Ant Colony Optimization applied to the Multi-Agent Patrolling Problem

Fabrice Lauri and François Charpillet

LORIA-INRIA
B.P. 239, Cedex 54506, Vandœuvre-Lès-Nancy, France
{Fabrice.Lauri,Francois.Charpillet}@loria.fr

Abstract - Patrolling an environment involves a team of agents whose goal usually consists of 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 instances of the multi-agent patrolling problem where all agents are placed on the same starting node. These instances are often encountered in robotics applications, where e.g. drones start to patrol from the same area. The Ant Colony Optimization is adopted as the solution approach to these problem instances. Two novel ACO algorithms are proposed here, in which several ants' colonies are engaged in a competition for finding out the best multi-agent patrolling strategy. Experimental results show that, for three graph topologies out of the six which were evaluated, one of our ACO techniques, GU/AA, significantly outperforms the reinforcement learning technique proposed by Santana *et al.* (2004) (i.e. GBLA), irrespective of the number of the involved patrolling agents. For the other graph topologies, GU/AA approaches the results obtained with GBLA.

1 INTRODUCTION

A patrol is a mission involving a team of several individuals whose goal consists of *continuously* visiting the relevant areas of an environment, in order to efficiently supervise, control or protect it. A group of postmen on their daily rounds, an ant colony searching for food and gathering it, or a squad of marines securing an area are all examples of a patrol. Such a task requires that all of the involved members coordinate their actions to achieve optimal performances.
Techniques resolving the multi-agent patrolling problem (MAPP) can be used in numerous applications, ranging from network management [15], the rescue by robots of people in danger after a disaster [14, 10] to the detection of enemy threats or the protection of cities in computer games [11].

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

In this paper, we adopt the Ant Colony Optimization (ACO) as the solution approach to solve the multi-agent patrolling problem. This method draws its inspiration from the ability of natural ant colonies to cooperatively resolve numerous combinatorial problems [4, 7, 6]. The two novel ACO algorithms presented here extend the usual practice that ants belong to only one colony. Several ants' colonies are used: they all are engaged in a competition, with each one of them trying to construct the best multi-agent patrolling strategy, whereas ants belonging to a same colony work together in order to determine the best individual strategies.
We have chosen to use an ACO approach for solving the multi-agent patrolling problem for two main reasons. First, 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. Second, it can deal with situations where all of the agents are located at the same node at the initial time. In some applications indeed, especially those relating to robotics (e.g. when a group of drones have to patrol), agents are usually brought together to a given place, from where they start to patrol. Under this condition, the patrolling task always begins with a preliminary phase where the agents spread out in the graph. This step cannot be handled by the most efficient techniques that is based on Single Cycle [2, 1], since it requires that the agents are located at different nodes.

The remainder of this paper is organized as follows. Section 2 describes the commonly used framework of a patrolling problem and gives an overview of the related works. The reinforcement learning approach of Santana *et al.* (i.e. GBLA [16]) is also presented in this section. GBLA and our ACO algorithms both can deal with situations where agents start from the same node. For this reason they will be compared. Section 3 reviews the fundamental concepts of ACO. Section 4 presents our ACO algorithms, and experimental results are shown in section 5. Finally, concluding remarks and future research issues are given in section 6.

2 THE PATROLLING PROBLEM

2.1 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 representing the strategically relevant areas and E the safe ways of movement or communication between them. A cost c_{ij}, associated with each edge (i, j), measures the time required to go from node i to node j.

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.

Resolving the patrolling problem consists of 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). Similarly the *instantaneous worst graph idleness* WI_t is the highest INI encountered since t time steps of simulation.
A multi-agent patrol strategy π can be evaluated after T cycles of simulation using either the *average idleness* criterion AI_π or the *worst idleness* WI_π. The average idleness denotes the mean of the IGI over the T simulation cycles, whereas the *worst idleness* is the highest INI observed during the T-time steps of the simulation.
As emphasized by [3], the optimal strategy π is the one that minimizes the *worst idleness*, as $WI_\pi \geq AI_\pi$ for any strategy π.

