Multi-agent Patrolling: An Empirical Analysis of Alternative Architectures

Aydano Machado[1], Geber Ramalho[1], Jean-Daniel Zucker[2], and Alexis Drogoul[2]

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

[2] 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 ranging from computer network administration to computer wargame simulations. Despite its wide range of potential applications, multi-agent architectures for patrolling have not been studied in depth yet. First state of the art approaches used to deal with related problems cannot be easily adapted to the patrolling task specificity. Second, the existing patrolling-specific approaches are still in preliminary stages. In this paper, we present an original in-depth 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 different architectures of multi-agent systems, various evaluation criteria, two experimental scenarios, and we have implemented a patrolling simulator. The results show which kind of architecture can patrol an area more adequately according to the circumstances.

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 simulators and computer games, performing this patrolling task efficiently can be useful for various application domains where distributed surveillance, inspection or control are requires. For instance, patrolling agents can be used for helping administrators in the surveillance of failures or specific situations in a Intranet [2], for detecting recently modified or new web pages to be indexed by search engines

J.S. Sichman, F. Bousquet, P. Davidsson (Eds.): MABS 2002, LNAI 2581, pp. 155-170, 2003.

[6], for identifying objects or people in dangerous situations that should be rescued by robots [14], etc.

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 [4], steering behaviors [3, 5, 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 [9, 11, 12] do not present a systematic evaluation of the possible coordination strategies, agent models, agent society organizations, communication constraints, and so on.

In this paper we present an original in-depth analysis of the patrolling task issues and the possible multi-agent-based solutions. To do this study we followed a methodology. First, we have defined some criteria for evaluating the solutions. Second, we have proposed several multi-agent architectures varying parameters such as agent type (reactive vs. cognitive), agent communication (allowed vs. forbidden), coordination scheme (central and explicit vs. emergent), agent perception (local vs. global), decision-making (random selection vs. goal-oriented selection), etc. Third, we have implemented a dedicated simulator to enable the experimental tests. Fourth, we have tested the different solutions using two scenarios.

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 the methodology we mentioned early. 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 [17]. These tasks can involve some sort of patrolling, which may exhibits 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 [16]. In these cases, one of the techniques used to change the representation of the terrain is *skeletonization* [15, 16], which consist 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 case study, we have reduced the patrol task to graphs with the following characteristics: static edges (no mobile obstacles in the terrain), unitary edged length (the distance between two connected nodes is one), uniform patrolling (the same priority for all nodes). In other words, given N agents, K nodes, connected by edges with equal weights, the patrolling problem is to achieve a global behavior that minimizes the time lag in which any agent has not visited a node.

3 Methodology

As discussed in the introduction, despite the applicability of 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 defining performance measures adapted to the patrolling task is still an open problem in the related works [9, 11, 12]. As discussed next, we have chosen the following evaluation criteria: idleness, worst idleness and exploration time. Other criteria could have been adopted, but the ones we propose are adequate to measure the quality of the solutions.

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 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 last 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 exhibits 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):

$$normalized\ \_value = absolute\ \_value \times \frac{number\ \_of\ \_agents}{number\ \_of\ \_nodes} \tag{1}$$

3.2 Multi-agent Systems to Be Investigated

In order to define the MAS architectures that would be interesting to be evaluated in the patrolling task, 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 explore the choice of agent architectures described in the literature in a methodological way. 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 [19]. The architectures displayed in Table 1 follow this principle. In the same way, we have only considered homogeneous groups of agents (i.e., all the agents share the same architecture), except, of course, the last two ones, which require an explicit coordinator agent

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

Architecture Name	Basic Type	Communication	Next Node Choice	Coordination Strategy
Random Reactive	reactive	none	locally random	emergent
Conscientious Reactive			locally individual idleness	
Reactive with Flags		flags	locally shared idleness	
Conscientious Cognitive	cognitive	none	globally individual idleness	
Blackboard Cognitive		blackboard	globally shared idleness	
Random Coordinator		messages	globally random	central
Idleness Coordinator			globally shared idleness	

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 perceive a depth d ($d > 1$) of graph, in this work the agents can perceive the whole graph and use *path-finding* techniques (Floyd-Warshall Algorithm in our case) to reach any goal-node.

