A Two-Step Evolutionary and ACO Approach for Solving the Multi-Agent Patrolling Problem

Fabrice Lauri, Abderrafiâa Koukam

***Abstract*— Patrolling an environment involves a team of agents whose goal usually consists in continuously visiting its most relevant areas as frequently as possible. For such a task, agents have to coordinate their actions in order to achieve optimal performance. Current research that tackles this complex multi-agent problem usually defines the environment as a graph, so that a wide range of applications can be dealt with, from computer network management to computer games and vehicle routing. In this paper, we consider only the instances of the multi-agent patrolling problem where all the agents are located on the same starting node. These instances are often encountered in robotics applications, where e.g. drones start from the same area, disperse over it and finally patrol around distant locations. We introduce a new Ant Colony Optimization (ACO) algorithm that is combined with an Evolutionary Algorithm (EA) technique. The novel ACO algorithm uses several ant colonies that are engaged in a competition for finding out the best multi-agent patrolling strategy. The goal of the EA is to find the best set of distant nodes enabling each agent to disperse efficiently over the graph. Experimental results show that, irrespective of the number of the involved patrolling agents and for all the graphs evaluated, our two-step EA and ACO algorithm outperforms significantly and with efficiency the best techniques proposed in the literature since now.**

I. Introduction

Patrolling consists in *continuously* visiting the relevant areas of an environment, in order to efficiently supervise, control or protect it. An ant colony searching for food and gathering it, a group of postmen on their daily rounds, or a squad of marines securing an area are all examples of a patrol. Performing such a task requires for all the involved members an efficient coordination of their actions.

Most techniques that solve the multi-agent patrolling problem (MAPP) use a graph as the area to be patrolled. For these reason, these techniques can be used easily in numerous applications, ranging from network management [15] to the detection of enemy threats or the protection of cities in computer games [11].

The multi-agent patrolling problem has been rigourously addressed only recently [12], [1], [16], [2], [10]. In these works, many patrolling strategies have been devised and experimentally validated using common evaluation criteria [12]. They are based on approaches of different fields, such as heuristic laws enabling agents to better choose the next node to visit [12], negotiation mechanisms [1], reinforcement learning techniques [16], schemes based on graph theory [2] or ACO [10]. Most of these solutions yield good empirical results on graphs composed of less than eighty nodes and one hundred edges.

In this paper, we adopt the Ant Colony Optimization (ACO) coupled with an Evolutionary Algorithm (EA) as the solution approach to efficiently solve the multi-agent patrolling problem.

Two algorithms are employed in a two-step approach for the following reasons. In some applications, especially those relating to robotics (e.g. when a group of drones have to patrol), agents are usually brought together to a given place. Under this hypothesis, the patrolling task always begins with a preliminary phase where the agents spread out over the graph. Once this phase is finished, the agents, that are now as distant from each other as possible, start patrolling from their new locations. As we intend to implement our techniques on mobile robots, we decided to only deal with situations where all the agents are located at the same node at the initial time. The most distant nodes the agents have to head for in the preliminary phase are found thanks to an Evolutionary Algorithm.

Once the agents are as distant from each other as possible, begins the patrolling task. An ACO procedure is then used to determine the patrolling strategy of each agent. The novel ACO algorithm presented here uses several ant colonies that are involved in a competition for finding the best solution. Each colony tries to elaborate the best multi-agent patrolling strategy from the individual strategies found by the ants belonging to the same colony. Thus, in our ACO approach, each ant only find a partial solution to the problem. Besides, contrary to the other approaches in the literature, our ACO method directly optimize the *average idleness* of the graph. Using this criteria enables to yield the best patrolling strategies [3].

The applicability of ACO for the Multi-Agent Patrolling Problem (MAPP) is obvious. The problem is inherently graph-based, and clearly requires coordination and path-following behavior, tasks for which ants excel. Besides, as belonging to the family of the metaheuristics, an ACO algorithm can theoretically cope with any graph topology and any size of the agents' population, so that a large range of situations can be considered.

The remainder of this paper is organized as follows. Section II describes the commonly used framework of the patrolling problem and gives an overview of the related works. Section III presents the evolutionary algorithm that enables agents to disperse around a graph and the ACO algorithm that enables agents to patrol. Experimental results are shown in section IV and concluding remarks and future research directions are given in section V.

II. The patrolling problem

A. Mathematical framework

The patrolling problem is usually specified formally as follows [12], [2], [16]. The environment to patrol is reduced to a graph $G = (V, E)$. V stands for the strategical areas and E the ways of movement (or communication) between them. A cost c_{ij}, associated with each edge (i, j), measures the time required to reach the node j from the node i.

Let r agents bound to visit the areas defined in the graph G at regular intervals. Each agent is located at one of the nodes of V at the initial time.

Solving the patrolling problem consists in elaborating a multi-agent graph coverage strategy π. Such a strategy must optimize a given quality criterion. $\pi = \{\pi_1 \cdots \pi_r\}$ is made up of the r individual strategies π_i of each agent i. An individual strategy π_i is defined such that $\pi_i : \mathbb{N} \rightarrow V$, $\pi_i(j)$ denoting the j-th node visited by the agent i, with $\pi_i(j+1) = x$ iff $(\pi_i(j), x) \in E$.

It is commonly admitted that a relevant patrolling strategy is one that minimizes, for each node, the time span between two visits to the same node.