2.2 OVERVIEW OF THE RELATED WORKS

To our knowledge, [12, 11] are the pioneer works dealing with the patrolling problem. In these papers, the authors considered respectively non-weighted graphs and graphs with real distances. Several multi-agent architectures and the evaluation criteria explained in the previous paragraph were addressed. Each architecture was a combination of some parameters, such as the agent communication (allowed vs forbidden), the agent perception (local vs global), the heuristic of selection of the next node (randomly, using the individual or shared idleness, the path length...), 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. Each agent randomly receives a set of nodes to visit, and uses a system of auctions to trade its undesirable nodes with the other agents. Each agent tries to keep nodes that are reachable within a reasonable amount of time. Negotiating thus allows them to reach a mutual agreement.
Chevaleyre [3, 2] 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 of the *Traveling Salesman Problem*[1]. 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. An action allows an agent to go to the adjacent nodes in the graph. An environment state stands for the minimal information required by an agent to precisely decide what to do. Two architectures were addressed: one in which agents cannot communicate, and one in which they can indirectly (through the environment) communicate their intention for the next action. The latter approach, named *Gray-Box Learner Agents (GBLA)*, was the most successful of these. As we compare it with our ACO technique, it is explained more in detail in the next section.

All of the previously described approaches were evaluated in [1] and were compared in twelve configurations (i.e. for 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 of the other schemes presented in [1] and [16] gave equivalent performances.

2.3 GRAY-BOX LEARNER AGENT'S APPROACH

Each agent in GBLA employs a *policy*, computed from an MDP, to perform the appropriate action in every encountered environmental situation.
Recall that an MDP can be defined as a 4-tuple $< S, A, T, R >$ where: S is the finite set of states of the environment; $A = \bigcup_{s \in S} A(s)$ is the finite set of actions enabling an agent to influence the environment state, $A(s)$ being the subset of actions available in state s; $T : S \times A \rightarrow S$ is the one-step dynamics of the environment, where $T(s, a)$ denotes the state which is always reached when action a is executed in state s; and $R : S \times A \rightarrow \mathbb{R}$ is a real-valued bounded reward function. Both functions T and R, representing the environment model, must satisfy the Markov property.
Then, a policy π for a given MDP is a mapping $\pi : S \rightarrow A$, where $\pi(s)$ is the action the agent performs at state s. When the environment model $< T, R >$ is not available, solving an MDP consists of finding the optimal policy π^* such that $Q^{\pi^*}(s, a) = max_\pi \; Q^\pi(s, a) \; \forall s \in S$ and $\forall a \in A(s)$. $Q^\pi(s, a)$ is the *action-value function* which represents the expected reward when starting from state s, performing action a and following policy π thereafter. When considering an infinite-horizon MDP, $Q^\pi(s, a)$ is generally defined as a discounted sum of the rewards obtained:

$$Q^\pi(s, a) = E_\pi \left\{ \sum_{k=0}^{+\infty} \gamma^k \, r_{t+k+1} \middle| s_t = s, a_t = a \right\}$$

where γ is the *discount rate* such that $0 \leq \gamma < 1$.
Therefore, an optimal policy π^* can be constructed greedily by considering that $\pi^*(s) = \arg\max_{a \in A(s)} \; Q^{\pi^*}(s, a)$. See e.g. [18] for more details on the RL issues.

In GBLA, each agent's MDP is learned using *Q-Learning*, which is the most widely used RL algorithm for solving an MDP when the environment model is not available. It consists of updating the value $Q(s, a)$ of a state-action pair at each time step t using the following equation:

$$Q_t(s, a) \leftarrow Q_{t-1}(s, a) + \alpha \left[R(s, a) + \gamma \, max_{a'} \, Q_t(s', a') \; - \; Q_{t-1}(s, a) \right]$$

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

where s' is the state reached when performing action a in state s and α is the *learning rate*. This algorithm is guaranteed to converge to the optimal action-value function Q^* under the condition that each state is visited an infinite number of times.

Considering that d stands for the graph degree and $|V|$ is the number of nodes of the graph, the state space S in GBLA is made up of the following components: (1) the node where the agent is ($|V|$ possible values) (2) the edge from which it came (d possible values) (3) the neighbor node which has the highest (worst) idleness (d possible values) (4) the neighbor node which has the lowest idleness (d possible values) and (5) the list of the adjacent nodes which are intended to be visited by other agents (2^d possible values). The cardinality of the action set was equal to the graph degree d, each action enabling an agent to reach an adjacent nodes.

