. Each Idl_G value in the table is an average of three trials with the given algorithm, team size and map.

Additionally, as expected, all algorithms display increasing performance only until reaching a certain group size, around which the group productivity stagnates and even drops with the addition of robots; e.g. HCR in environment B as illustrated by Figure 4. In theory, productivity should grow during size scale-up; however, spatial limitations increase the interference between robots causing the decrease of performance. For example, calculating Balch's speedup measure [29] for increasing team sizes:

$$S[i] = \frac{\frac{P[1]}{i}}{P[i]} \tag{10}$$

where $P[i]$ is the performance for i robots, it is straightforward to conclude that such systems rapidly enter in sublinear performance ($S[i] < 1$), as shown in Figure 5(a) for environment A. On the other hand, in Figure 5(b) the interference, measured in the same environment, is presented. Interference is calculated as the number of times that robots had to avoid each other in order not to collide. Online planning strategies were the ones which presented more interference. It can be seen that speedup and interference are negatively correlated. For larger team sizes, instead of cooperating, robots tend to compete to firstly reach a given vertex than their teammates. Designing strategies which account for the teammates' goal can be beneficial for multi-robot patrolling, since they can take advantage of cooperation over competition between agents.

It is also interesting to see that Figure 4 and Tables 3–4 show that, even though map B has a larger area to patrol when compared to map A, all algorithms obtain lower Idl_G values for the same number of robots in environment B, due to its greater connectivity. These results prove that graph connectivity is a very important parameter to consider when employing a patrolling algorithm in a given environment. Expectedly, the performance of the team is also greatly affected by graph dimension. However, when independent of the connectivity, graph size is seen as a scale factor when considering fixed team sizes.

Furthermore, the median graph idleness value corresponds typically to around 85% of the average graph idleness, meaning that the frequency distribution is usually positively skewed (this is true in 96% of the trials). CR is the algorithm which has closest values between Idl_G and the median,

Table 3. Numerical results for map A.

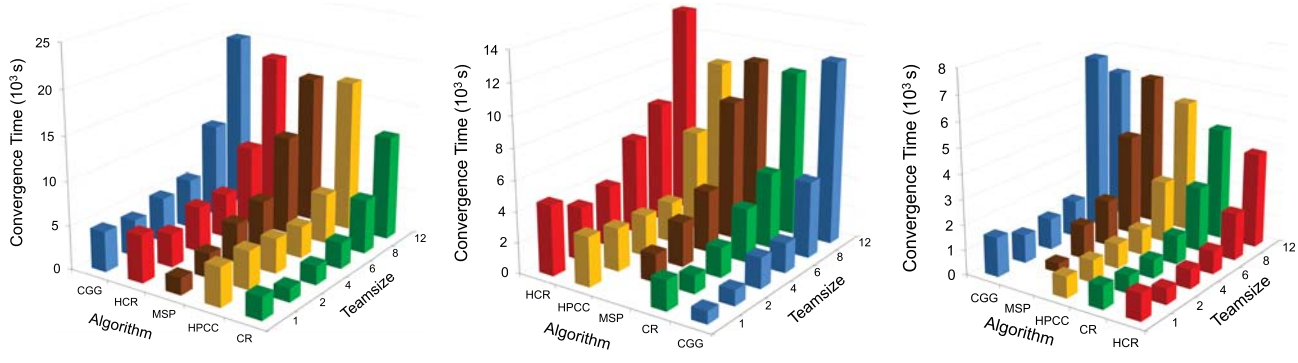
Team size	CR Idl_G	HCR Idl_G	HPCC Idl_G	CGG Idl_G	MSP Idl_G
1	1734.09	1962.42	1740.37	1717.36	1704.36
2	843.93	1146.27	791.20	845.49	930.04
4	433.38	652.84	434.11	451.70	476.92
6	367.11	506.90	377.73	348.46	381.97
8	271.70	442.39	361.62	288.72	253.19
12	287.14	412.65	352.79	265.47	183.74

Table 4. Numerical results for map B.