Several criteria have been devised in [12] in order to evaluate the quality of a multi-agent patrol strategy after T time steps (or *cycles*) of simulation. All of them are based on the notion of *instantaneous node idleness* (INI). The INI $I_t(i)$ of a node i at time t is the number of time steps this node remained unvisited. By convention, at the initial instant, $I_0(i) = 0$, $\forall i = 1, 2, \cdots, |V|$. At a given instant t, GI_t is the *instantaneous average graph idleness* (IGI), i.e.: $GI_t = \frac{1}{|V|} \sum_{i \in V} I_t(i)$. Similarly the *instantaneous worst graph idleness* WI_t is the highest INI encountered since t time steps of simulation, i.e.: $WI_t = max_{i \in V}\ I_t(i)$. A multi-agent patrol strategy π can be evaluated after T cycles of simulation using either the *average idleness* criterion AI_T or the *worst idleness* WI_T. The average idleness denotes the mean of the IGI over the T simulation cycles, i.e.: $AI_T = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{|V|} \sum_{i \in V} I_t(i)$. The *worst idleness* is the highest INI observed during the T-time steps of the simulation, given by the following equation: $WI_T = max_{t=1}^{T}\ max_{i \in V}\ I_t(i)$.

B. Overview of the related works

To our knowledge, [12], [11] are the pioneer works dealing with the patrolling problem. Several multi-agent architectures and the evaluation criteria explained in the previous paragraph were addressed in these papers. Each architecture was a combination of some parameters, such as the agent communication (allowed vs forbidden), the agent perception (local vs global), etc.

[1] improved the best architectures proposed in [11] by devising agents able to exchange messages freely and conduct negotiations about the nodes they have to visit.

Chevaleyre [2], [3] formulated the patrolling problem in terms of a combinatorial optimization problem. He first proved that a patrolling strategy with one agent could be obtained using an algorithm which solves a variant[1] of the *Traveling Salesman Problem*. He then studied three possible multi-agent patrolling strategies, and showed that they all were able to obtain a close to optimal strategy.

In [16], the agents are able to learn to patrol using the Reinforcement Learning (RL) framework. Each agent implements a *Markov Decision Process* (MDP) that is employed to know which action to perform in every environment state. Two architectures were proposed. In the most successful architecture, named *Gray-Box Learner Agents (GBLA*), agents can communicate their intention through the environment.

All of the previously described approaches were evaluated in [1] and were compared on six graph topologies with 5 and 15 agents. It has been shown that the single-cycle based strategy [2] gave the best results in all the configurations except one, whereas the two most efficient architectures proposed in [11], [12] yielded here the worst results. All the other schemes presented in [1] and [16] gave equivalent performances.

Finally, Lauri *et al.* [10] proposed an Ant Colony Optimization technique, named *GU/AA*, where all the agents start to patrol from the same initial node and come back to this node at the end of their cycle. This technique has proven its efficiency over *GBLA* for four out of the six evaluated graphs, for populations composed of 2 to 20 agents.

III. Two-step evolutionary and ACO approach to the patrolling problem

Recall that only the instances of the multi-agent patrolling problem where all the r patrolling agents are initially located on the same node will be dealt with the subsequent approach described in this section. Two questions have to be answered. The first one is: how can the agents efficiently spread out over the graph, or in other words, which node each agent has to head for to be as distant from each other as possible ? The second question is: how do they patrol from their new locations ?

The hybrid algorithm we propose here, named *EA-AD+GG-AA*, consists of two stages. In the first stage, all the agents have to spread out over the graph. For this stage, an Evolutionary Algorithm (EA) is performed to find the r most distant nodes in the graph representing the environment that the r agents have to patrol. The second stage stands for the patrolling task itself, and will be carried out by an ACO algorithm, named *GG-AA*.

The next subsections are organized as follows. The first two subsections recap the notions of classical evolutionary algorithms and describe the EA method for agent dispersion. The two last subsections will describe in an incremental manner the different parts of the ACO algorithm *GG-AA*.

A. Classical Evolutionary Algorithms

Classical evolutionary algorithms are basically methods for solving numerical optimization problems. Similar to most

[1] i.e. the *graphical-TSP* variant, in which the graph is not necessarily complete.

optimization techniques, evolutionary algorithms look for the best solution in a given search space by maximizing a gain function. The search is guided by a simulation of Darwin's natural selection scheme, that associates the diversity of species created by the random and the survival of the fittest individuals.

Typically, evolutionary algorithms start from an initial population of candidate solutions (or *individuals*) and they try to improve this population of solutions over a sequence of N_{IT} iterations in order to reach a population that contains better solutions. In the terminology of evolutionary algorithms, a solution is represented by one or more *chromosomes*. Each chromosome is represented by a vector of different *genes*. A gene stands for one of the decision parameters of the associated optimization problem. A solution s is characterized by a *fitness function* $fitness(s)$, that indicates the quality of adequacy of the solution s to the considered problem.

To generate a new population of solutions from a given population, standard evolutionary algorithms use three operators in each iteration, i.e. a *crossover* (or reproduction) operator, a *mutation* operator and a *selection* operator. The crossover operator can be seen as a way of providing an exchange of information between candidate solutions. Once the children have been generated by the crossover operator, they can be subjected to mutations. The idea behind the mutation procedure is the introduction of some variations within the population. Finally, the selection operator enables the fittest individuals of the current population to survive. These individuals then constitute the population of the next generation.