3 ANT COLONY OPTIMIZATION

3.1 REAL ANTS

Ants in nature are relatively straightforward insects that are almost blind, have a very limited amount of memory and behave seemingly in a random way [4]. Despite these limited individual characteristics, ants is one of the only species that can be found all over the world, in every type of land environment. Their force comes owing to the fact that they are able to cooperate in order to perform complex tasks, such as: searching areas for food, finding the shortest paths to food, building and protecting their nest, forming bridges [17]. To achieve these tasks, ants communicate indirectly through environmental stimuli; this form of communication is named *stigmergy*. The ants deposit a volatile hormone (*pheromone*) to warn other ants that this position should be reached again. The pheromone thus stands for a signalling system, acting as a means of indirect communication, and leading to cooperative behaviors such as trail following [17].

3.2 ARTIFICIAL ANTS

The efficiency of the highly collaborative behavior of real ants has been used by Dorigo *et al.* to define an optimization tool, *Ant Colony Optimization (ACO)*, for solving hard combinatorial optimization problems [4, 7]. According to its creators, ACO is versatile, in the sense that it can be easily used in different variants of the same problem, or with minimal changes to different combinatorial optimization problems; is robust, because artificial ants are able to maintain their activities even if some of them is failing; is population-based, which allows positive feedback to be used as the primary search mecanism; and is flexible, meaning that an ants' colony is capable of adapting its behavior to changes of the environment.
For ACO to be successfully applied to a problem [5, 7], it must be formulated in terms of a search on a graph by some agents. Under this condition, the artificial ants in ACO can build a solution to the given problem, by exchanging information via pheromone deposited on the edges of the problem graph. While moving, ants modify the problem representation by adding collected information to the graph [6].
As ACO is not intended as a faithful simulation of ants in nature, but as an optimization tool, artificial ants working inside it have important differences from real ants. Indeed, artificial ants in ACO have memory for storing actions they have performed or to record locations they have been. Besides, they are not completely blind, but have sight based on distance, pheromone levels, etc; and their environment operates on discrete time.

3.3 ACO RELATED APPLICATIONS

Since its first development, ACO has been applied to a variety of problem areas. Only some of those related to the patrolling problem are mentioned here. These include the traveling salesman problem (or TSP) [4, 5, 7, 6], and the distributed covering by ant-robots [19, 20, 21].
Two of the earliest works on ACO are by Colorni *et al.* [4, 5]. Their first paper [4] both introduces the ACO paradigm and describes three algorithms for solving the TSP: ant-density, ant-quantity and ant-cycle. The most efficient algorithm, ant-cycle, found good tour solutions for 50-city and 75-city problems. Their second paper [5] investigates for their ant-cycle algorithm appropriate parameter settings, computational complexity and result quality against some TSP heuristics. In [7], the same authors compare ant-cycle against both tabu search and simulated annealing. In addition, the application of ACO to the asymmetric TSP (ATSP), the quadratic assignment problem and the job-shop scheduling is also considered. The ant-cycle is extended in [6] to create an algorithm, called ant colony system, for solving both TSP and ATSP. By including some problem-specific heuristics (such as *2-opt*), the authors report some of the best ever results for large ATSP instances.
Wagner *et al.* [19, 20, 21] consider the problem of robotic covering, assuming that a graph describes the parts of the region to be covered "as fast as possible" by a group of robots. Covering a graph then reduces to visiting all its edges. To achieve this task, they define several strategies, in terms of local behavior rules which have to be followed by the robots. A strategy is considered better if it covers the region faster, and the efficiency of the different strategies is evaluated by measuring the time from the start until the last yet unvisited node is reached. Robots are equipped with small memory (enabling them only to backtrack) and are able to leave or sense *traces*. In [19], an algorithm called *Vertex Ant Walk (VAW)* is presented, and its ability to cover the vertices of a graph by means of sensing the labels at neighboring vertices was proven. In [20], the authors describe the algorithm *Edge Ant Walk*, which also can cover a graph, but where agents needs to see the labels of outgoing edges only at its current location. EAW is used in [21] to patrol a graph. The *blanket time*, defined as the first time at which the ratio between the number of traversals of any two edges in the graph does not exceed two, was used as the evaluation criteria to measure the performance of the strategy EAW. It was proven that the blanket time of EAW do not exceed an upper bound.
It has to be noted that the evaluation criteria of a patrolling strategy, as well as the assumptions made on the characteristics of the robots and of a patrolling strategy in Wagner *et al.*'s works thus strongly differ from those described in section 2, and on which are based our ACO patrolling algorithm.

