

# Multi-Agent Movement Coordination in Patrolling

Aydano Machado<sup>1</sup>, Alessandro Almeida<sup>1</sup>, Geber Ramalho<sup>1</sup>, Jean-Daniel Zucker<sup>2</sup>  
and Alexis Drogoul<sup>2</sup>

<sup>1</sup> Centro de Informática (CIn) – Universidade Federal de Pernambuco  
Caixa Postal: 7851  
50732-970 Recife-PE Brasil  
{apm, glr}@cin.ufpe.br

<sup>2</sup> Laboratoire d'Informatique de Paris VI (LIP6) – Université Paris 6  
Boîte 169 – 4 Place Jussieu  
75252 PARIS CEDEX 05  
{Jean-Daniel.Zucker, Alexis.Drogoul}@lip6.fr

**Abstract.** A group of agents can be used to perform patrolling tasks in a variety of domains, especially in computer games. Despite its wide range of potential applications, multi-agent architectures for patrolling have not been studied in depth yet. Firstly, state of the art approaches used to deal with related problems cannot be easily adapted to the patrolling task specificity. Furthermore, the existing patrolling-specific approaches are still in preliminary stages. In a previous and pioneer work, we presented an initial discussion of multi-agent patrolling task issues, as well as an empirical evaluation of possible solutions. In order to accomplish this study, we have proposed some architectures of multi-agent systems, evaluation criteria and experimental scenarios. In this paper we deepen this study, including new evaluation criteria and a more realistic representation of the patrolled terrain. The results show the adequacy of each agent architecture according to the characteristics of a given patrol task.

## 1. Introduction

To patrol is literally “the act of walking or travelling around an area, at regular intervals, in order to protect or supervise it”[1]. This task is by nature a multi-agent task and there are a wide variety of problems that may reformulate as particular patrol task. As a concrete example, during the development of the Artificial Intelligent component of an interactive computer wargame, we did face the problem of coordinating a group of units to patrol a given rough terrain in order to detect the presence of “enemies”. The quality of the agent architecture used for patrolling may be evaluated using different measures. Informally, a good strategy is one that minimizes the time lag between two passages to the same place and for all places.

Beyond our specific game, performing multiagent patrolling efficiently can be useful for various game categories where distributed surveillance, inspection or control are required. This is typically the case of strategy games (such as StarCraft, Civilization III) and RPG (such as Diablo and Ultima Online) for detecting mobile characters and new enemy buildings, protect cities and resources, etc. Multiagent patrolling can also be useful in combat games (such as Panzer Command and

StarWars Rogue Squadron), in especial in first-person shooters (such as Unreal Tournament and CounterStrike), for the detection of enemies and the discovery of items. Arcade games (such as PacMan) and simulators can directly use multiagent patrolling algorithms, too. Even collective sport games (such as FIFA Soccer and NBA basketball) can perhaps take profit of some kind of multiagent patrolling for positioning characters.

Despite its relevance, the patrolling task has not been seriously studied yet. On one hand, the literature has sound works concerning related studies, such as network mapping [10], the El Farol [3], steering behaviors [2, 4, 13]. However, these studies' characteristics are quite different from the patrolling task, which requires specific solutions. On the other hand, the works devoted precisely to patrolling tasks [8, 11, 12] do not present a systematic evaluation of the possible coordination strategies, agent models, agent society organizations, communication constraints, and so on.

In a previous and pioneer work, we presented an initial discussion of multi-agent patrolling task issues, as well as an empirical evaluation of possible solutions [9]. In order to accomplish this study, we have proposed some architectures of multi-agent systems, evaluation criteria and experimental scenarios. In this paper we deepen this study, including new evaluation criteria and a more realistic representation of the patrolled terrain. The results show the adequacy of each agent architecture according to the characteristics of a given patrol task.

The remainder of this paper is organized as follows. Next section defines precisely what we mean by the patrolling tasks. Section 3 shows the main steps we have followed in our study according to our proposed methodology. Section 4 presents the results and the discussion about them. Section 5 draws some conclusions and indicates directions for future work.

## 2. The Patrolling Task

There are many situations where one needs to protect, rescue, search, detect, oversee or track either something or someone. Computers can already perform these tasks in virtual worlds (such as those of computer games or computer networks) and, probably, in real world in a near future [16]. These tasks can involve some sort of patrolling, which may exhibit slightly different characteristics according to the domain and circumstances. It is then necessary, for our study, to have a more precise definition of patrolling.

In terms of area, the most complex case is the patrolling of continuous terrain, since the search space is large [15]. In these cases, one of the techniques used to change the representation of the terrain is *skeletonization* [14, 15], which consists of replacing the real terrain by a graph (skeleton) representing the possible paths as shown in **Fig. 1**. Voronoi diagrams, visibility graphs and C-cells can be used to generate such a graph. Once the terrain abstraction is available, the patrolling task is equivalent. A further advantage of adopting such an abstract representation is that the patrolling solutions proposed to it can be applied to different kind of problems, from terrain motion to web navigation. Given a graph, the patrolling task refers to

continuously visiting all the graph nodes so as to minimize the time lag between two visits.