An important issue in performing collectively a patrolling task is the communication among the agents. Taking into account the real-word situations agents may face while patrolling, there are roughly three ways for the agents communicate to each other: via *flags*, via *blackboard*, and via *messages*. In the first case, agents leave flags or marks in the environment [7, 8]. 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, such as the next node choice.

Decision-making is also an important point in generating possible solutions. In other words, the question is to determine how the *next node* will be chosen. Two aspects should be considered: the field of vision, which can be *local* or *global* as discussed earlier; and the choice criteria, which can be random or heuristically based on node idleness. In the node idleness case, there are two variations depending on whether an agent knows about what the other agents have been doing. In the *individual idleness* heuristic, the agent considers only its own visits, whereas in the *shared idleness* heuristic, it takes into account the movement of all agents. Choosing nodes according to the individual idleness is equivalent to follow a gradient, as this technique is used in multi-agent systems [3, 5].

Finally, a 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.

There are several possible MAS architectures of combining these four parameters. We have studied all of them and then chosen the ones that seemed to be the most appropriated to the task (Cf. Table 1, column 1).

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.

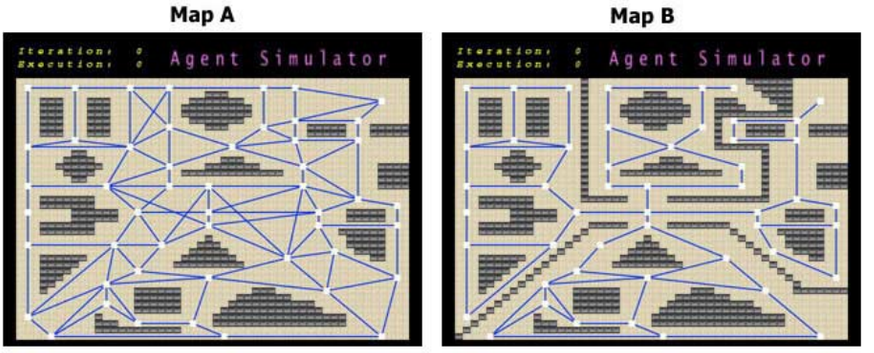

Fig. 1. Maps A and B in our simulator. Black blocks in the maps represent obstacles. The graphs of possible paths are shown. The graph of Map A has 106 edges and 50 nodes, and the graph of Map B, with more bottlenecks, has 69 edges and the same 50 nodes. These figures are also snapshots of the simulator.

3.4 Simulator

In order to accomplish a larger number of experiments, we have developed a dedicated simulator using C++/OpenGL, which is a commonly used development platform in 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 (column 1), we have run 360 (2 x 6 x 30) simulations, corresponding to 2 maps (A and B), six different number of agents (1, 2, 5, 10, 15 and 25) and 30 different staring points (i.e., initial positions of the agents). The 30 initial positions are randomly chosen once and then used in testing the different architectures.

Each simulation is composed of 3000 cycles, i.e. the agents change from a node to another 3000 times. At the beginning of 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 750 (except for the Random Reactive Agents and the Random coordinator whose behavior may sometimes be highly unstable). The graph idleness is the measured along the remaining 2250 cycles.