3.4 ACO FOR THE TSP

As our algorithm is strongly inspired by Colorni *et al.*'s ant-cycle algorithm, we provide here a brief informal description of it.
In each cycle of the algorithm, ants are located on different graph nodes (cities) [2] Then, for each time step, ants

[2] The authors experimentally showed that the performance of the algorithm was better when as many ants as nodes were used and when ants were placed on different nodes.

choose probabilistically which node to move to nextly, according to both the pheromone concentration and length of the links they are considering crossing: links with more pheromone and shorter distances are preferred. Each ant visits only the nodes it has not been visited before (by using a *tabu list*), until it has followed a complete tour. Once all the ants have completed their tours, pheromone is deposited, for each ant, on each link of the ant's tour. The pheromone deposited is greater on shorter tours; it evaporates a little on all links. The cycle is then repeated, with all ants starting from their original nodes. The best tour ever found after a certain number of cycles is kept as the final solution.

4 APPLYING ACO TO THE MAPP

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.

We have developed two variants on our ACO algorithm: global update using all ants' pheromone (GU/AA) and global update using the agent's working ants pheromone (GU/AWA). Before being described in details in the next sections, both are based on the same basic structure, as follows.

4.1 BASIC ALGORITHM

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) [13]. Recall that 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 nodes in G will result in the best possible single-agent patrolling strategy.
It has been shown in [13] 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 ij 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 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 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 same salesman who wants to follow the shortest tour that visits all the cities (nodes) of a graph only once.
In 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, here we will have m ants that will work for one agent. 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 $a_{j,1}, a_{j,2}, \ldots, a_{j,r}$ are grouped together into a colony j. So, ants belonging to the same colony will work together in order to find out the best individual patrolling strategies, whereas the m ants' colonies will be engaged in a competition, with each one of them trying to construct 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. The definition of this probability depends on the variant of the algorithm and will be given in the corresponding sections (see sections 4.2 and 4.3).
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. Whereas in the GU/AA variant, the pheromone is deposited on the edges of the patrolling graph, we consider in the GU/AWA variant that each of the r agent has a copy of the patrolling graph, on which its m working ants can deposit pheromone. Sections 4.2 and 4.3 describe in more details the pheromone-updating for GU/AA and GU/AWA, respectively.

4.2 GLOBAL UPDATE WITH ALL ANTS' PHEROMONE (GU/AA)

In GU/AA, the probability of selection of a node by the ant k of the colony l is given by:

$$p_{ij}^{k,l} = \begin{cases} \frac{[\tau_{ij}(T)]^\alpha \, [\eta_{ij}]^\beta}{\sum_{u \in allowed_{k,l}} [\tau_{iu}(T)]^\alpha \, [\eta_{iu}]^\beta} & \text{if } j \in allowed_{k,l} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

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

$$\tau_{ij}(T + 1) = \rho \, \tau_{ij}(T) + \Delta\tau_{ij} \quad (2)$$

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 colonies in this cycle.

4.3 GLOBAL UPDATE WITH AGENTS' WORKING ANTS (GU/AWA)

In GU/AWA, we consider that each agent has a copy of the patrolling graph, on which its m working ants can deposit pheromone. Thus the probability of selection of a node by the ant k of the colony l is given by:

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

where $\tau_{ij}^k(T)$ is the pheromone intensity on edge (i,j) at cycle T for the graph of agent k.
The pheromone intensity is updated as follows:

$$\tau_{ij}^k(T+1) = \rho\, \tau_{ij}^k(T) + \Delta\tau_{ij}^k \qquad (4)$$

where $\Delta\tau_{ij}^k$ is the pheromone quantity deposited by all of the m ants working for agent k on the edge (i,j).
This variant suggests that the ants working for different agents will not interfere between them.