There are some variations in the graph to be patrolled. In some situations, such as a terrain containing mobile obstacles, the graph edges may change. In other situations, priorities may be set to some regions covered by sub-graphs. The edges may have different associated lengths (weights) corresponding to the real distance between the nodes.

In our previous work [9] we have reduced the patrol task to graphs with the following characteristics: unitary edged length (the distance between two connected nodes is one), static edges (no mobile obstacles in the terrain), uniform patrolling (the same priority for all nodes). In the present work, we have considered the *real distance* between the nodes, representing them as the edge weights. In fact, the “unitary length” constraint is an oversimplification of the terrain representation, especially in application such as games, since the distance or cost between adjacent nodes is not constant.

### 3. Methodology

As discussed in the introduction, despite the applicability of building multi-agent systems for patrolling tasks, as far as we know, there is no systematic study concerning the subject in the literature. In particular, various questions remain opened, such as: which kind of multi-agent system (MAS) architecture should be chosen by the MAS designer for a given patrolling task? What are the means to evaluate an implemented MAS? To what extent parameters, like size and connectivity, influence the overall MAS performance?

To answer these questions we have adopted a methodology that consists of the following steps: definition of performance measures, proposition of different MAS architectures, definition of some case studies (patrolling scenarios), and the implementation of the simulator to perform the experiments. These steps will be explained in the rest of this session.

#### 3.1. Evaluation Criteria

One of the contributions of our work lies in the choice of evaluation criteria for comparing different MAS architectures, since no related work has used performance measures adapted to the patrolling task. As discussed next, we have chosen the following evaluation criteria: idleness, worst idleness, exploration time and search cost.

Considering that a cycle is the time necessary for an agent to go from a node to an adjacent one, we call *instantaneous node idleness* the number of cycles that a node has remained unvisited. This *instantaneous node idleness* is measured at each cycle. The *instantaneous graph idleness* is the average instantaneous idleness of all nodes in a given cycle. Finally, the *graph idleness*, or simply idleness, is the average *instantaneous graph idleness* over an n-cycle simulation.

In the same context, another interesting measure is the *worst idleness*, i.e. the biggest value of instantaneous node idleness occurred during the whole simulation.

The next evaluation criterion is, what we call, the *exploration time*, which consists of the number of cycles necessary to the agents to visit, at least once, all nodes of the graph. This corresponds intuitively to the notion of exploring an area in order to create its geographic diagram.

These performance measures naturally tend to exhibit better results as the number of agents patrolling the graph grows. However, if the coordination among the agents is not good enough, the improvement caused by the insertion of new agents may be minimized. In order to measure coordination quality as the number of agents augments, we have decided to measure the individual contribution of the agents, normalizing the three criteria (idleness, worst idleness and exploration time) as stated in equation 1:

$$\text{normalized\_value} = \text{absolute\_value} \times \frac{\text{number\_of\_agents}}{\text{number\_of\_nodes}} \quad (1)$$

The criteria discussed previously, which are the only criteria used in our preliminary study [9], are supposed to measure the *quality* of the patrolling solution. However, as in all search-like problems, we must deal with the search (solution) *cost*. For this reason, in our present work we have included another criterion, named *decision cost* [14]. This cost can be given by the sum of the time required by each agent to decide the node to be visited. The inclusion of this cost is extremely relevant since, in a realistic (real time) patrolling task, the longer the decision making time is, the longer the nodes remain without being visited.

### 3.2. Multi-Agent Systems to Be Investigated

In order to define which MAS architectures would be interesting to be evaluated in patrolling, we have explored four basic parameters (as displayed in Table 1). Of course, other parameters could have been taken into account, but the idea underlying our choice was to go from simple to complex architectures, avoiding premature experimentation. This bottom-up and incremental approach to agent design has been shown to be essential for understanding and measuring the impact of these architectures on the dynamics of a collective problem-solving process [6]. The architectures displayed in Table 1 follow this principle. For the same reason, we have considered only homogeneous architectures where the agents are all the same. The exceptions are the last two architectures, which contain a coordinator and various patrollers.

Name	Basic Type	Communication	Next Node Choice	Coordination Strategy
Random Reactive	reactive	none	random	emergent
Conscientious Reactive			idleness-based heuristics	
Reactive with Flags		flags		
Conscientious Cognitive	cognitive	none		
Blackboard Cognitive		blackboard		
Random Coordinator		messages	random	
Idleness Coordinator	idleness-based heuristics			

**Table 1.** Resume of the main features of the chosen agents.

Straightforwardly, the first parameter we have considered is the classical difference between reactive and cognitive agents: whereas reactive agents simply act based on their current perception, cognitive ones may pursue a goal. A further and natural constraint we have imposed is that the field of vision of reactive agents is one-node depth, i.e., a reactive agent only perceives the adjacent nodes. This follows the fact that reactive agents can not, by definition, plan a path to distant nodes. Cognitive Agents can choose any node of the whole graph as a goal-node and use path-finding techniques (Floyd-Warshall Algorithm in our case) to reach that node.