The specific characteristics of the evolutionary algorithms we used for the agent dispersion problem are detailed in the next section.

B. Agent dispersion with an Evolutionary Algorithm

Let n be the number of nodes of the graph to patrol and r the number of patrolling agents. The problem of agent dispersion then consists in finding the list l^* composed of the r most distant nodes that maximizes the sum of the minimum distances between the r nodes, i.e.:

$$l^* = \underset{l \in L}{\operatorname{argmax}}\ d(l) \quad (1)$$

such that:

$$d(l) = \sum_{i \in l} \min_{j \in l, i \neq j} c(i,j) \quad (2)$$

where $c(i,j)$ is the cost standing for the time required to reach the node j from the node i and L is the set of *feasible solutions*, i.e. the set of lists composed of exactly r nodes.

For this problem, the number of feasible solutions equals the binomial coefficient $C(n,r)$, which for $n = 50$ and $r = 20$ equals 4.7×10^{13}.

Using an Evolutionary Algorithm for tackling this NP-hard problem enables to find rapidly good solutions. In the EA for agent dispersion, named *EA-AD*, each individual is represented by a list l of r node numbers such that $l = (n_1, n_2, \ldots, n_r)$ where $n_{i+1} > n_i$ for all $i = 1, 2, \ldots, r-1$.

The three evolutionary operators, i.e. the crossover operator, the mutation operator and the selection operators defined in *EA-AD* are described below.

1) Crossover operator:

It consists in:

- selecting pairs of different parents among the individuals of the current population,
- determining the genes of the children from those of the parents.

The first step is commonly performed by using a roulette wheel to select parents [13]. Both parents are selected with a probability proportional to their fitness function. The higher the value of its fitness function, the more likely the corresponding individual will be selected as a parent. This means that each individual can be selected several times. If N_I is the number of individuals in the current population, then $N_I/2$ pairs of parents are defined in this step.

Such a selection strategy, consisting in selecting two different parents, avoids to converge too rapidly towards local solutions. On the contrary, an elitist strategy selects for each pair of parents the best current individual and this one is merged with another individual selected with a roulette wheel.

Once all the $N_I/2$ pairs of parents have been defined, the genes of the parents of each pair are combined to generate two offsprings. Here, the gene combination consists in using an interpolation factor i_f whose value is generated randomly within a given range. The values of the genes of both offsprings are obtained by a linear combination of the values of the genes of the parents by using this interpolation factor.

For instance, two parent vectors x_1 and x_2 may produce two offspring vectors y_1 and y_2, such that:

$$y_1 = i_f\, x_1 + (1 - i_f)\, x_2$$
$$y_2 = i_f\, x_2 + (1 - i_f)\, x_1$$

Notice that this crossover operator guarantees that every offspring vector is an ordered list of node numbers.

2) Mutation operator:

It consists in carrying out the following two operations for every gene of the children created during the reproduction step. Let v^k be the vector representing the child k and v_i^k the value of the gene i of the child k. Firstly, a random number $r \in [0;1[$ is generated. Secondly, if $r < p_m$, p_m being the mutation probability, then the new value v_i^k of the gene i belonging to the child k is given by the following algorithm:

Mutation operation on a gene in *EA-AD*

1: **repeat**
2: $v_i^k \leftarrow random(|V|)$
3: **until** the value v_i^k appears only once in v^k
4: Reorganize the vector v^k such that $v_{i+1}^k > v_i^k\ \forall i$

Recall that $|V|$ is the number of nodes of the graph to patrol. Note that a reorganization stage is here mandatory in case the ordered list of node numbers has been disturbed by the mutation.

3) Selection operator:

In *EA-AD*, the N_I parents of the population of the next generation are selected from the current population by keeping the N_I most individuals among the parents and the generated children. This strategy of selection avoids to reach local solutions in an early stage of the search [13].

C. Structure of the ACO algorithm used for agent patrolling

Once the agents have been dispersed over the graph, they start to patrol from their new locations. The ACO algorithm we propose now for determining the multi-agent patrolling strategy that agents have to follow are based on the following principles.

As has been stated in [2], every instance of the single-agent patrolling problem can be reduced to an instance of the *graphical*-TSP (GTSP) [14]. A salesman in GTSP has to find a closed walk in a connected graph G, which is not necessarily complete. The salesman may only use edges of G, but is allowed to visit a city or to cross an edge more than once. Finding the smallest closed-path covering all the nodes in G will result in the best possible single-agent patrolling strategy.

It has been shown in [14] that a GTSP can be easily transformed to a TSP as follows. Let $G_n = (V_n, E_n)$ be the complete graph on n nodes. For every pair (i, j) of nodes, the shortest path from node i to j in the graph G is computed. The weight of edge (i, j) in G_n is then set to be equal to the length d_{ij} of this path.

Once a TSP solution has been found, the shortest tour in G_n can finally be transformed to a shortest closed walk in G visiting all the nodes.

A practical method for solving the single-agent patrolling problem (i.e. the GTSP) would then consists in using the ant-cycle algorithm of Colorni *et al.* for dealing with TSP and A^* [8], [9] for computing the paths between all the pairs of nodes of G.

From this straightforward algorithm that can efficiently tackle the single-agent patrolling problem, we have elaborated a generalization to the multi-agent patrolling problem.