Team size	CR Idl_G	HCR Idl_G	HPCC Idl_G	CGG Idl_G	MSP Idl_G
1	1315.79	1283.59	1235.67	1347.30	1401.80
2	675.44	654.61	670.44	675.64	749.42
4	363.46	373.45	298.77	335.45	375.15
6	238.57	273.60	254.96	234.18	248.92
8	198.90	217.38	225.44	172.39	185.28
12	172.4	255.62	212.3	143.94	-

Table 5. Numerical results for map C.

Team size	CR Idl_G	HCR Idl_G	HPCC Idl_G	CGG Idl_G	MSP Idl_G
1	715.30	714.23	737.93	767.25	766.41
2	353.06	351.15	358.45	385.09	423.60
4	193.30	186.59	188.03	200.53	209.82
6	141.68	138.64	135.74	142.94	148.09
8	104.00	108.45	118.75	113.71	95.22
12	101.82	105.64	118.36	94.35	-



(a) Map A Convergence Time. (b) Map B Convergence Time. (c) Map C Convergence Time.

Figure 6. Average convergence time for each algorithm with different team size.

which shows that the algorithm normally does not let points in the environment stay idle for too long, balancing more its visits to the graph's vertices when compared to other approaches.

Table 6. ANOVA table.

Source	Sum Sq.	d.f.	Mean. Sq.	F	Prob>F
Algorithm	64953.125	4	16238.281	14.603	0
Team size	12763618.172	5	2552723.634	2295.671	0
Connectivity	2574860.148	2	1287430.074	1157.789	0
Algorithm*team size	36814.014	20	1840.701	1.655	0.0891
Algorithm*connectivity	120574.630	8	15071.829	13.554	0
Team size*connectivity	1279680.351	10	127968.035	115.082	0
Error	42254.968	38	1111.973	-	-
Total	17234127.010	87	-	-	-

The maximum idleness (most unvisited vertex of each graph) was also calculated. It is typically around 2.7 times larger than the average graph idleness. This ratio grows consistently with team size for all algorithms, being lower (around 2 times in average) for small team sizes and increasing for higher team sizes. As expected, CR due to its balanced property is the approach with a lower overall ratio of around 2.25 and surprisingly, if we consider the little difference between the two approaches, HCR is the algorithm with a higher ratio of around 3.25. The other three approaches have a ratio of around 2.6–2.7.

In terms of time taken to conclude the patrol task, it can be seen that CR usually needs less time to converge than the remaining approaches, as shown in Figure 6. This happens due to its property of constantly visiting places that have been idle for a long time, regardless of the distance to them. Consequently, it maintains a similar visit rate to all places. Despite this interesting aspect, it does not lead to better performance, when compared to other approaches. In fact, there is no apparent relation between performance and convergence time. Differences between the approaches are more marked with low number of robots as well as with different environment connectivity. Nevertheless, global trends can be observed. For instants, convergence time generally raises when team size increases from 8 to 12 robots. In such cluttered situations, robots spend inestimable time avoiding teammates, which highly affects convergence time.

In order to verify the significance of the problem's parameters tested in the experiments, three-way ANOVA [30] was applied, measuring quantitatively the group's variable effect. The parameters addressed were: algorithm, map connectivity and team size. Linear models were considered, assuming that the probability distribution of the response is normal, mutually independent and homoscedastic (i.e. the variance of the data inside the groups is equivalent). The test made use of the F-statistics distribution with 95% of confidence bounds and first-order interaction effects between pairs of factors, as seen in Table 6.

The only factor that presents no relevant significance is the algorithm*team size interaction, seeing as the null hypothesis was accepted. In fact, a clear indication of the low significance of this interaction is given by the Idl_G values of Tables 3–5, which do not differ much when each of the associated columns are compared as a whole. In addition, analyzing the individual factors, it can be seen that the influence of team size and connectivity in the results is greater than in the algorithm case. As a consequence, the interaction factor between team size and connectivity is the most significant interaction. These results are the natural evidence that performance relies heavily on the number of members in the team and the environment to patrol. Therefore, research should be guided towards approaches that ensure scalability and are appropriate, or perhaps can adapt, to all kinds of environment.