An important issue in performing collectively a patrolling task is the communication among the agents. Taking into account the real-world situations agents may face while patrolling, there are roughly three ways the agents can communicate with each other: via *flags*, via *blackboard*, and via *messages*. In the first case, agents leave flags or marks in the environment [5,6]. These marks are recognized by themselves or by the other agents. In the second case, the information about the environment is stored in a common base (e.g., a command and control center) that can be accessed by all agents. In the last case, agents can communicate with the others directly by exchanging *messages*. In a first moment, the agents can only exchange messages with the coordinator, when it exists. Enabling this kind of communication among all agents would require more complex architectures, including, for instance, negotiation mechanisms for conflict solving.

Another key aspect in multi-agent movement coordination is to use a central coordinator, which chooses the goal-node of each (cognitive) agent, or a decentralized one, where coordination emerges from agent interaction.

Next node choice is also an important point in generating possible solutions. The choice criteria can be *random* or *heuristically based on node idleness*, where an agent tends to visit the nodes with the highest idleness. Of course, two aspects directly influence this decision making. The first is the agent field of vision, which can be *local* or *global* as discussed earlier. The second, which applies to heuristic-based choice only, is the fact that, according the communication possibilities, a given agent can or cannot know what the other agents have been doing. Consequently, the node idleness is *individual*, in the sense that the agent considers only its own visits, or

*shared*, when the agent takes into account the movement of all agents. Considering these variations of the decision making process, the agents will perform the next node choice according to the strategies shown in Table 2.

Architecture Name	Detailed Next Node Choice
Random Reactive	locally random
Conscientious Reactive	locally individual idleness
Reactive with Flags	locally shared idleness
Conscientious Cognitive	globally individual idleness
Blackboard Cognitive	globally shared idleness
Random Coordinator	globally random
Idleness Coordinator	globally shared idleness

**Table 2.** Detailed decision making strategy used by the different agent architectures, considering the field of vision and communication.

It is worth noting that choosing nodes according to the individual idleness is equivalent to follow a gradient, as this technique is used in multi-agent systems [2,3]. In our case, nodes with high idleness act as valleys (attractors) and those with low idleness act as mountains (repulsors).

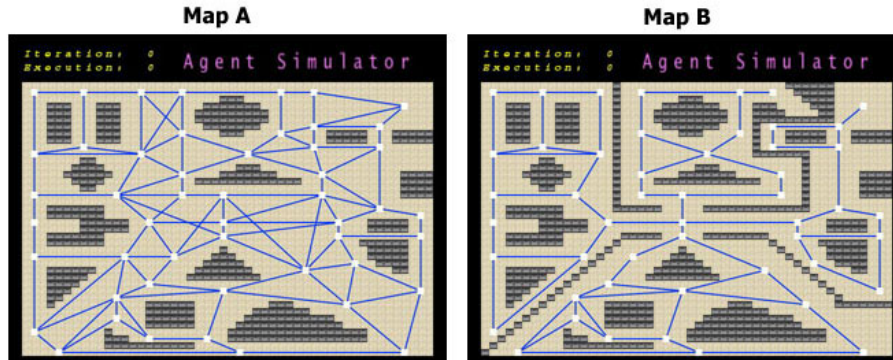
There are several possible MAS architectures resulting from the combination of these four parameters. We have studied all of them and then chosen the ones that seemed to be the most appropriated to the task.

It is finally worth indicating that we have also considered another coordination parameter (not shown in Table 1): the *monitoring capability*. While a cognitive agent is following a path to its goal-node, it is useful to monitor whether any other agent is visiting this given node in the meantime in order to reassign another goal. After the first experiments, we have noticed that agents with monitoring capabilities always performed better than the equivalent agents without monitoring. In order to keep the presentation of results graphics more readable, we have just included in this paper the agents capable of monitoring (whenever monitoring is feasible), i.e. Blackboard Cognitive Agent, Random Coordinator and Idleness Coordinator.

### 3.3. Experiment scenarios

After reflecting about the influence the environment parameters could have on the system performance, we have realized that more than the number of nodes, it is important to control the graph connectivity, i.e. the number of edges. In this perspective, we have created two different maps (as shown on **Fig. 1**). Map A has few obstacles and a highly connected graph, representing the fact that it is easy to go to any region. Map B has some bottlenecks, generating a graph with the same number of nodes but with much fewer edges.

Instead of changing the number of nodes, we have equivalently changed the number of agents. We have used populations of 1, 2, 5, 10, 15, and 25 agents in order to keep adequate ratios between the number of nodes (50) and the number of agents.