Let us emphasize first that solving the TSP with the ACO approach consists in putting m ants in competition, each ant "working" for the single salesman and trying to find the shortest tour that visits all the cities (nodes) of a graph only once.

For the MAPP, each of the r patrolling agents i want to find an individual strategy π_i (i.e. the list of nodes it has to visit) such that the multi-agent patrolling strategy $\pi = \{\pi_1, \pi_2, \ldots, \pi_r\}$ optimizes a given quality measure. Using the preceding metaphor of ants working for a salesman and supposing m ant colonies will be used, then m ants (one ant of each colony) will work for one agent. Hence in our approach, each ant elaborates an individual strategy and each colony build a solution standing for a multi-agent patrolling strategy.

Let $\{a_{i,1}, a_{i,2}, \ldots, a_{i,m}\}$ be the set of the m ants working for agent i, $\forall i \in [1; r]$. Since a good patrolling strategy requires that agents have to coordinate their actions and share between them the nodes they have to visit, so will do the ants working for them. Organizing them into colonies will made them collaborative. Then, ants $a_{1,j}, a_{2,j}, \ldots, a_{r,j}$ are grouped together into a colony j. Thus ants belonging to the same colony will work together in order to find out the best individual patrolling strategies, whereas the m ant colonies will be engaged in a competition, with each one of them trying to determine the best multi-agent patrolling strategy.

Inside a colony, an ant goes from one unvisited node to another adjacent unvisited node. Each ant k of the colony l record the list of the nodes it has already visited in $tabu_{k,l}$, and knows the already visited nodes of every other ant of the same colony. An ant selects the next node to visit according to the probability $p_{ij}^{k,l}$. The higher $p_{ij}^{k,l}$, the more likely the node j will be visited. In *GG-AA*, the probability of selection of a node by the ant k of the colony l is given by:

$$p_{ij}^{k,l} = \begin{cases} \frac{[\tau_{ij}(T)]^{\alpha}\ [\eta_{ij}]^{\beta}}{\sum_{u \in allowed_l} [\tau_{iu}(T)]^{\alpha}\ [\eta_{iu}]^{\beta}} & \text{if } j \in allowed_l \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

where V is the set of graph nodes, $allowed_l = \{V - \sum_{i=1}^{r} tabu_{i,l}\}$ is the set of the unvisited nodes of colony l, $\tau_{ij}(T)$ is the pheromone intensity on edge (i, j) at cycle T, $\eta_{ij} = 1/c_{ij}$ is the visibility of node j from node i and α and β are parameters that control the relative importance of pheromone intensity and visibility.

For the pheromone-updating rule, we adopt that used by Colorni *et al.* [4], [5], [7] in their ant-cycle algorithm. Thus on each algorithm cycle, the pheromone intensity on each graph edge evaporates a little, and every ant deposits, on each edge it traverses during its tour, a quantity of pheromone that depends on the tour length. In *GG-AA*, all the ants deposit their pheromones on the edges of a single graph.

It has to be emphasized that the objective function used to measure the quality of one multi-agent patrolling strategy generated by an ant colony is the average idleness of the graph (equation II-A). Using this criteria ensures to find the most effective patrolling strategies. The appendix gives the proof that using the sum of the tour lengths of each ant of a colony as the objective function to minimize, as is done in [10], is not justified from a mathematical point of view, since it does not guarantee to produce multi-agent patrolling strategies that minimize the worst idleness of the graph when simulated.

The pheromone intensity is updated as follows:

$$\tau_{ij}(T+1) = \rho\ \tau_{ij}(T) + \Delta\tau_{ij} \quad (4)$$

where ρ is the evaporation coefficient, $\tau_{ij}(T+1)$ and $\tau_{ij}(T)$ are the pheromone intensities on edge (i, j) at cycle $T+1$ and cycle T, respectively. $\Delta\tau_{ij}$ is the pheromone quantity

deposited on the edge (i, j) by all the ants of all the colonies in this cycle.

D. Formal description of EA-AD+GG-AA

Formally, the implemented ACO algorithm *GG-AA* consists in the following stages:

ACO algorithm for multi-agent patrolling: *GG-AA*

1: $T \leftarrow 0$ (T is the cycles counter)
2: **for** every edge (i, j) of G **do**
3: Set the pheromone quantities $\tau_{ij}(T) = c$ (or $\tau^k_{ij}(T) = c$, $\forall k$)
4: Set $\Delta\tau_{ij} = 0$ (or $\Delta\tau^k_{ij} = 0$, $\forall k$).
5: **end for**
6: Place the r ants of each colony on the starting node of the corresponding agents.
7: **for** each colony l **do**
8: **for** every ant k **do**
9: Reset $tabu_{k,l}$ and place the starting node of ant k in $tabu_{k,l}(0)$.
10: **end for**
11: **end for**
12: **for** each colony l **do**
13: $s \leftarrow 0$
14: **repeat**
15: $s \leftarrow s + 1$
16: **for** each ant k **do**
17: Choose the node j to move to, with probability $p^{k,l}_{ij}$ given by equation 3
18: Move the ant k at node j and insert node j in $tabu_{k,l}(s)$
19: **end for**
20: **until** $\cup_{i\in[1;r]} tabu_{i,l}$ is full
21: **end for**
22: **for** each colony l **do**
23: **for** each ant k **do**
24: Place the ant k on the node indicated by $tabu_{k,l}(0)$
25: Build in $walk_{k,l}$ the closed-walk in G using the paths computed with A^*.
26: **end for**
27: Compute L_l as being the average idleness of the graph after T time steps of simulation.
28: **end for**
29: **for** every edge (i, j) of G **do**
30: **for** every ant k **do**
31: **for** each colony l **do**
32: $\sigma^{k,l}_{ij} = \begin{cases} \frac{Q}{L_l} & \text{if } (i,j) \in walk_{k,l} \\ 0 & \text{otherwise} \end{cases}$
33: $\Delta\tau_{ij} = \Delta\tau_{ij} + \sigma^{k,l}_{ij}$
34: **end for**
35: **end for**
36: **end for**
37: **for** every edge (i, j) of G **do**
38: Compute $\tau_{ij}(T + 1)$ (or $\tau^k_{ij}(T + 1)$, $\forall k$), according to equation 4.
39: **end for**
40: $T \leftarrow T + 1$
41: **for** every edge (i, j) of G **do**
42: Set $\Delta\tau_{ij} = 0$ (or $\Delta\tau^k_{ij} = 0$, $\forall k$).
43: **end for**
44: **if** $T < T_{MAX}$ **then**
45: Empty all the tour lists
46: Goto line 7
47: **else**
48: Display the walks of the r ants of the best colony
49: Stop
50: **end if**