4.4 FORMAL DESCRIPTION OF THE ACO ALGORITHMS

Formally GU/AA and GU/AWA thus consists in the following stages:

1. Set $T = 0$ (T is the cycles counter)
   For every edge (i,j) of G
   Set the pheromone quantities $\tau_{ij}(T) = c$ (or $\tau_{ij}^k(T) = c$, $\forall k$)
   Set $\Delta\tau_{ij} = 0$ (or $\Delta\tau_{ij}^k = 0$, $\forall k$).
   Place the r ants of each colony on the starting node of the corresponding agents.
2. For each colony l
   For every ant k
   Reset $tabu_{k,l}$ and place the starting node of ant k in $tabu_{k,l}(0)$.
3. For every colony l
   Set s = 0
   Repeat until $\cup_{i\in[1;r]}\, tabu_{i,l}$ is full
   Set s = s+1
   For each ant k
   Choose the node j to move to, with probability $p_{ij}^{k,l}$ given by equation (1) or (3)
   Move the ant k at node j and insert node j in $tabu_{k,l}(s)$
4. For every colony l
   For each ant k
   Place the ant k on the node indicated by $tabu_{k,l}(0)$
   Compute the length $L_{k,l}$ of the tour traversed by ant k, using the distance of the paths computed with A^*.
   Build in $walk_{k,l}$ the closed-walk in G using the paths computed with A^*.
   For every edge (i,j) of G
   For every ant k
   $\Delta\tau_{ij}^k = 0$ $\hookleftarrow$ **only in GU/AWA**
   For each colony l
   $\sigma_{ij}^{k,l} = \begin{cases} \frac{Q}{L_{k,l}} & \text{if } (i,j) \in \text{walk described by } walk_{k,l} \\ 0 & \text{otherwise} \end{cases}$
   $\Delta\tau_{ij} = \Delta\tau_{ij} + \sigma_{ij}^{k,l}$ $\hookleftarrow$ **only in GU/AA**
   $\Delta\tau_{ij}^k = \Delta\tau_{ij}^k + \sigma_{ij}^{k,l}$ $\hookleftarrow$ **only in GU/AWA**
5. For every edge (i,j) of G
   Compute $\tau_{ij}(T+1)$ (or $\tau_{ij}^k(T+1)$, $\forall k$), according to equation (2) or (4).
   Set $T = T + 1$
   For every edge (i,j) of G
   Set $\Delta\tau_{ij} = 0$ (or $\Delta\tau_{ij}^k = 0$, $\forall k$).
6. If($T < T_{MAX}$) Then
   Empty all tour lists
   Goto step 2
   Else
   Display the walks of the r ants of the best colony, i.e. the colony for which the sum of tour length of all ants is the lowest.
   Stop
   Endif

5 EXPERIMENTAL RESULTS

Multi-agent patrolling strategies were computed on six different graph topologies of various complexity (table 1) with a population constituted from one to twenty agents (more precisely 1,2,3,4,5,6,7,8,9,10,15 and 20 agents).

TABLE 1. Graph Topologies (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

We have developed a simulator facilitating the implementation and evaluation of any multi-agent behavior that could be exhibited in a three-dimensional environment. We used it (figure 1) to evaluate our multi-agent patrolling strategies.

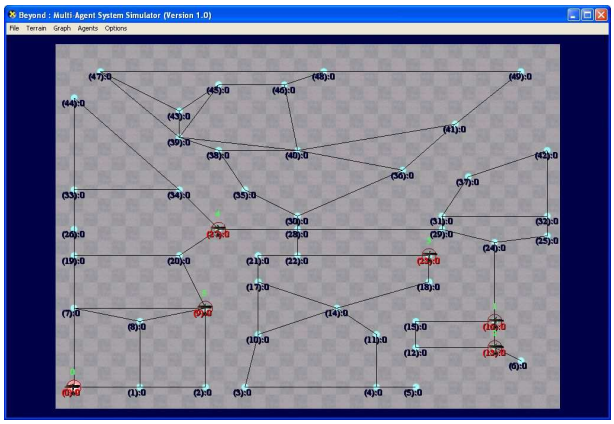

Figure 1. Our multi-agent system simulator

5.1 EXPERIMENTAL PROTOCOL

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 iteration per trial, the learning rate α, the discount factor γ and the exploration probability ϵ. Table 2 shows the empirically determined values we used to perform the training of the agents' MDPs.
For the ACO techniques, we used the control parameters shown in table 3.

In the next sections are presented the experiments results we obtained with GBLA and our ACO algorithms to assess the robustness of our multi-agent patrolling strategies, when the node where all agents start to patrol is changed.

