# Hybrid ACO/EA Algorithms applied to the Multi-Agent Patrolling Problem

Fabrice Lauri
IRTES-SeT
Rue Thiery-Mieg, 90010 Belfort, FRANCE
Email: fabrice.lauri@utbm.fr

Abderrafiaa Koukam
IRTES-SeT
Rue Thiery-Mieg, 90010 Belfort, FRANCE
Email: abder.koukam@utbm.fr

***Abstract*—Patrolling an environment consists in visiting as frequently as possible its most relevant areas in order to supervise, control or protect it. This task is commonly performed by a team of agents that need to coordinate their actions for achieving optimal performance. We address here the problem of multi-agent patrolling in known environments where agents may move at different speeds and visit priorities on some areas may be specified. Two classes of patrolling strategies are studied: the single-cycle strategies and the partition-based strategies. Several single-core and multi-core variants of a template state-of-the-art hybrid algorithm are proposed for generating partition-based strategies. These are experimentally compared with a state-of-the-art heuristic-based algorithm generating single-cycle strategies. Experimental results show that: the heuristic-based algorithm only generates efficient strategies when agents move at the same speeds and no visit priorities have been defined; all single-core variants are equivalent; multi-core hybrid algorithms may improve overall quality or reduce variance of the solutions obtained by single-core algorithms.**

## I. Introduction

A patrol is a mission involving a team of several individuals whose goal consists in *continuously* visiting the relevant areas of an environment, in order to efficiently supervise, control or protect it. A group of drones searching for wildfires in order to contribute in the forest conservation, a team of vaccum cleaning robots searching for dirt, postmen on their daily rounds, or a squad of marines securing an area are all examples of patrols. Performing such a task implies that all of the involved members coordinate their actions efficiently.

The techniques that solve the multi-agent patrolling problem (MAPP) are divided into two families. The techniques of the first family have a complete knowledge of the environment. In this case, the environment to patrol can be represented by a graph. Each node of this graph represents a convex area in the environment. Techniques of the second family have only a local knowledge of the environment. In this case, each patrolling member has to maintain a cognitive map from the perception of its environment, in order to properly act. In this paper, we focus on the problem of the multi-agent patrolling of a known environment.

Techniques of the first family can be used in numerous applications, ranging from network management [15], the rescue by robots of people in danger after a disaster [14], [6] to the protection of a territory to face enemy threats [5], [14], [2], [11].

The multi-agent patrolling problem in known environments has been formulated recently [12]. This problem consists in determining a patrolling strategy that minimizes a given performance criterion. A patrolling strategy is made up of several individual patrolling strategies, one for each involved agent. An individual strategy represents the graph nodes an agent has to visit. It can be defined prior to the patrol or while the agents are patrolling.

In this paper we address the problem of multi-agent patrolling where agents may move at different speeds and visit priorities on nodes may be specified. We propose studying especially two classes of patrolling strategies, namely the single-cycle strategies and the partition-based strategies [3]. The strategies belonging to the first class may be computed by an algorithm based on the Lin-Kernighan heuristic [10], as pointed out by Chevaleyre [3]. The strategies of the second class may be computed by an hybrid algorithm based on a combination of an Evolutionary Algorithm (EA) and an Ant-Colony Optimization (ACO) algorithm [9]. We propose single-core and multi-core variants for solving this problem. All these algorithms are experimentally compared on different graph topologies, agent populations, with or without the same agent speeds and visit priorities on nodes.

The remainder of this paper is organized as follows. Section II formalizes the patrolling problem and Section III gives an overview of the related works. Section IV presents the hybrid algorithms for solving this problem, with an emphasize on a biggest class whose single-cycle strategies and partition-based strategies belong to: the cyclic patrolling strategies. Section V presents and discusses the experimental results. Finally, concluding remarks and future research issues are given in section VI.

## II. Problem Formulation

The environment that has to be patrolled consists of a directed connected graph $G = (V, E)$. $V$ represents the strategically relevant areas and $E \subset V^2$ the means of transport between them, such that for any $x, y \in V$:

- $(x, x) \in E$.
- $(x, y) \in E \implies (y, x) \in E$.
- there exists a path $(x_1, x_2, \ldots, x_n)$ linking node $x = x_1$ to node $y = x_n$, where $n > 1$ and for any $i = 1, \ldots, n-1$, $(x_i, x_{i+1}) \in E$.