IV. Experimental results

Multi-agent patrolling strategies were computed on six different graph topologies of various complexity (table I) with populations constituted from two to twenty agents (more precisely 2,3,4,5,6,7,8,9,10,15 and 20 agents).

TABLE I
Graph Topologies (the number of states was computed using *GBLA*)

Name	# nodes	# edges	Degree	Complexity (# states)
Circle	56	56	2	1345
Corridor	70	69	2	1641
Map B	50	69	5	59145
Grid	80	142	4	97633
Island	50	84	6	726337
Map A	50	106	7	16520713

A. Experimental protocol

The approach described in the section III is compared here with two techniques of the literature, i.e. *GBLA* [16] and *GU/AA* [10].

The learning of all the patrolling strategies using *GBLA* were carried out as follows. To obtain strategies as robust as possible, their training were divided into trials. At each trial, graph statistics (the node idlenesses and the average graph idleness) were set to zero, all the agents were placed at the same starting node and they learned to patrol during several iterations using *GBLA*. The starting node changed from one trial to the other. Thus, a total of 72 patrolling strategies (12 agents' population $\times$ 6 graph topologies) were trained.

We conducted some preliminary experiments in order to determine the learning parameters of *GBLA*, such as the number of trials, the number of iterations per trial, the learning rate α, the discount factor γ and the exploration probability ϵ. Table II shows the empirically determined values we used to perform the training of the agents' MDPs.

For the ACO techniques (*GU/AA* and *GG-AA*), we used the control parameters shown in table III. *GU/AA* is the technique proposed in [10] and *EA-AD+GG-AA* is the two-step technique described in section III. The number appearing

TABLE II

PARAMETERS USED IN THE TRAINING PHASE OF *GBLA*

# Trials	1000
# Iterations per Trial	10000
Learning Rate (α)	0.9
Discount Factor (γ)	0.9
Exploration Probability (in Q-Learning)	10 %

TABLE III

PARAMETERS USED BY THE ACO TECHNIQUES

# Working ants per agent (n)	10
Pheromone Initialization Constant (c)	0.01
Pheromone Quantity (Q)	100
Pheromone Relative Importance (α)	1
Visibility Relative Importance (β)	5
Evaporation Rate (ρ)	0.8

at the end of a technique indicates the number of iterations T_MAX used to find the solution.

For *EA-AD+GG-AA*, $p_m = 0.001$ and i_f was randomly generated within the range $[0; 2[$.

The next sections present the experimental results obtained with *GBLA*, *GU/AA* and *EA-AD+GG-AA* to assess the robustness of our multi-agent patrolling strategies, when the node where all the agents start to patrol is changed.

B. Comparison of GBLA and the ACO Algorithms

Figures 2, 3, 4, 5, 6 and 7 present the **average graph idleness** obtained after a multi-agent patrolling simulation using either strategies trained with *GBLA* or strategies computed by *GU/AA* or *EA-AD+GG-AA*.

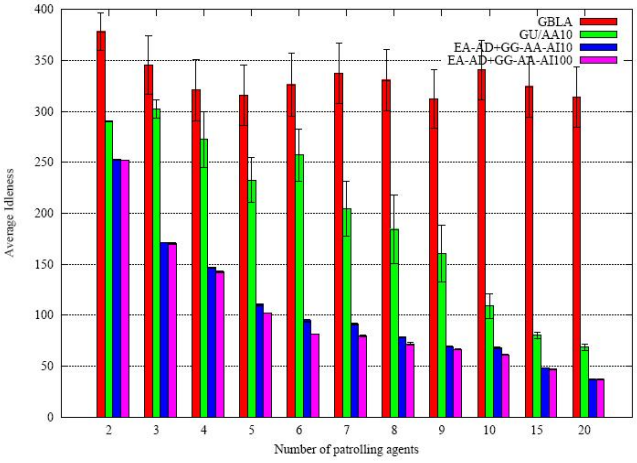

Fig. 1. Comparison results on Circle Map

Each patrolling strategy was evaluated 20 times by changing the starting node of the agents involved in the patrolling and by using 50 000 cycles of simulation. Thus, subsequent results represent the average graph idleness over the 20 runs. Confidence intervals indicated on figures were computed using a risk of 5%.