**Fig. 1.** Maps A and B in our simulator. Black blocks in the maps represent obstacles. The graphs of possible paths are shown. The graph of Map A has 106 edges and 50 nodes, and the graph of Map B, with more bottlenecks, has 69 edges and the same 50 nodes. These figures are also snapshots of the simulator.

### 3.4. Simulator

In order to accomplish a larger number of experiments, we have developed a simulator using C++/OpenGL, which is a commonly used development platform in the computer games community. This simulator implements the agents defined in Table 1 and emulates the patrolling task recording the data for later analysis.

A map is described in a proprietary format, allowing the researcher to indicate all the characteristics of environment, such as the size of the map, the obstacles, the graph, the agents initial position, the number and kind of MAS architecture to use (according to Table 1), the number of steps to run, etc.

## 4. Experimental Results and Discussion

For each of the seven MAS architectures of Table 1, we have run 12 simulations (2 x 6), corresponding to 2 maps (A and B) and 6 different populations of agents (1, 2, 5, 10, 15 and 25). For each population, the initial position of the agents is the same in all architectures.

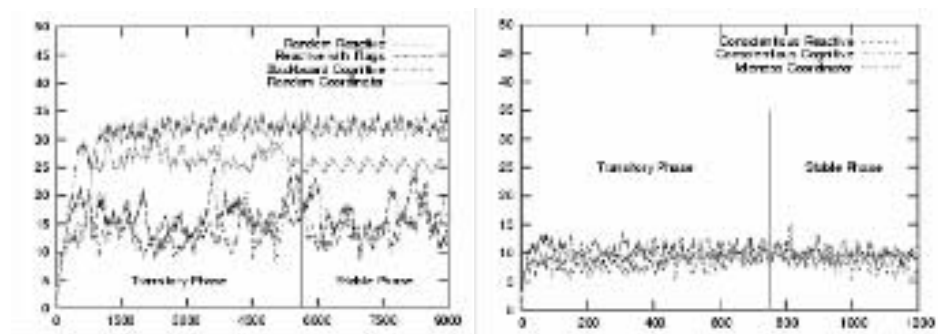
As previously mentioned, the agents were initially implemented considering that the weights of the edges (distance between nodes) were unitary. After implementing the agents considering real distances, the tests had to be redone, so as to determine what would change.

When the weights were not considered, each simulation was composed of 3000 cycles, i.e. the agents traversed 3000 edges during the simulation. In the real distance case, an agent, in order to traverse an edge, will need a number of cycles proportional to the edge weight (distance). Hence, the quantity of cycles of a simulation using real distance should be so that the agents can traverse the same quantity of edges, in

average, as those traversed using unary distance. It was then necessary to determine the mean weight of the edges of each map's graph. This mean weight is about 7,6 distance units in both cases. Therefore, to traverse each edge when we consider real distance, we need about 7,6 cycles. Since we want that each agent traverse 3000 edges,  $7,6 * 3000 \cong 23000$  cycles are needed in the simulation. Consequently, each simulation is composed of 23000 cycles.

At the beginning of a simulation, we consider that all instantaneous node idleness is zero, as they had just been visited. Consequently, there is a sort of transitory phase in which the instantaneous graph idleness tends to be low, not corresponding to the reality in a steady-state phase, as shown in **Fig. 2**. For this reason, the (final) graph idleness is measured only during the stable phase. According to some early experiments, we have noticed that the transitory phase always finishes before the cycle 5600 (except for the Random Reactive Agents and the Random Coordinator whose behavior may sometimes be highly unstable). The graph idleness is, then, measured along the remaining 17400 cycles.

Finally, since the number of agents does not affect the individual decision cost of each agent, we did not vary the number of agents while measuring the decision cost. A number of 5 agents was arbitrarily chosen in both maps for each architecture.



**Fig. 2.** The graphics show the evolution of 5 agents during a simulation (idleness in y-axis and cycles in x axis).

#### 4.1. Results graphics

In the following graphics, each different line type represents a different MAS architecture and the vertical bars in the idleness graphics show the standard deviations on the average calculated along a simulation. At first, we compare the usage of unitary and real distances. The graphics of the configurations using unitary distance are on the left, whereas those using real distance are on the right. **Fig. 3** and **Fig. 4** show the graphics using unitary and real distance in Map A according to the criteria idleness and worst idleness, respectively. **Fig. 5** compares the exploration time in Map B using unitary and real distance.