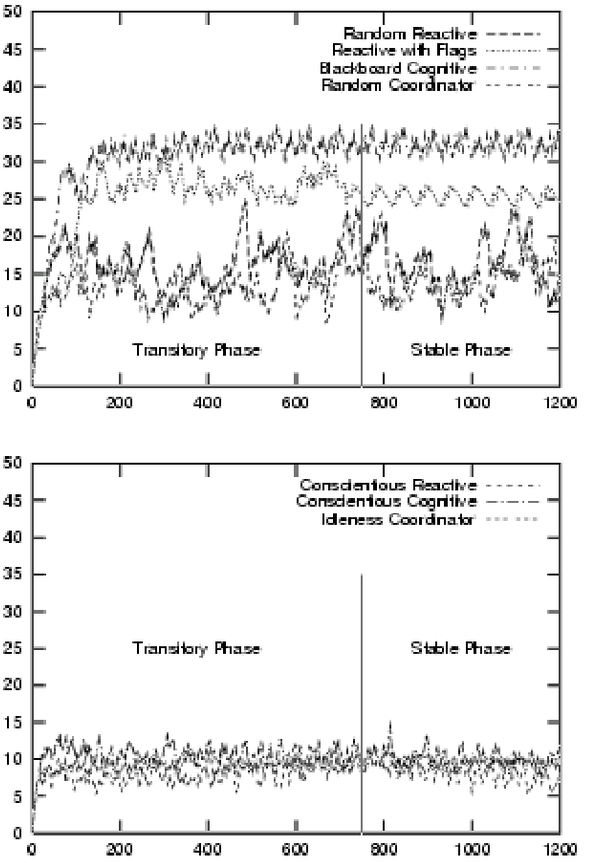

Fig. 2. The graphics show the evolution of 5 agents during a simulation (idleness in y-axis and cycles in x axis).

4.1 Results Graphics

In the following graphics, each different line type represents a different MAS architecture and the vertical bars show the standard deviations on the average calculated from the 30 variations of initial position of agents. The graphics are shown in pairs, showing respectively the absolute and normalized performances.

Fig. 3 and Fig. 4 show the graph idleness measures (in y-axis), and the corresponding normalized value, in Map A respectively, as the number of agents grows (x-axis).

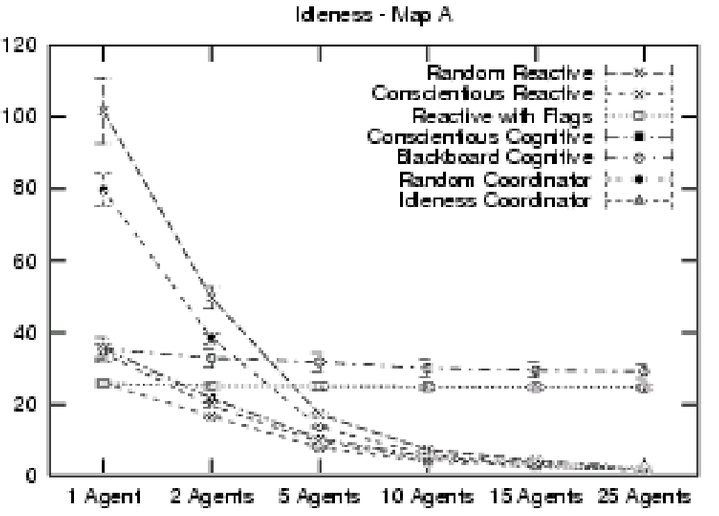

Fig. 3. Graph representing Idleness for Map A.

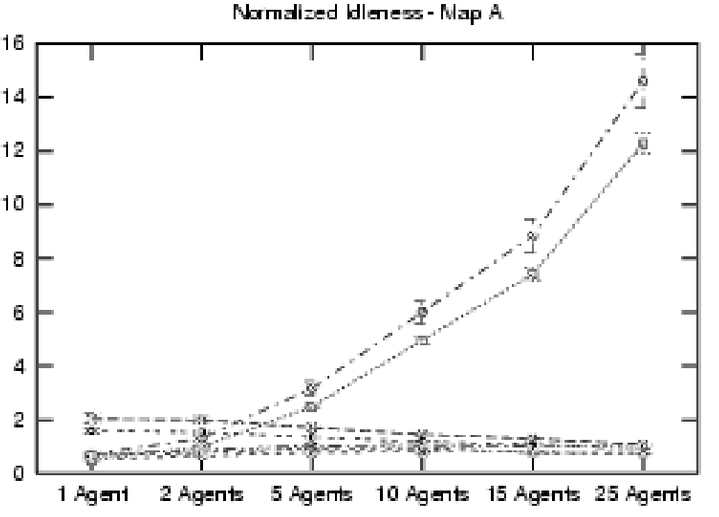

Fig. 4. Graph representing Normalized Idleness for Map A.

Fig. 5 and Fig. 6 show the graph idleness measures (in y-axis), and the corresponding normalized value, in Map B respectively, as the number of agents grows (x-axis).