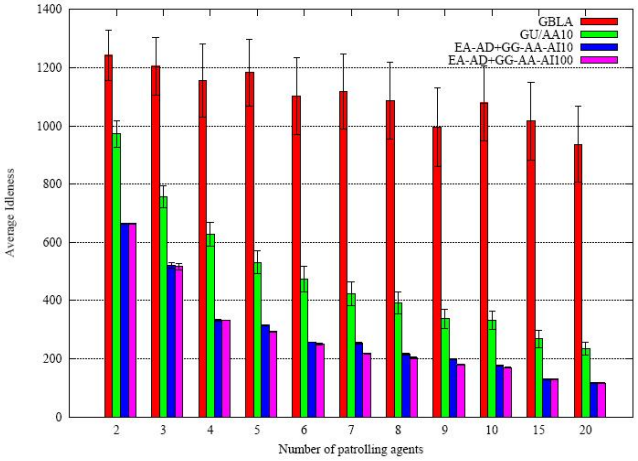

Fig. 2. Comparison results on Corridor Map

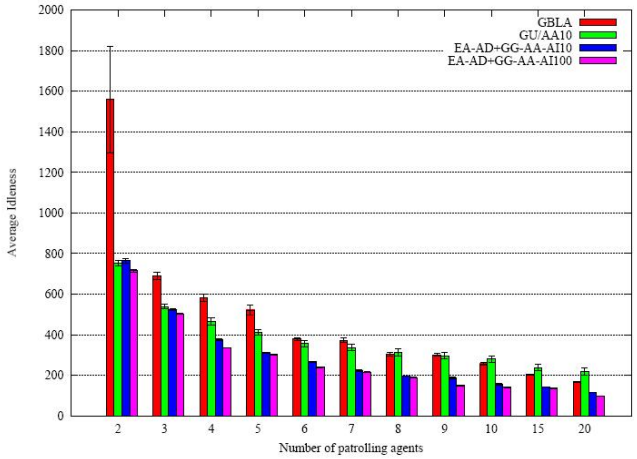

Fig. 3. Comparison results on Map B

One can already see that strategies computed by the ACO algorithms enable agents to coordinate their actions for any graph topology, since the average graph idleness decreases when the number of agents increases. It is not the case when using *GBLA*, where agents learned to coordinate their actions only on the Map A, the Map B, the Grid-shaped Map and the Island-shaped Map. Despite their lowest degree, learning to patrol on the Corridor-shaped graph and on the Circle-shaped graph thus seems to be more complicated.

The ACO algorithm *GU/AA* is significantly better than *GBLA* for four graphs (i.e. Corridor Map, Circle Map, Grid Map and Map A), irrespective of the number of the involved patrolling agents. This technique remains significantly better than *GBLA* on Map B and on the Island-shaped Map when at most 6 agents are patrolling. Then *GBLA* gives better results than *GU/AA* when more than 6 agents are patrolling.

The two-step EA+ACO algorithm, i.e. *EA-AD+GG-AA*, not only outperforms significantly *GBLA* on the six graph topologies and irrespective of the number of the involved patrolling agents, but it also outperforms *GU/AA* where

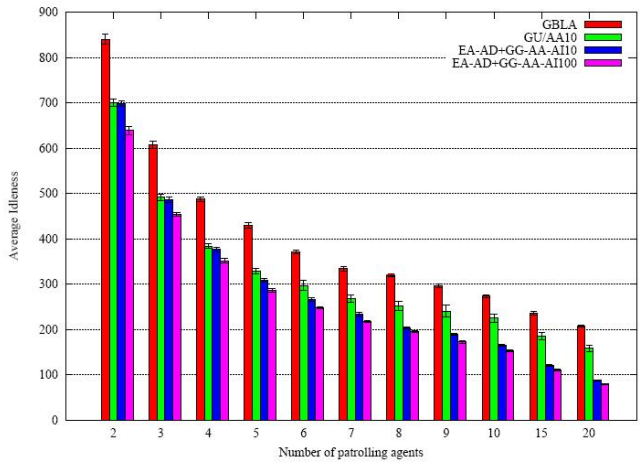

Fig. 4. Comparison results on Grid Map

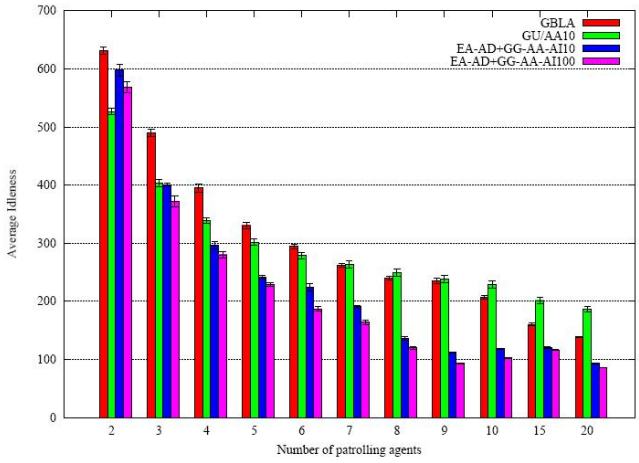

Fig. 5. Comparison results on Island Map

no dispersion phase is computed. Thus it seems that the dispersion phase is mandatory, especially when a lot of agents are patrolling. When more iterations are used, as is done in *EA-AD+GG-AA*100, the most distant nodes can be found with more precision, and the improvement is even better than using 10 iterations with *EA-AD+GG-AA*10.