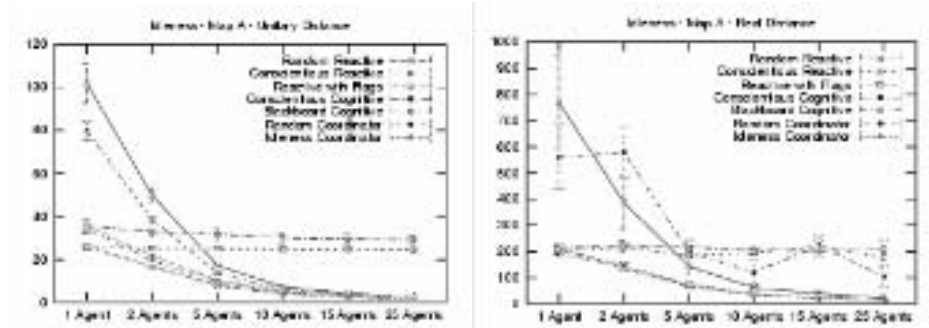


Fig. 3. Comparison of the idleness considering unitary distance and real distance

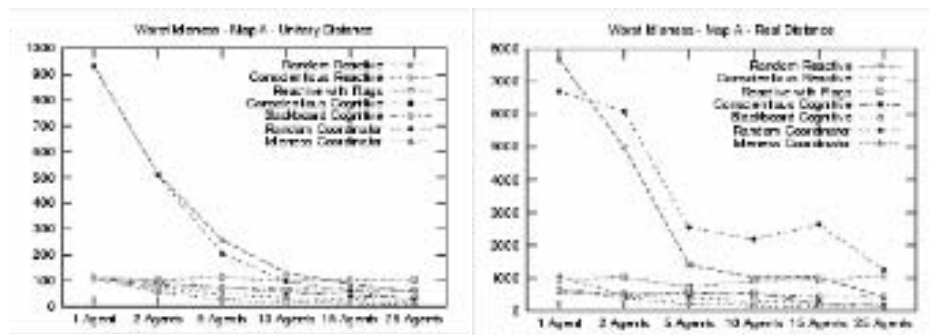


Fig. 4. Comparison of the worst idleness considering unitary distance and real distance

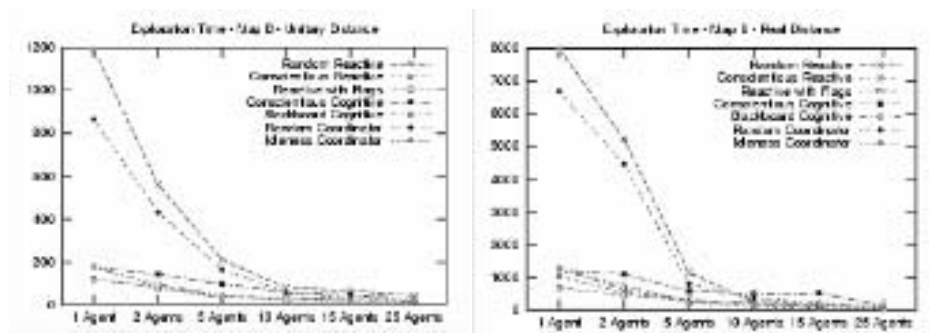


Fig. 5. Comparison of the exploration time considering unitary distance and real distance

The following graphics are also shown in pairs, showing the absolute performance on the left and the normalized one on the right. All the architectures shown consider the real distances between the nodes.

Fig. 6 and Fig. 7 show the graph idleness measures (in y-axis), and the corresponding normalized value, in Map A and B respectively, as the number of agents grows (x-axis). Similarly, Fig. 8 and Fig. 9 present the results of the graph

worst idleness. The main difference in this case is that worst idleness is measured over the 23000 cycles, whereas (average) idleness only considers the stable phase.

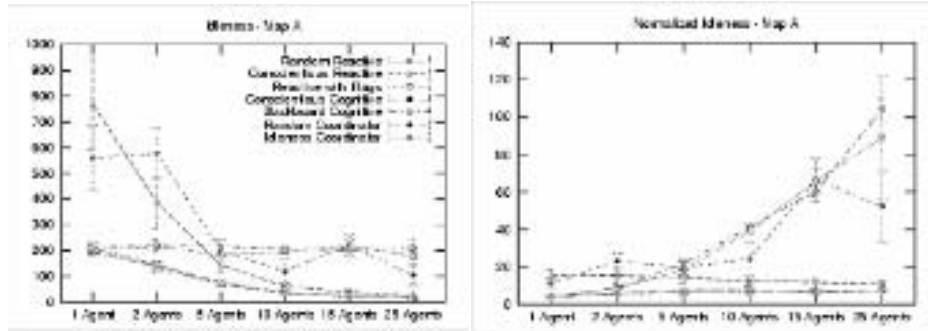


Fig. 6. Graph representing Idleness and Normalized Idleness for Map A.