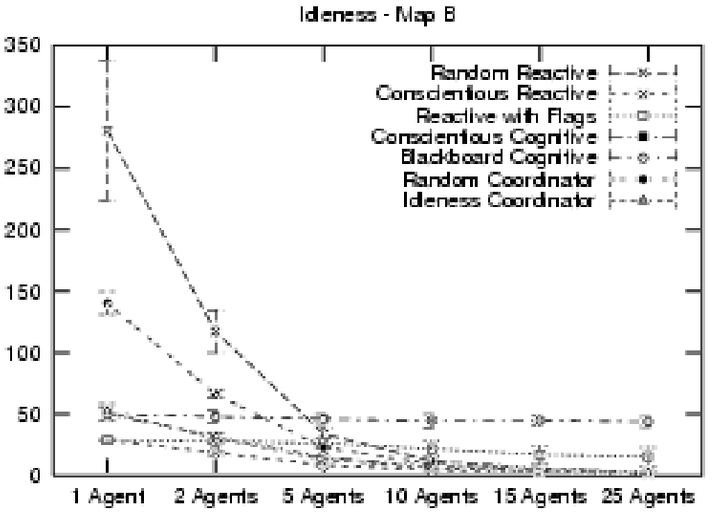

Fig. 5. Graph Idleness for Map B.

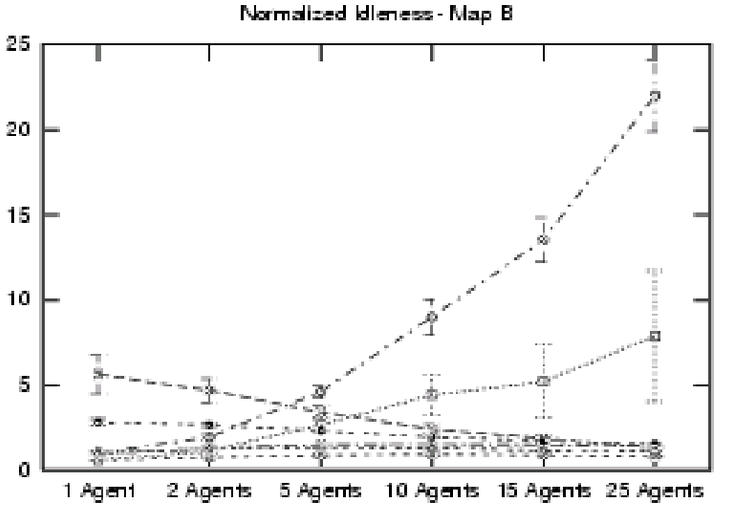

Fig. 6. Graph Normalized Idleness for Map B.

Similarly, Fig. 7 and Fig. 8 present the results of the graph worst idleness. The main difference in this case is that worst idleness is measured over the 3000 cycles, whereas (average) idleness does only consider the stable phase.

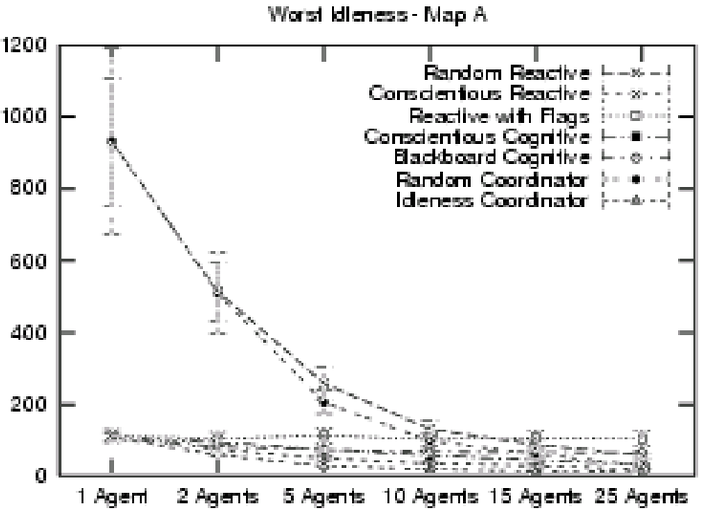

Fig. 7. Graph Worst Idleness in Map A.