By observing the agents' behaviors on our simulator, we have gathered the following remarks that can explain this results.

First, *GBLA* is able to create patrolling agents that can be classified into two different classes. The first class is composed of agents that are responsible of only one region of the graph: they patrol only nodes of that region during the whole simulation. The second class is made up of agents that cross from one region to another one, especially in order to visit the node that links several regions, thus avoiding to decrease performances. But sometimes, the path that agents take is not optimal, i.e. they can traverse an edge twice within a relatively small delay. These behaviors were only observed on the four more complex graphs (Map A, Map B,

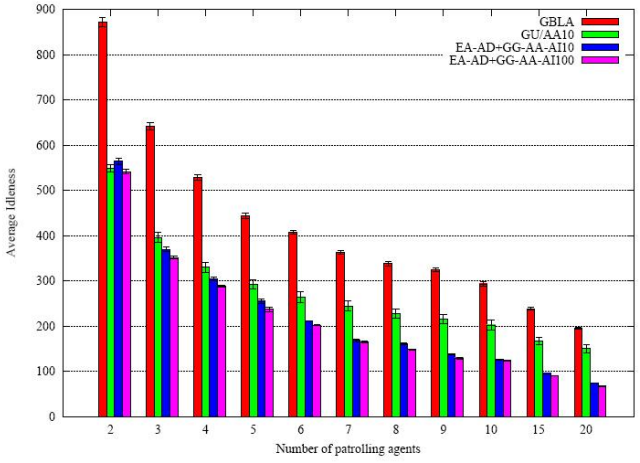

Fig. 6. Comparison results on Map A

Island and Grid). On the corridor-shaped graph and on the circle-shaped graph, all the agents follow the same patrolling policy: they all cross the graphs in the same direction and at the same time. This observation explains why the average graph idleness does not decrease when the agents' population grows.

Second, the ACO algorithms can deal with the patrolling problem at a global level, by enabling ants (and thus agents) to be able to go directly to nodes that are not adjacent to the current node in the patrolling graph G. Third, the patrolling strategies computed by *GU/AA* constraint the agents to start from a node and to come back at the starting node at the end of their tour. This strategy may be efficient only for some graph topologies.

With *EA-AD+GG-AA*, the agents are as distant from each other as possible after the dispersion phase, so they can more efficiently find which nodes to visit preferably, irrespective of the node they start from.

V. Conclusion and Future Works

We have described the application of Ant Colony Optimization (ACO) to the multi-agent patrolling problem, where agents are located at the same starting node. A novel two-step algorithm was presented and it was compared experimentally against the reinforcement learning approach (*GBLA*) of Santana [16] and the ACO approach (*GU/AA*) of Lauri [10]. The EA and ACO based algorithm proposed in this paper consists of two stages. These two steps correspond to the two-time strategy widely used in nature when starting to patrol an environment from the same area: first dispersion then patrolling. In the first stage of the algorithm, an EA is performed to find out the r most distant nodes in the graph representing the environment that the r agents have to patrol, so as to enable them to spread out over the graph. The second stage represents the patrolling task itself, and was carried out by an ACO algorithm. The ACO algorithm, named *GG-AA*, employ competitive colonies of ants: each colony tries to find out the best multi-agent patrolling strategy, and each ant of a

colony coordinates its action with the other ants of the same colony to elaborate an individual agent's patrolling tour as short as possible.

It has been showed experimentally in this paper that our two-step method significantly outperforms *GBLA* and *GU/AA*, irrespective of the number of the involved patrolling agents and for all the six evaluated graphs. Besides, it took no more than **two minutes** to compute one multi-agent patrolling strategy with *EA-AD+GG-AA*100 on a Pentium 4 2.8 GHz with 512 Mb (see table IV).

Several research direction can be explored to find better solutions to this complex multi-agent problem by using ACO techniques. First, some experiments are currently under progress to validate the proposed algorithm on larger graphs. Second, other ways of propagating and communicating information between ants of the same colony and between ants belonging to different colonies would surely yield better solutions. Finally, it is anticipated that integrating heuristic elements into our ACO approaches, as has been done by Dorigo and Gambardella [6] for the TSP, would improve both algorithm efficiency and solution quality.

TABLE IV
CPU TIMES

	***GBLA*(in minutes)**		***EA-AD+GG-AA*100 (in seconds)**	
# Agents	**Circle**	**Map A**	**Circle**	**Map A**
5	4 mn	39 mn	8 s	5 s
10	7 mn	168 mn	23 s	16 s
15	9 mn	300 mn	44 s	35 s
20	14 mn	520 mn	79 s	59 s

APPENDIX
MINIMIZING THE SUM OF THE TOUR LENGTHS DOES NOT INVOLVE TO MINIMIZE THE WORST IDLENESS OF THE GRAPH

Using the sum of the distances between the nodes visited by each ant of a colony as the objective function to minimize in ACO, as is done in [10], does not guarantee to produce multi-agent patrolling strategies that minimize the worst idleness of the graph when simulated.