7. Conclusion and future work

In this work, a study of the scalability and performance of five different multi-robot patrolling strategies was presented. This study is unprecedented in this field because it overcomes many limitations and simplifications of previous works by using generic environments with different topological connectivity properties and weighted edges; realistic simulations that consider the robots' dynamics; and is based on the actual time in its performance metric instead of atomic iterations or simulation cycles. It was shown that different types of algorithms perform differently according to the environment and the number of robots running the patrol task. Consequently, the choice of a patrolling strategy for teams of multiple robots should take into consideration these two important parameters. Moreover, to improve the team's performance, scalable methods should be developed to minimize interference between robots.

In the future, we intend to deepen the study on the scalability properties of multi-robot patrolling algorithms by presenting an estimation method to dimension a team of robots in such missions according to the environment to patrol. Additionally, we intend to develop new scalable approaches for multi-robot patrol and test it in mobile robots and real scenarios.

Acknowledgements

This work was financially supported by a PhD grant (SFRH/BD/64426/2009) from the Portuguese Foundation for Science and Technology (FCT) and the Institute of Systems and Robotics (ISR-Coimbra).

Notes

1. The open neighborhood of a vertex v in a graph G is the induced subgraph of G consisting of v and all vertices adjacent to it, as well as all connecting edges between them.[19]
2. I.e. graphs in which every pair of distinct vertices is connected by a unique edge.[20]
3. The ROS simulation code is available at: http://www.ros.org/wiki/patrolling_sim

Notes on contributors

David Portugal is a PhD student and researcher at the Institute of Systems and Robotics (ISR), University of Coimbra, Portugal. He is under the supervision of Prof. Rui P. Rocha, being sponsored by a PhD scholarship from the Portuguese Foundation for Technology and Sciences. He holds an MSc. degree in Electrical Engineering and Computers obtained in 2009. Currently, he is collaborating as a researcher in the nationally funded Project CHOPIN, which deals with cooperation between human and robotics teams in catastrophic incidents. His research interests include cooperative robotics, multi-agent systems and optimization.

Rui P. Rocha is an assistant professor with the Department of Electrical Engineering and a senior researcher at the Institute of Systems and Robotics (ISR), both at the University of Coimbra, Portugal. He received the Engineering degree, the MSc degree, and the PhD degree in Electrical and Computer Engineering from University of Porto in 1996, 1999 and 2006, respectively. His main research topics are cooperative robotics, multi-robot systems, distributed robotics, cybernetic transportation systems, and 3D map building. He has been involved in several FP6 and FP7 European funded projects developed in consortium for the past few years and is currently the PI of a nationally funded research project about cooperation between human teams and robotics teams.

References

[1] Portugal D, Rocha RP. On the performance and scalability of multi-robot patrolling algorithms. In: Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR'2011); 2011 Nov 1–5; Kyoto (Japan). p. 50–55.

[2] Machado A, Ramalho G, Zucker J, Drogoul A. Multi-agent patrolling: an empirical analysis of alternative Architectures. In: Multi-Agent-Based Simulation, 3rd International Workshop; 2002 July 15–16; Bologna (Italy). p. 155–170.

[3] Almeida A. Patrulhamento Multiagente em Grafos com Pesos [M.Sc. Thesis]. Recife: Centro de Informática, Univ. Federal de Pernambuco; 2003. (In Portuguese).

[4] Elmaliach Y, Agmon N, Kaminka G. Multi-robot area patrol under frequency constraints. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2007). 2007 April; Italy. p. 385–390.

[5] Chevaleyre Y. Theoretical analysis of the multi-agent patrolling problem. In: Proceedings of the Intelligent Agent Technology: IAT'04, IEEE/WIC/ACM International Conference. 2004 Sep 20–24; Beijing (China). p. 302–308.

[6] Portugal D, Rocha R. MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning. In: Proceedings of 25th ACM Symposium on Applied Computing (SAC'2010), Special Track on Intelligent Robotic Systems; 2010 March 22–26; Sierre (Switzerland). p. 1271–1276.

[7] Sempé F, Drogoul A. Adaptive patrol for a group of robots. In: Proceedings of the International Conference on Intelligent Robots and Systems (IROS'2003); Oct 2003; Las Vegas (NV).