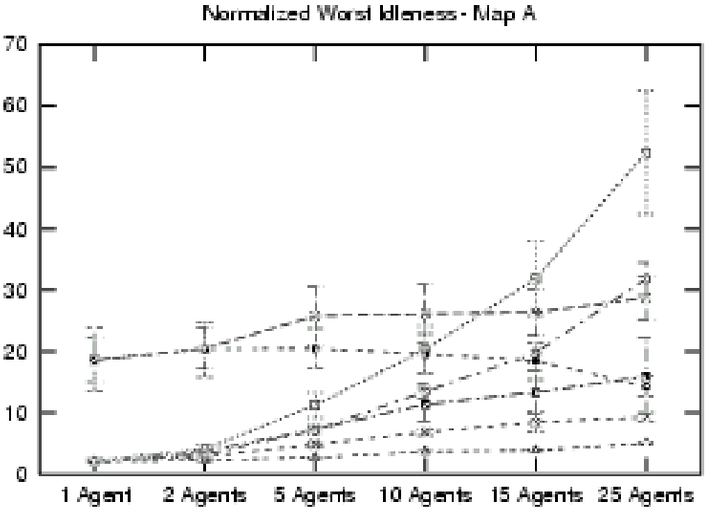

Fig. 8. Graph Normalized Worst Idleness in Map A.

Fig. 9 and Fig. 10 present the results of the graph worst idleness, in Map B, and the corresponding normalized value.

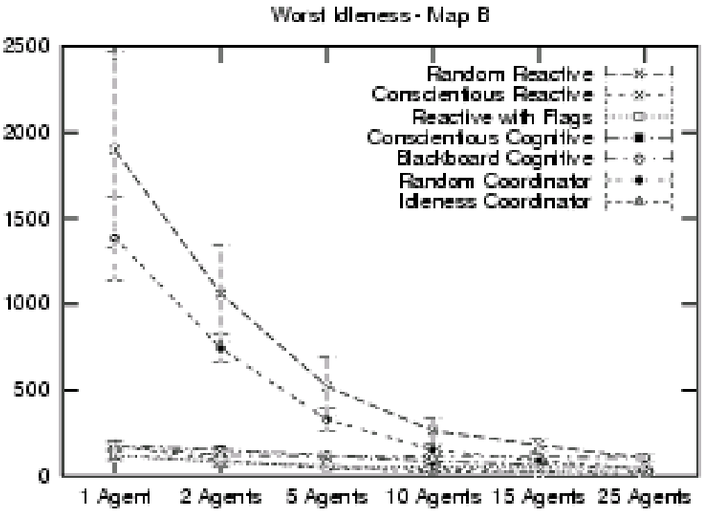

Fig. 9. Graph of Worst Idleness in Map B.

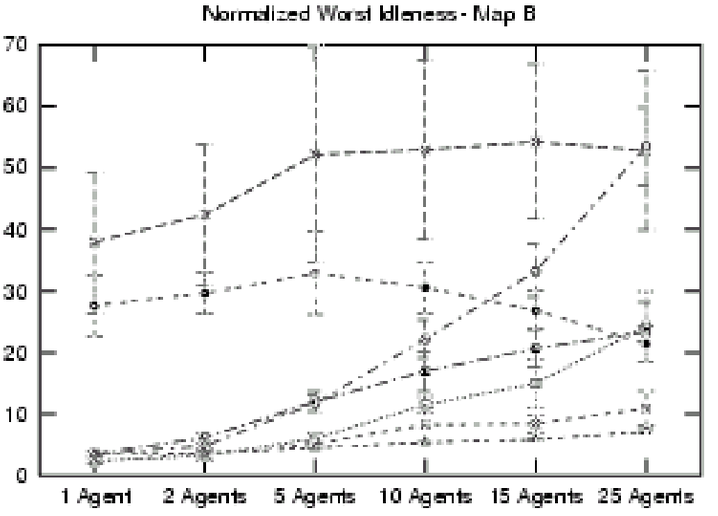

Fig. 10. Graph of Normalized Worst Idleness in Map B.

As opposed to previous graphics, Fig. 11 (resp. Fig. 12) plots the number of cycles (resp. normalized number of cycles) required for a complete exploration of the Map A.