A cost $c(x, y) \in \mathbb{R}$ is associated with any edge $(x, y) \in E$. It may measure the distance (in meters for example) required to reach node $y$ from node $x$. The cost function $c : E \to \mathbb{R}$ satisfies the following properties:

- $c(x, y) \geq 0$ for any $(x, y) \in E$
- $c(x, y) = 0$ iff $x = y$,

Let $r < |\mathcal{V}|$ denote the number of agents patrolling graph $G$. Each agent $i$ is assumed to be located at node $sn_i \in V$ prior to the patrolling and to possess a movement speed $s_i > 0$ (in m/s for instance). Node $sn_i$ represents the deployment site of agent $i$. Agent $i$ reaches node $y$ from node $x$ after $\frac{c(x,y)}{s_i}$ units of time (seconds for instance).

With any node $x$ is associated an *instantaneous node idleness*, which represents the time period this node remains unvisited, and a *discount factor* $\gamma_x \in \mathbb{R}_{+*}$ [1], which influences the increase in the node idleness. When any node receives the visit of an agent, its idleness drops to zero. If node $x$ has been left unvisited for a period $\Delta t$, its idleness equals $\gamma_x \Delta t$.

Let $\mathcal{I} = (G, r, \overrightarrow{sn}, \overrightarrow{s}, \overrightarrow{\gamma})$ be an instance of the multi-agent patrolling problem, where $G$ is the patrolling graph, $r$ the number of patrolling agents, $\overrightarrow{sn} \in V^r$ the agent deployment sites, $\overrightarrow{s} \in \mathbb{R}^r_{+*}$ the agent speeds and $\overrightarrow{\gamma} \in \mathbb{R}^r_{+*}$ the discount factors of the nodes. Solving the multi-agent patrolling problem on $\mathcal{I}$ consists in elaborating a coverage strategy $\pi^{\mathcal{I}}$ of graph $G$ by $r$ agents such that any node of $G$ is visited infinitely often. Such a patrolling strategy must optimize a given quality criterion. For the sake of clarity, a multi-agent patrolling strategy will be from now on noted $\pi$ whenever there is no ambiguity on the instance $\mathcal{I}$.

Let $\Pi$ be the set of all the multi-agent patrolling strategies $\pi$ defined as follows:

- $\pi = (\pi_1, \pi_2, \cdots, \pi_r)$ is made up of $r$ individual strategies.
- Any individual strategy $\pi_i : \mathbb{N}^* \to V$ maps a discrete time space into the node set, with $\pi_i(1) = sn_i$. $\pi_i(j)$ denotes the $j$-th node that agent $i$ has to visit, with $\pi_i(j + 1) = x$ only if $(\pi_i(j), x) \in E$.

We are concerned to determining patrolling strategies that minimize the idleness of any node $x \in V$. Several criteria have been devised in [12] in order to evaluate the quality of a multi-agent patrolling strategy on a graph. For the sake of theoretical analysis, only the criterion based on the *worst idleness* will be used in this paper. The interested reader can consult Machado *et al.* [12] for other evaluation criteria. Knowing that the chosen criterion, that is the worst idleness of the graph, upper bounds the others ([3]), minimizing it implies minimizing the others.

All of the evaluation criteria can be formulated from the notion of instantaneous node idleness (INI). Assuming the agents follow strategy $\pi$ on graph $G$, the INI $I^{\pi}_t(x) \in \mathbb{R}_{+*}$ of node $x$ at time $t$ is the elapsed *discounted* duration since this node has received the visit of an agent. If node $x$ has been visited at time $t$ by an agent and if $\Delta t$ is the elapsed time since the last visit at node $x$, then the instantaneous idleness of node $x$ at time $t + \Delta t$ is given by:

$$I^{\pi}_{t+\Delta t}(x) = \gamma_x \Delta t \tag{1}$$

Discount factors can be used to set *visit priorities* on nodes. The higher the discount factor, the faster the idleness of the corresponding node grows. By convention, at initial time, $I^{\pi}_0(i) = 0$, for any strategy $\pi$ and for any node $i = 1, 2, \cdots, |\mathcal{V}|$.