TABLE 2. 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 3. Parameters used in the ACO techniques

# Cycles (T_{MAX})	10
# 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

5.2 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 our ACO algorithms.
Each patrolling strategies was evaluated 20 times by changing the starting node of agents and by using 50000 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%.

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 algorithms are significantly better than GBLA for three graphs (i.e. Corridor, Circle and Map A). For the Grid-shaped Map, only GU/AA is significantly better than GBLA, irrespective of the number of the involved patrolling agents. GU/AA 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 both ACO techniques when more than 6 agents are patrolling.

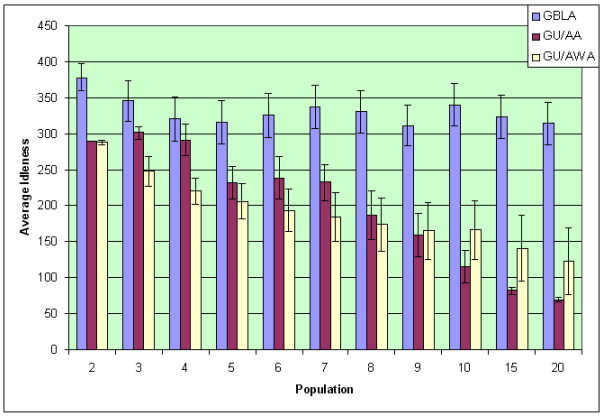

Figure 2. Comparison results on Circle Map

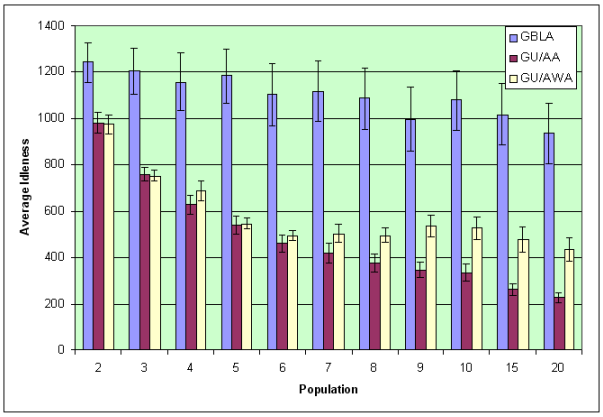

Figure 3. Comparison results on Corridor Map

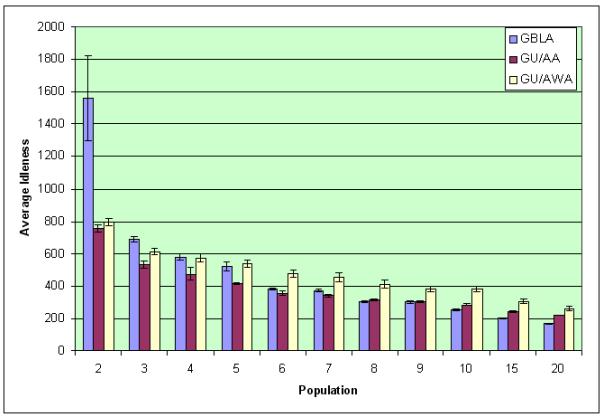

Figure 4. Comparison results on Map B

By observing the agents' behavior 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