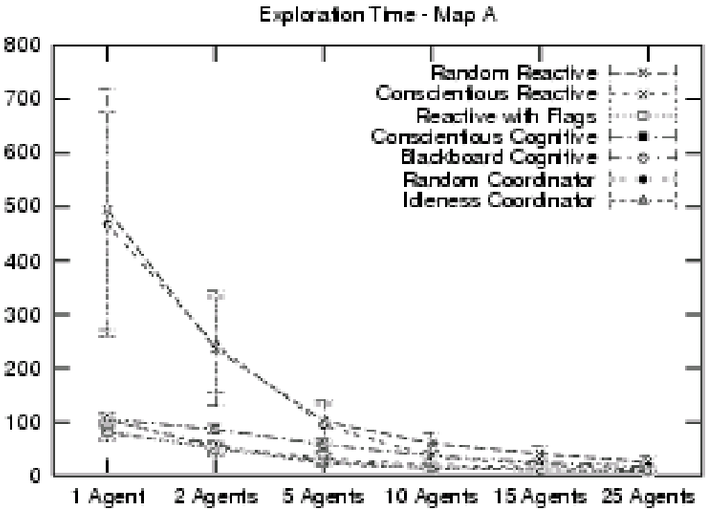

Fig. 11. Graph Exploration Time in Map A.

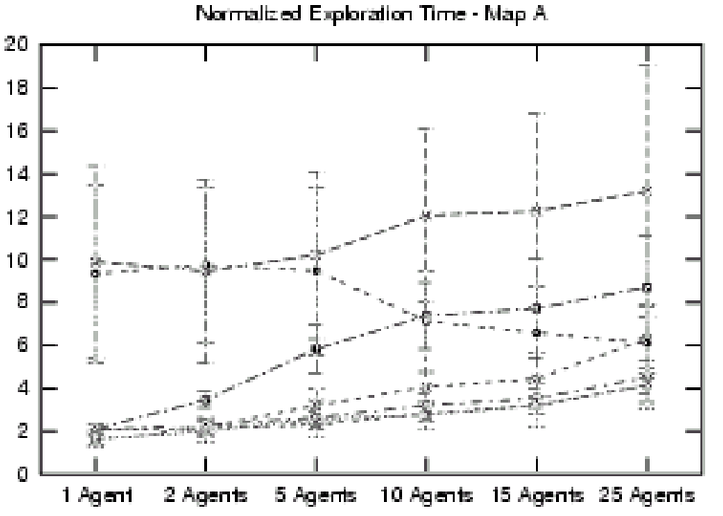

Fig. 12. Normalized Exploration Time in Map A.

As opposed to previous graphics, Fig. 13 (resp. Fig. 14) plots the number of cycles (resp. normalized number of cycles) required for a complete exploration of the Map B.

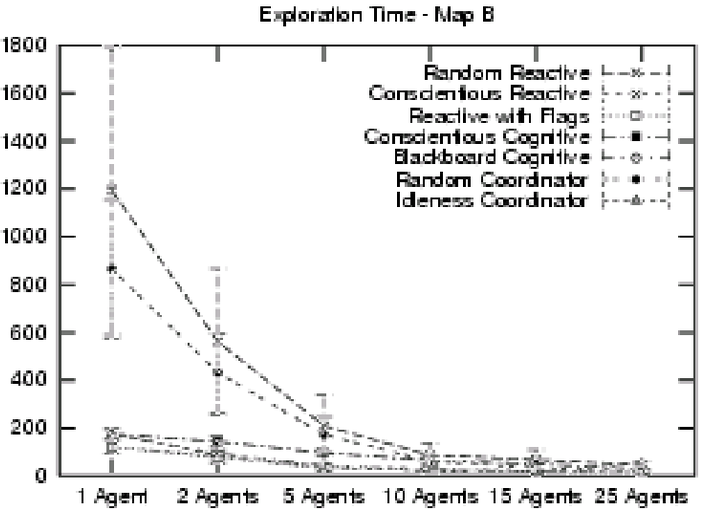

Fig. 13. Graph Exploration Time in Map B.