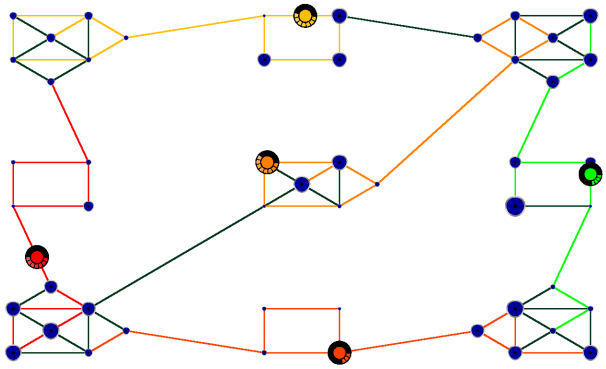


Fig. 1. Snapshot of a multi-agent patrolling strategy involving 5 agents on a 50-node graph. Blue circles represent nodes. Sliced circles are agents. The instantaneous idleness of a node is indicated by the size of its circle. Colored edges link the nodes visited infinitely often by the agents in their cycles. Black edges are not taken by any agent during cycles.

A snapshot of a multi-agent patrolling strategy is depicted in the Fig. 1.

Evaluating the multi-agent patrolling strategy $\pi$ using the worst idleness criterion consists in using the following equation:

$$WI^{\pi} = \limsup_{t \to +\infty} WI^{\pi}_t \tag{2}$$

where $WI^{\pi}_t$ denotes the *instantaneous worst graph idleness* which is the highest instantaneous node idleness over the set $V$ of nodes of $G$ at time $t$, that is:

$$WI^{\pi}_t = \max_{x \in V} I^{\pi}_t(x) \tag{3}$$

Solving the multi-agent patrolling problem thus consists in determining a strategy $\pi^*$ such that for any strategy $\pi$, $WI^{\pi^*} \leq WI^{\pi}$, that is:

$$\pi^* \in \operatorname*{argmin}_{\pi \in \Pi} WI^{\pi} \tag{4}$$

## III. Related Works

The papers by Machado [12], [11] are the seminal works about multi-agent patrolling on a graph. More recent works are those by Almeida [1], Chevaleyre [3], [4], Santana [16], Lauri [7], [9], Marier [13] and Poulet [14].

In [12], [11], several multi-agent architectures and multi-agent patrolling strategy evaluation criteria 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

[1] $\mathbb{R}_{+*} = \{x \in \mathbb{R} | x > 0\}$

of the next node (randomly, using the individual idleness or a shared idleness, the path length...), etc.

[1] improved the best architectures proposed by [11]. They have devised 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 the nodes that are reachable within a reasonable amount of time. Thus negotiations allow agents to attain a mutual agreement.

Chevaleyre [3] has formulated the patrolling problem in terms of a combinatorial optimization problem. He first proved that a patrolling strategy involving one agent could be obtained using an algorithm that solves the *Graphical Traveling Salesman Problem*. In this variant of the *Traveling Salesman Problem*, graphs are not necessarily complete. He then studied several possible classes of multi-agent patrolling strategies and showed that they all were able to represent close to optimal solutions.

In [16], the agents 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 each 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 (i.e. through the environment) communicate their intention for the next action. The latter agent architecture, named *Gray-Box Learner Agents (GBLA)*, was the most successful of both in this paper.

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 was shown that the single-cycle based strategy, one of the classes of multi-agent patrolling strategies presented in [3], gave the best results in all of the configurations except one, whereas the two most efficient architectures proposed in [12], [11] yielded the worst results for these experiments. All of the other schemes presented in [1], [16] gave equivalent performances.

Lauri *et al.* [7], [9] proposed several Ant Colony Optimization (ACO) techniques. In [7], the proposed ACO algorithms seek for patrolling strategies that constraint all the agents to start patrolling from and end patrolling at the same initial deployment node. Some of the strategies found by the ACO techniques in [7] are clearly suboptimal, due to the strong assumption imposed on them. In [9], an attempt toward extending the class of the multi-agent patrolling strategies is investigated. An hybrid algorithm combining an evolutionary algorithm (EA) with an ACO algorithm is proposed in order to allow agents to visit some nodes prior to patrolling. This stage prior patrolling consists in spreading the agents over the graph so that they are at the most distant distances from each other. Spreading them over the graph allow them to perform precyles, that is agents may visit some nodes once before entering their patrolling cycle. This technique has proven its efficiency over GBLA for the majority of the evaluated graphs and for populations composed of 2 to 20 agents. Most strategies computed by [9] are partition-based strategies, according to the terminology by [3].

Marier *et al.* [13] define the multi-agent patrolling problem as a Generalized Semi-Markov Decision Process (GSMDP). This mathematical model can handle continuous time and uncertainties in the execution of a patrol. The authors present an anytime algorithm based on a GSMDP-state search heuristic. Validation experiments address graphs patrolled by two agents and with at most 50 nodes.