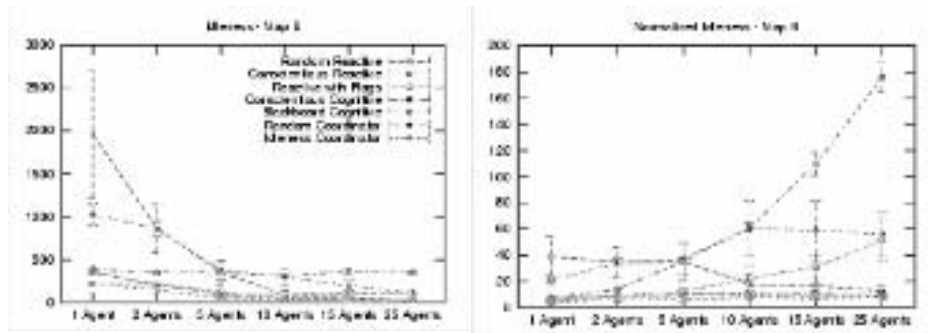


Fig. 7. Graph Idleness and Normalized Idleness for Map B.

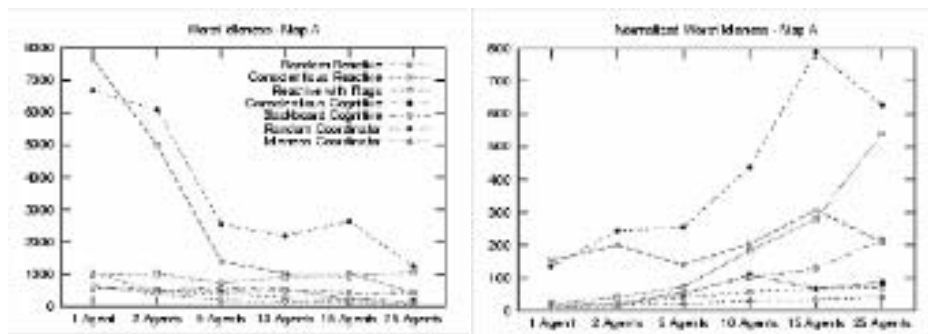


Fig. 8. Graph Worst Idleness and Normalized Worst Idleness in Map A.

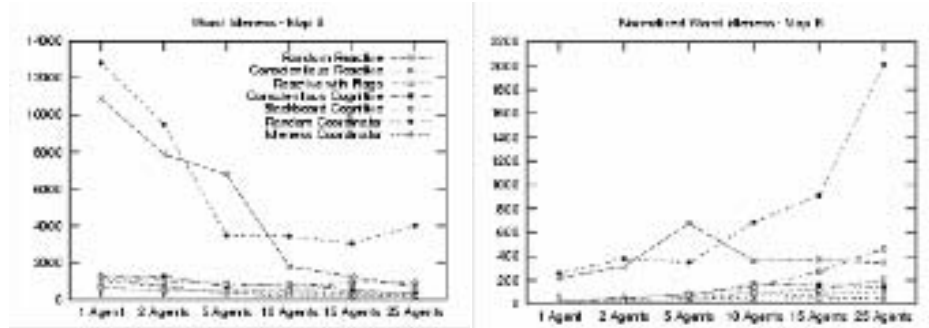


Fig. 9. Graph of Worst Idleness and Normalized Worst Idleness in Map B.

Finally, **Fig. 10** and **Fig. 11** plot the number of cycles required for a complete exploration (exploration time) of the maps A and B, respectively.

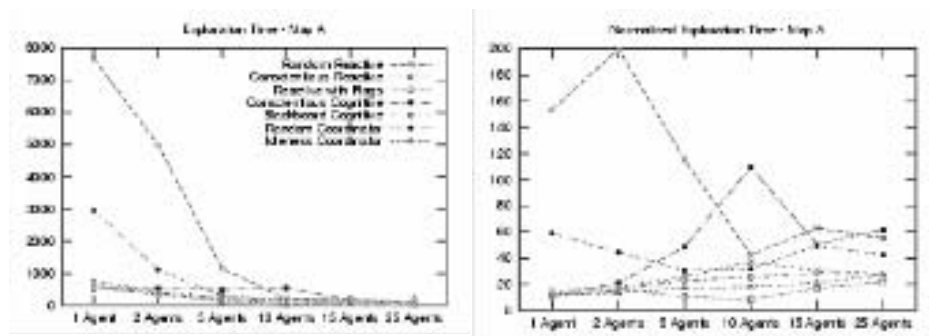


Fig. 10. Graph Exploration Time and Normalized Exploration Time in Map A.