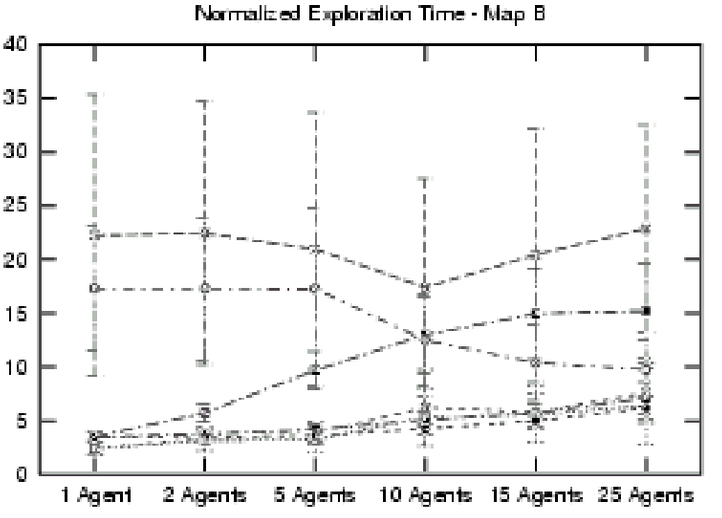

Fig. 14. Normalized Exploration Time in Map B.

4.2. Discussion

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.

Table 2. MAS acrchitecture groups

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

The *top group* obtained the best results in all metrics. The Conscientious Reactive Agent performance has been a little better that the other agents of this group, but this small difference has tended to zero as the population increased. 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 an expected behavior: every agent tends to go to the same places at the same moment. Consequently, the groups somehow behave as a single agent.

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, in all experiments, using all metrics. The *top group* has been less affected by the bottlenecks in Map B. 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.

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 discard. Moreover, according to the desired idleness, the number of agents can be determined (the ratio one agent for 10 nodes yields enough good results).

5 Conclusions

This work presents a many-fold contribution to the problem of multi-agent patrolling. First, a general characterization of the multi-agent patrolling problem is given, and its wide range of application in a variety of domains is pinpointed. Second, the method proposed for evaluating different MAS architectures for patrolling is generic and meant to be used as a basis for future analysis. Third, different MAS architectures for patrolling are suggested as well as a preliminary typology, according to some coordination parameters (this typology follows the same approach we have been using in other tasks [18]). Finally, this work furnishes some preliminary guidelines for MAS designers interested in patrolling tasks.

The simulator developed for the purpose of this study is available upon request for other researchers interested in experimenting with their own 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. In the same direction, we plan to use the exact distance between two nodes, instead of a unitary one, in order to use more realistic heuristics for choosing the next node. Finally, explicit communication between agents is another direction of exploration, which would give the opportunity to explore negotiations mechanisms for solving conflicts

References

1. Abate, Frank R.: The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers. Oxford Univ. Press. 1996
2. Andrade, R. de C., Macedo, H. T., Ramalho, G. L., and Ferraz, C. A. G.: Distributed Mobile Autonomous Agents in Network Management. Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications, 2001
3. Arkin, Ronald C.: Behavior-Based Robot Navigation for Extended Domains. Adaptive Behaviors. Fall 1992, vol. 1(2):201–225
4. Arthur, W. B.: Inductive Reasoning and Bounded Rationality (The El Farol Problem). American Economic Review (1994) 84: 406–411.
5. Balch, Tucker and Arkin, Ronald C.: Behavior-Based Formation Control for Multi-robot Teams. IEEE Transactions on Robot and Automation (1999) vol. XX
6. Cho J., Garcia-Molina, H.: Synchronizing a database to Improve Freshness. In Proceedings of 2000 ACM International Conference on Management of Data (SIGMOD), May 2000.
7. 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
8. Ferber, Jacques: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley (1999) 439–445.
9. Howland, Geoff: A Practical Guide to Building a Complete Game AI: Volume II. http://www.lupinegames.com/articles/prac_ai_2.html, 1999

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. RoboCup Rescue home page: http://www.r.cs.kobe-u.ac.jp/robocup-rescue/, 2001.
15. Russell, Stuart J. and Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall (1995) 796–808
16. Stout, Brian W.: Smart Moves: Intelligent Path-Finding. Game Developer (October/ November 1996) 28–35
17. 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.
18. 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
19. 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