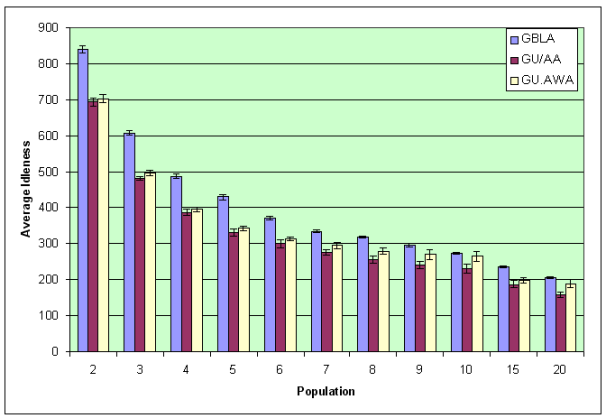

Figure 5. Comparison results on Grid Map

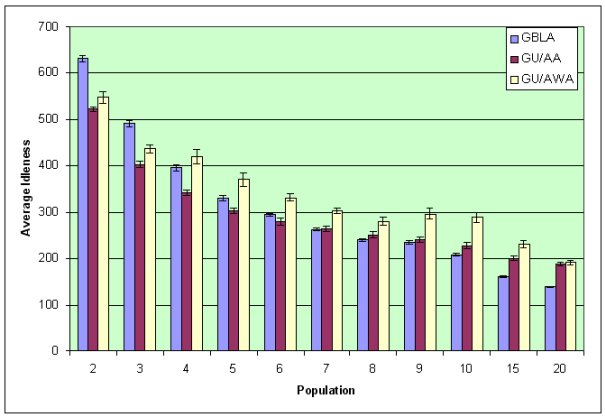

Figure 6. Comparison results on Island Map

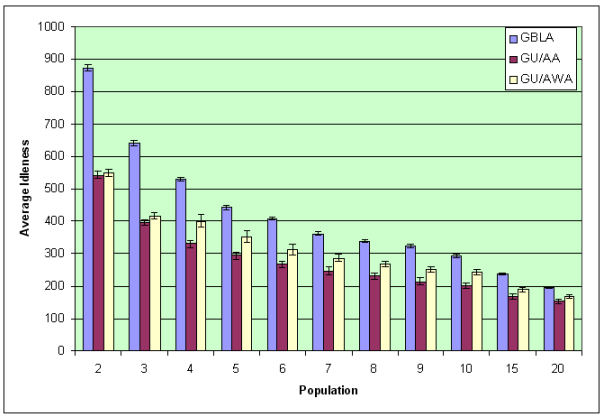

Figure 7. Comparison results on Map A

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

Second, GU/AA and GU/AWA can both 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 and GU/AWA 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.

6 CONCLUDING REMARKS 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. Two novel algorithm variants were presented and they were compared experimentally against the reinforcement learning approach of Santana *et al.* [16]. These two ACO algorithms employ competitive colonies of ants, with each colony trying 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. Both ACO methods significantly outperform the reinforcement learning technique proposed by Santana *et al.*, for three graph topologies (out of six) and irrespective of the number of the involved patrolling agents.

Several research direction can be explore to find better solutions to this complex multi-agent problem by using ACO techniques. First, detailed investigation is currently in progress to determine the appropriate parameter settings for our algorithms. Second, as has been emphasized in section 5, patrolling strategies found by the current ACO methods constraint each agent to start from a node and to come back to that node at the end of the tour. This strategy gives good results only for some graph topologies. Hence when agents start from the same node, two stages should be required. The first stage would consist in determining the individual strategies of all agents when starting from the same node. The second stage would consist in finding the individual agents' strategies when starting from the node they reached before coming back to the starting node at the preceding stage. Finally, it is anticipated that integrating problem-specific heuristic elements (like 2-opt) into our ACO approach, as has been done by Dorigo and Gambardella [6] for the TSP, would improve both algorithm efficiency and solution quality.

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. http://www.lamsade.dauphine.fr/čhevaley/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] H. Kitano. RoboCup Rescue : A Grand Challenge for Multi-Agent Systems. In *Proceedings of the 4th International Conference on Multi Agent Systems*, pages 5–12, 2000.

[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.

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

[14] RoboCup Rescue, Last visit in November 2005.

[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.

[17] R. Schoonderwoerd, O.E. Holland, J.L. Brutten, and L.J.M. Rothkrantz. Ant-based load balancing in telecommunication networks. In *Adaptive Behaviour*, volume 5, pages 169–207, 1997.

[18] R. Sutton and A. Barto. *Reinforcement Learning : An Introduction*. Cambridge, MA, 1998.

[19] I.A. Wagner, M. Lindenbaum, and A.M. Bruckstein. Efficiently Searching a Graph by a Smell-Oriented Vertex Process. In *Annals of Mathematics and Artificial Intelligence*, volume 24, pages 211–223, 1998.

[20] I.A. Wagner, M. Lindenbaum, and A.M. Bruckstein. Distributed Covering by Ant-Robots Using Evaporating Traces. In *IEEE Transactions on Robotics and Automation*, volume 15, pages 918–933, 1999.

[21] V. Yanowski, I.A. Wagner, and A.M. Bruckstein. A Distributed Ant Algorithm for Efficiently Patrolling a Network. In *Algorithmica*, volume 37, pages 165–186, 2003.