A proof of this statement can be formulated as follows. Let π_1 and π_2 be two multi-agent patrolling strategies. For all i, $c(\pi_i)$ denotes the sum of the distances between the nodes visited by the patrolling agents following π_i. WI_{π_i} is the worst idleness of the graph over the T simulation steps when agents follow π_i for a long period of time, i.e. when $T \rightarrow \infty$. Using these notations, one has to prove that for all π_1 and π_2, it is equivalent to say that $c(\pi_1) < c(\pi_2)$ and that $WI_{\pi_1} < WI_{\pi_2}$, i.e. $c(\pi_1) < c(\pi_2) \Leftrightarrow WI_{\pi_1} < WI_{\pi_2}$.

First we have to prove that for all π_1 and π_2, $c(\pi_1) < c(\pi_2) \Rightarrow WI_{\pi_1} < WI_{\pi_2}$. But this logical sentence does not hold, since it exists at least one π_1 and one π_2 such that $c(\pi_1) < c(\pi_2) \Rightarrow WI_{\pi_1} > WI_{\pi_2}$.

Indeed, for example, let $G = (V, E)$ be an undirected graph composed of three nodes $V = \{1, 2, 3\}$, two edges $E = \{(1, 2), (2, 3)\}$ and such that the costs between edges are defined as follows: $c_{12} = c_{21} = 50$ and $c_{23} = c32 = 30$. Consider that G has to be patrolled by 2 agents. Let $\pi_1 = (\pi_{1,1}, \pi_{2,1})$ and $\pi_2 = (\pi_{1,2}, \pi_{2,2})$ be two multi-agent patrolling strategies, such that: $\pi_{1,1} = (1, 2, 1)$, $\pi_{2,1} = (2, 3, 2)$, $\pi_{1,2} = (1, 2, 3, 2, 1)$ and $\pi_{2,2} = (3, 2, 1, 2, 3)$. Then the tour length of $\pi_{1,1}$, $\pi_{2,1}$, $\pi_{1,2}$ and $\pi_{2,2}$ respectively equal $c(\pi_{1,1}) = 100$, $c(\pi_{2,1}) = 60$, $c(\pi_{1,2}) = 160$ and $c(\pi_{2,2}) = 160$. Thus $c(\pi_1) = 160$ and $c(\pi_2) = 320$. However, the worst idleness WI_{π_1} and WI_{π_2} of these two strategies respectively equal $WI_{\pi_1} = 100$ and $WI_{\pi_2} = 80$. So the best multi-agent patrolling strategy is here π_2, despite the fact that it has the greater tour length sum.

Hence it is not true to say that for all π_1 and π_2, $(c(\pi_1) < c(\pi_2) \Leftrightarrow WI_{\pi_1} < WI_{\pi_2})$. □

REFERENCES

[1] A. Almeida, G. Ramalho, and *al.* Recent Advances on Multi-Agent Patrolling. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, pages 474–483, 2004.

[2] Y. Chevaleyre. Theoretical Analysis of the Multi-Agent Patrolling Problem. In *International Joint Conference on Intelligent Agent Technology*, pages 302–308, 2004.

[3] Y. Chevaleyre. The Patrolling Problem. In *Annales du LAMSADE nř4, Paris-Dauphine University, France*, 2005. available in french at http://www.lamsade.dauphine.fr/~chevaley/papers/anna_patro.pdf.

[4] A. Colorni, M. Dorigo, and V. Maniezzo. Distributed Optimization by ant colonies. In *First European Conference on Artificial Life*, pages 134–142, 1991.

[5] A. Colorni, M. Dorigo, and V. Maniezzo. An Investigation of some properties of an ant algorithm. In *Parallel Problem Solving from Nature Conference*, pages 509–520, 1992.

[6] M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the travelling salesman problem. In *IEEE Transactions on Evolutionary Computation*, volume 1, pages 53–66, 1997.

[7] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. In *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, volume 26, pages 1–13, 1996.

[8] P.E. Hart, N.J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. In *IEEE Transactions on Systems Science and Cybernetics SSC4 (2)*, pages 100–107, 1968.

[9] P.E. Hart, N.J. Nilsson, and B. Raphael. Correction to 'A Formal Basis for the Heuristic Determination of Minimum Cost Paths'. In *SIGART Newsletter 37*, pages 28–29, 1972.

[10] F. Lauri and F. Charpillet. Ant Colony Optimization applied to the Multi-Agent Patrolling Problem. In *IEEE Swarm Intelligence Symposium, Indianapolis, Indiana, USA*, 2006.

[11] A. Machado, A. Almeida, and *al.* Multi-Agent Movement Coordination in Patrolling. In *Proceedings of the 3rd International Conference on Computer and Game*, 2002. available at http://www-poleia.lip6.fr/~machado/files/aicg_patrol_final.pdf.

[12] A. Machado, G. Ramalho, and *al.* Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures. In *Proceedings of the 3rd International Workshop on Multi-Agent Based Simulation*, pages 155–170, 2002.

[13] Z. Michalewicz. *Genetic Algorithm + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[14] G. Reinelt. The Traveling Salesman: Computational Solutions for TSP Applications. In *Lecture Notes in Computer Science 840, Springer Verlag*, 1994.

[15] E. Reuter and F. Baude. System and Network Management Itineraries for Mobile Agents. In *4th International Workshop on Mobile Agents for Telecommunications Applications*, pages 227–238, 2002.

[16] H. Santana, G. Ramalho, and *al.* Multi-Agent Patrolling with Reinforcement Learning. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1122–1129, 2004.