Finally, Poulet *et al.* [14] formulate another version of the multi-agent patrolling problem, by introducing priorities on the nodes, metric performance criteria and an agent population whose size is dynamic. They lead several experiments for evaluating the performance of the algorithms proposed in [1] and [3] on instances of the new defined problem. Empirical results show that the single-cycle strategies by [3] and those generated from one of the heuristics proposed by [1] are clearly better than others.

## IV. Hybrid algorithms for the MAPP

Due to the relative success of the single-core hybrid algorithm based on ACO and EA for solving the multi-agent patrolling problem [9], we investigate hereafter some new single-core and multi-core variants of this hybrid algorithm. Before reviewing these algorithms, let us define first the search space of the multi-agent patrolling strategies commonly used by all these algorithms.

### A. Search space of the multi-agent patrolling strategies

All the subsequent algorithms seek for multi-agent patrolling strategies that belong to the class of *consistent cyclic multi-agent patrolling strategies*. They are generalizations of single-cycle strategies, partition-based strategies and mixed strategies as defined by Chevaleyre [3]. In single-cycle strategies (Fig. 2a), the same cycle involving all the graph nodes is performed by every agent in the same order and in the same direction but with time lags between agents. In partition-based strategies (Fig. 1 and Fig. 2b), the graph nodes are partitioned among agents, so that two given agents visit different nodes during their cycles. Mixed strategies (Fig. 2c) are partition-based strategies among groups of agents, so that agents of the same group visit the same set of nodes in the same order and direction but with time lags.

A multi-agent patrolling strategy $\pi$ is cyclic iff each of its individual strategy $\pi_i$ is parameterized by a tuple $(\mu_i, l_i)$ where:

- $\mu_i = (\mu_i(1), \ldots, \mu_i(l_i), \ldots, \mu_i(N_i))$ is a finite sequence of $N_i$ nodes,
- $\mu_i(1) = sn_i$,
- $l_i$ represents the index of the node that begins and ends the cycle of agent $i$, that is: $\mu_i(l_i) = \mu_i(N_i)$.

and such that:

$$\pi_i(j) = \begin{cases} \mu_i(j) & \text{for} \quad j < N_i \\ \mu_i(l_i + (j - l_i) \mod (N_i - l_i)) & \text{for} \quad j \geq N_i \end{cases} \tag{5}$$

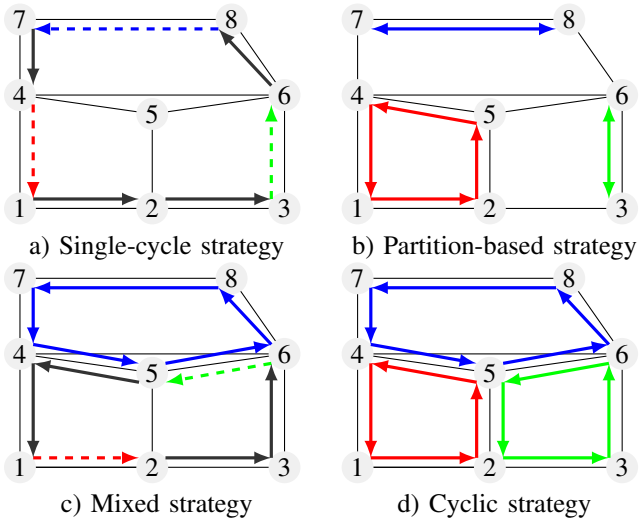



Fig. 2. Examples of patrolling strategies involving 3 agents. Dashed arrows emphasize the time lags encountered by agents patrolling the same set of nodes.