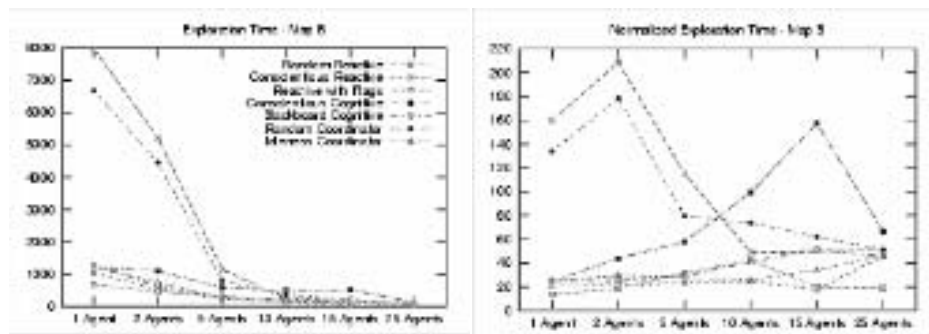
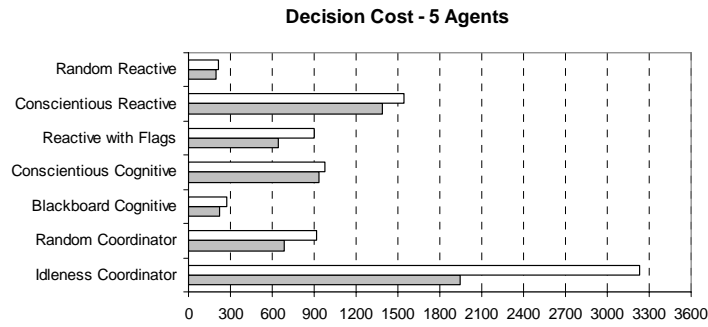


Fig. 11. Graph Exploration Time and Normalized Exploration Time in Map B.

## 4.2. Decision Cost Result Graphics

**Fig. 12** shows the graphics of the decision cost for 5 agents. The results of all architectures in both maps are displayed.



**Fig. 12.** Decision cost for 5 agents in Maps A and B.

## 4.3. Discussion

When we consider the real distance between two nodes instead of a unitary distance, the change is not significant for the majority of agent architectures. However, a difference can be noticed in the random architectures due to its inherent random behavior. In other architectures, there are also some changes, but they are extremely small. In all the evaluation criteria, the best architectures when we consider unitary distances are still the best when considering real distance. The same occurs with the worst architectures.

From a general perspective, we can see three distinct groups which we call *random*, *“non-coordinated”* and *top group*. Table 2 indicates the composition of each group.

*The top group* obtained the best results in all metrics. The Conscientious Reactive Agent performance has been a little better than the other agents of this group, but this small difference has tended to zero with the increasing of the population.

*The random group* has presented the worst results for small populations. These results have been improved, being almost equivalent to the top group, with more numerous populations. A strong characteristic of this group is its unpredictable behavior (there is not exactly a “stable phase”), which is reflected in large standard deviations.

*The non-coordinated group* presented a behavior that we already expected: every agent tends to go to the same places at the same moment. Consequently, the groups somehow behave as a single agent.

Group	Agents
Random Group	Random Reactive Agent
	Random Coordinator
Non-coordinated Group	Reactive Agent with Flags
	Blackboard Cognitive Agent
Top Group	Conscientious Reactive Agent
	Conscientious Cognitive Agent
	Idleness Coordinator

**Table 3.** MAS architecture groups

Concerning Map A vs. Map B. (i.e., graph connectivity) the multi-agent system performance has been always worse in Map B than in Map A. This has been detected in all experiments, using all metrics. *The top group* has been less affected by the bottlenecks in Map B and got similar results in both maps. *The random group* was the most affected, with truly bad results in Map B.

The *normalized results* are very interesting since they show the individual contribution of each agent in the architecture. Moreover, the normalized results indicate clearly the coordination capability of a given architecture: the best is the coordination, the strongest is the impact of adding new agents to the architecture. For instance, the performance of the non-coordinated group is even worse considering the normalized measured. Regarding the *exploration time*, curiously increasing population does not yield a significant increasing of the individual performance, no matter the kind of MAS architecture used.

Looking from a decision cost perspective, we can notice a quite better performance in Map B. This is mainly owing to the fact that in Map A, as graph connectivity is higher, there are more choices to the agents at each reasoning step of the simulation. If we compare the architectures *Reactive with Flags* with *Conscientious Reactive* and *Blackboard Cognitive* with *Conscientious Cognitive*, we will notice that architectures with the presence of communication always perform better because in those architectures each agent access a global graph rather than accessing its own local graph. With the absence of communication, there is an overhead of updating the idleness in many local graphs instead of updating just one global graph. Moreover, if we compare the architectures *Conscientious Cognitive* with *Conscientious Reactive* and *Blackboard Cognitive* with *Reactive with Flags*, we will surprisingly notice that cognitive architectures always perform better than reactive ones. This can be explained by the fact that in the vast majority of steps, cognitive agents are just following a pre-determined path instead of deciding at each step which node to go next as reactive agents do. Lastly, when there exists explicit coordination by means of a central coordinator, performance is bound to decrease, probably because the coordinator needs to manage goal assignment among the agents.

Besides identifying the top group of MAS architectures, these experiments show us some preliminary guidelines in designing MAS for patrolling. The first and main step is to understand the application domain constraints and characteristics. It is essential to determine the path graph in terms of number nodes and connectivity, the availability of agent communication, the maximum accepted idleness, the desired average idleness and idleness variation over the cycles, and the necessity of an exploration phase and the maximum time lag for it. Knowing these characteristics, it

will be easier to choose the best MAS architecture. For instance, for graphs containing bottlenecks, random approaches are not recommended. If no significant variation on idleness is desired, random approaches should also be discarded. Moreover, according to the desired idleness, the number of agents can be determined (the ratio of one agent to 10 nodes yields good enough results).