[8] Santana H. Ramalho G, Corruble V, Ratitch B. Multi-agent patrolling with reinforcement learning. In: Proceedings of the Third Int. Joint Conference on Autonomous Agents and Multiagent Systems. Vol. 3; New York, NY; 2004. p. 1122–1129.

[9] Hwang K, Lin J, Huang H. Cooperative patrol planning of multi-robot systems by a competitive auction system. In: Proceedings of the ICROS-SICE International Joint Conference; 2009 Aug 18–21; Fukuoka, (Japan).

[10] 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'2011); 2011 Sep 25–30; San Francisco, CA. p. 2796–2801.

[11] Almeida A, Ramalho G, Sanana H, Tedesco P, Menezes T, Corruble V, Chaveleyre Y. Recent advances on multi-agent patrolling. In: Brazilian Symposium on Artificial Intelligence (SBIA'2004). Vol. 3171; Sep 29 –Oct 1 2004; São Luís (Brazil). p. 474–483.

[12] Marier J, Besse C, Chaib-draa B. Solving the continuous time multiagent patrol problem. In: Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA'2010); 2010 May 3–8; Anchorage, AK.

[13] Basilico N, Gatti N, Rossi T, Ceppi S, Amigoni F. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'09). Vol. 2; 2009 Sep Milan (Italy). p. 15–18.

[14] Sampaio P, Ramalho G, Tedesco P. The gravitational strategy for the timed patrolling. In: Proceedings of the, 22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI'10); 2010 Oct 27–29; Arras (France). p. 113–120.

[15] Guo Y, Parker L, Madhavan R. Collaborative robots for infrastructure security applications. Studies in Computational Intelligence (SCI). Vol. 50, 185200. Berlin: Springer-Verlag; 2007.

[16] Chu H, Glad A, Simonin O, Sempé F, Drogoul A, Charpillet F. Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. In: Proceedings of the International Conference on Tools with Art. Intelligence. Vol. 1; France; 2007. p. 442–449.

[17] Portugal D, Rocha R. A survey on multi-robot patrolling algorithms. In: Proceedings of the 2nd Doctoral Conference on Computing, Electrical and Industrial Systems (DoCEIS'11); 2011 Feb 21–23; Costa da Caparica, Lisbon, Portugal.

[18] Balch T. Behavioral diversity in learning robot teams [PhD Thesis]. College of Computing Georgia Institute of Technology; Dec 1998.

[19] Diestel R. Graph theory. 3rd ed. Graduate texts in mathematics. Vol. 173. Heidelberg: Springer-Verlag; 2005.

[20] Gries D, Schneider F. A logical approach to discrete math. Springer-Verlag; 1993. p. 436.

[21] Angluin D, Valiant L. Fast probabilistic algorithms for Hamiltonian circuits and matchings. J. Comp. Sys. Sci. 1979;18:155–193.

[22] Fiedler M. Algebraic connectivity of Graphs. Czech. Math. J. 1973:23.
[23] Chung F. Spectral graph theory. AMS, Providence, RI: CBMS Lecture Notes; 1997.
[24] Gerkey B, Vaughan R, Howard A. The player/stage project: tools for multi-robot and distributed sensor systems. In: Proceedings of the Intelligent Conference on Advanced Robotics (ICAR'2003); 2003 July; Coimbra (Portugal); p. 317–323.
[25] Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R, Ng A. ROS: an open-source robot operating system. In: Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA'2009), Workshop On Open Source Software. 2009 May 12–17; Kobe (Japan).
[26] Marder-Eppstein E, Berger E, Foote T, Gerkey B, Konolige K. The office marathon: robust navigation in an indoor office environment. In: Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA'2010). 2010 May 3–8; Anchorage, AK. p. 300–307.
[27] Thrun S, Fox D, Burgard W, Dellaert F. Robust monte carlo localization for mobile robots. Artificial Intell. (AI). 2001;128:99–141.
[28] Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. Society for Industrial and Applied Mathematics (SIAM) J. Sci. Comput. 1998; 20:359–392.
[29] Balch T, Arkin R. Communication in reactive multiagent robotic systems. Auton. Robots. 1994;1:27–52.
[30] Scheffé H. The analysis of variance. New York, NY: John Wiley & Sons; 1959.