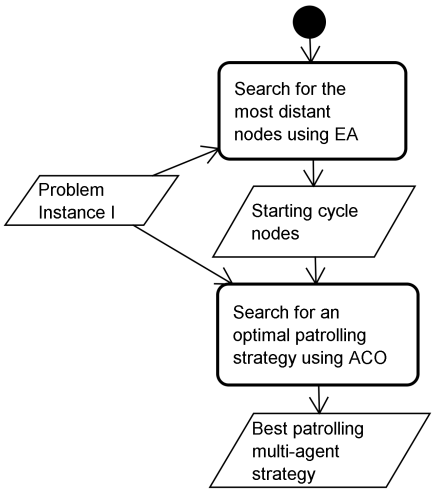



Fig. 3. Flowchart of a Single-Core ACO/EA Algorithm

The individual patrolling strategies in a cyclic multi-agent patrolling strategy are characterized by the existence of a cycle and possibly of a precycle. Both definitions follow. The patrolling cycle $\text{cyc}(\pi, i)$ of agent $i$ in a cyclic multi-agent patrolling strategy $\pi$ is the finite sequence of nodes of $\pi_i$ visited infinitely often by agent $i$, that is $\text{cyc}(\pi, i) = (\pi_i(l_i), \pi_i(l_i + 1), \cdots, \pi_i(N_i))$. The precycle of agent $i$ in a cyclic multi-agent patrolling strategy $\pi$ is the sequence of nodes of $\pi_i$ visited only once by agent $i$ from its deployment site $sn_i$ to the node $\pi_i(l_i)$ beginning its patrolling cycle. Whenever $l_i = 1$, there is no precycle in $\pi_i$. A cyclic multi-agent patrolling strategy is consistent if any node of $G$ is visited infinitely often by at least one agent in its patrolling cycle. This hypothesis of consistency is a necessary and sufficient condition for the worst idleness of a multi-agent patrolling strategy to be bounded. Fig. 2d is an example of a consistent cyclic multi-agent patrolling strategy that is neither single-cycle, nor partition-based, nor mixed. In this strategy example, nodes 4, 5 and 6 are visited infinitely often by all agents and nodes 2 and 5 are visited infinitely often by two agents.

In the sequel, $\Pi^{cyclic}$ denotes the set of all the consistent cyclic multi-agent patrolling strategies for a given instance of the multi-agent patrolling problem. The search spaces of all the following algorithms are included in $\Pi^{cyclic}$.

### B. Single-core algorithms for the MAPP

The single-core hybrid ACO/EA algorithm named *EA-AD+GG-AA* whose general architecture is depicted in Fig. 3 has been defined and experimentally validated by Lauri [9].

This hybrid algorithm seeks for an optimal partition-based strategy, by applying an evolutionary algorithm followed by an ACO algorithm. The EA consists in determining the set of the most distant nodes to which agents will head for from their deployment sites $\overrightarrow{sn}$. They also represent the first and last nodes visited by agents during their cycles. These starting cycle nodes are determined once by the EA and remain fixed afterward during the execution of ACO. The ACO algorithm consists in choosing the nodes that will be visited infinitely often by agents during their cycles. The ACO algorithm uses several competitive ant colonies. Each colony represents a complete patrolling strategy $\pi$. Any ant $k$ of a colony progressively builds an individual strategy $\pi_k$. So there are as many ants in a colony as patrolling agents, that is $r$. Each ant move probabilistically toward an *available* node $y$ by considering the distance between its current node $x$ and $y$ and the amount of pheromone deposited on the edge $(x, y)$ of a complete graph $G'$. The available nodes of an ant are those that have not been visited yet by ants of its colony. They are stored in a *tabu list*. A tabu list is associated with each colony. The probability that ant $k$ of colony $l$ moves to node $y$ from node $x$ is given by:

$$p_{xy}^{k,l} = \begin{cases} \frac{[\tau_{xy}(T)]^{\alpha}\ [\eta_{xy}]^{\beta}}{\sum_{u \in allowed_l} [\tau_{xu}(T)]^{\alpha}\ [\eta_{xu}]^{\beta}} & \text{if } y \in allowed_l \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

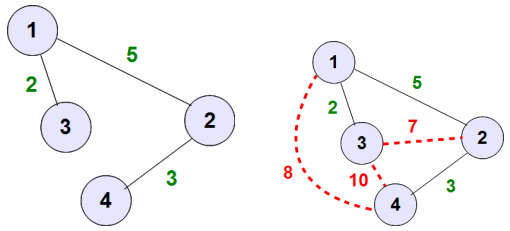
where $allowed_l = V - tabu_l$ is the set of the unvisited nodes of colony $l$, $\tau_{xy}(T)$ is the pheromone intensity on edge $(x, y)$ at iteration $T$ of the algorithm, $\eta_{xy} = \frac{1}{c(x,y)}$ is the visibility of node $y$ from node $x$, and $\alpha$ and $\beta$ are parameters that control the relative importance of pheromone intensity and visibility.

$G'$ is obtained from the initial graph $G$ by adding all the missing edges and setting their cost to the shortest-path distance between pairs of nodes (see Fig. 4).