## 5. Related Work

We have found some related works, but they do not study deeply the patrolling in terms of idleness, exploration time, or any evaluation measure of the solution quality.

In the majority of computer games, patrolling consists of moving to a series of specified positions in a given order. The order of these positions can be changed to make the movements less artificial [8]. The work of Pottinger is concerned with group movement. The techniques used are based on relative positions to a unit of the group, i.e., all the movement coordination is made through relative positions to a unit in special [11,12]. These works are focused on movement, not exactly in an emergent behavior that could improve patrolling efficacy.

In a different approach, but following the same basic concerns, there are the behavior-based robotics systems [2,4] and the steering behaviors agents [13]. These reactive systems provide navigation capabilities by combining simple behaviors that are based on the immediacy of sensory information, channeling this information directly to motor behavior without use of intervening symbolic representations. Patrolling is not directly addressed in these works.

## 6. Conclusions

This work presents some original contributions to the problem of multi-agent patrolling. First, we have given a general characterization of the multi-agent patrolling problem, pointing also its application in a variety of domains. Second, we have proposed a method for evaluating different MAS architectures for patrolling. Third, we have proposed some MAS architectures as well as a preliminary typology, according to some coordination parameters (this typology follows the same approach we have been using in other tasks [17]). Finally, this work furnishes some preliminary guidelines for MAS designer interested in patrolling tasks.

The simulator we have developed is available upon request for other researchers wanting to experiment their patrolling strategy.

In the future we intend to augment the complexity of the agent and MAS architectures. This includes features such as different path-finding techniques, which instead of searching shortest paths take into account the instantaneous idleness of the nodes in-between the current location and the goal. Finally, we will enable messages exchange among all agents, opening the opportunity to explore negotiations mechanisms for conflicts resolution.

## References

1. Abate, Frank R.: The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers. Oxford Univ. Press. 1996
2. Arkin, Ronald C.: Behavior-Based Robot Navigation for Extended Domains. Adaptive Behaviors. Fall 1992, vol. 1(2):201-225
3. Arthur, W. B.: Inductive Reasoning and Bounded Rationality (The El Farol Problem). American Economic Review (1994) 84: 406-411.
4. Balch, Tucker and Arkin, Ronald C.: Behavior-Based Formation Control for Multi-robot Teams. IEEE Transactions on Robot and Automation (1999) vol. XX
5. Dorigo, M. Maniezzo, V. & Coloni, A. The Ant System: optimization by a colony of cooperating agents. IEE Tarns. System, Man and Cybernetics B26(1) (1996). 29-41
6. Drogoul, A. et A. Collinot. Applying an Agent-Oriented Methodology to the Design of Artificial Organizations: a Case Study in Robotic Soccer. Journal of Autonomous Agents and Multi-Agent Systems 1(1): 113-129. 1998
7. Ferber, Jacques: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley (1999) 439-445.
8. Howland, Geoff: A Practical Guide to Building a Complete Game AI: Volume II. [http://www.lupinegames.com/articles/prac\\_ai\\_2.html](http://www.lupinegames.com/articles/prac_ai_2.html), 1999
9. Machado, A., Ramalho, G., Zucker, J.-D. and Drogoul, A. Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures. To appear in Multi-Agent Based Simulation (MABS'2002), Bologna, 2002.
10. Minar N., Hultman K, and Maes P. Cooperating Mobile Agents for Mapping Networks. In the Proceedings of the First Hungarian National Conference on Agent Based Computing, 1998
11. Pottinger, Dave C.: Coordinated Unit Movement. Game Developer (January 1999) 42-51
12. Pottinger, Dave C.: Implementing Coordinated Unit Movement. Game Developer (February 1999) 48-58
13. Reynolds, C.W.: Steering Behaviors for Autonomous Characters. Presented at Game Developers Conference (1999). <http://www.red3d.com/cwr/steer/>
14. Russell, Stuart J. and Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (1995) 61; 796-808
15. Stout, Brian W.: Smart Moves: Intelligent Path-Finding. Game Developer (October/November 1996) 28-35
16. Sukthankar, G. and Sycara K.: Team-aware Robotic Demining Agents for Military Simulation. Robotics Institute - Carnegie Mellon University. <http://www-2.cs.cmu.edu/~softagents/iaai00/iaai00.html>, 2000.
17. Zucker, J.-D. and C. Meyer. Apprentissage pour l'anticipation de comportements de joueurs humains dans les jeux à information complète et imparfaite: les "Mind-Reading Machines". Revue d'Intelligence Artificielle 14(3-4). (2000). 313-338