Once all the nodes have been visited by the ants of a colony, they deposit on the edges appearing in the tour built from the complete graph an amount of pheromone that is inversely proportional to the quality (worst idleness) of the associated strategy, as follows:

$$\tau_{xy}(T + 1) = (1 - \rho)\ \tau_{xy}(T) + \Delta\tau_{xy} \quad (7)$$

where $\rho$ is the evaporation coefficient, $\tau_{xy}(T+1)$ and $\tau_{xy}(T)$ are the pheromone intensities on edge $(x, y)$ at iterations $T + 1$ and $T$, respectively. $\Delta\tau_{xy}$ is the pheromone quantity deposited on edge $(x, y)$.




a) Initial Graph $G$ b) Complete Graph $G'$

Fig. 4. A complete graph is used by ACO from the initial patrolling graph. Costs of the missing edges are the cumulated distances of the shortest paths.

Several variants of ACO may be considered for improving the average performance or the variance of the found solutions by exploring in a more efficient way within the search space of the partition-based strategies. All these variants follow the template algorithm presented below.

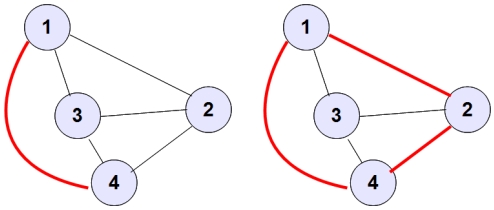
**ACO template algorithm for generating multi-agent patrolling cycles**

**Require:** $T_{Max} \in \mathbb{N}^*$: number of iterations, $N_c \in \mathbb{N}^*$: number of colonies in competition, $\overrightarrow{s} \in V^r$ : starting cycle nodes, parameters $c, \alpha, \beta, \rho \in \mathbb{R}$
**Ensure:** Multi-agent patrolling strategy.

For each edge of each pheromone graph, set the initial pheromone quantities to $c$.
$n \leftarrow 1$
**for** Each iteration $T = 1, 2, \ldots, T_{Max}$ **do**
  **for** Each colony $l = 1, 2, \ldots, N_c$ **do**
    Empty the tabu list $tabu_l$ associated with colony $l$ (all the nodes are available).
    Place each ant $k$ of colony $l$ on the starting cycle node $\overrightarrow{s}_k$.
    Add the starting cycle node of each ant of colony $l$ into $tabu_l$.
    Generate solution $s_n$ by moving all the ants of colony $l$ and keeping their node sequence until $tabu_l = V$. Each time an ant is moved to node $x$, $x$ is added into the tabu list.
    Evaluate the performance (worst idleness) $f(s_n)$ of solution $s_n$.
    $n \leftarrow n + 1$
  **end for**
  Update the pheromone quantities using the performances of the solutions found in the last iteration.
**end for**
**return** The best patrolling strategy: $s_{best} \in \text{argmin}_{s=s_1,s_2,\ldots,s_{n-1}} f(s)$.

Better solutions may be found by proposing new ways of storing and depositing pheromones. For example:

- Instead of using only one graph of pheromones, one pheromone graph per agent could be used. Pheromones deposited on the edges of the pheromone graph of agent $i$ would be those from the ants of index $i$ in every colony.
- Instead of being deposited on the edges directly linking the pairs of the nodes appearing in the cycle tour of an ant, pheromones could be also deposited on the intermediate edges of a valid path in $G$, as illustrated by Fig. 5.




a) Without using intermediate edges b) Using intermediate edges

Fig. 5. Pheromone updates assuming the node sequence $(1, 4)$ appears in the cycle of an agent that has to patrol on the graph of Fig. 4a.

- Instead of allowing the ants of all the colonies to deposit pheromones, only the ants of the best colonies would be allowed to do it.

These design choices lead to building several algorithm variants, whose name will explicitly give information on which previous features are taken into account, as follows:

| Acronym | Feature |
|---|---|
| **GG** | Global Pheromone Graph |
| **AG** | Agent Pheromone Graph |
| **AA** | All Ants deposit pheromones |
| **BA** | Only the Best Ants deposit pheromones |
| **IE** | Deposit pheromone on Intermediate Edges |

For example, *EA-AD+AG-BA-IE* refers to the hybrid algorithm that:

- spreads out the agents over the graph using the Evolutionary Algorithm proposed by [9],
- uses a pheromone graph for each patrolling agent,
- considers that only the ants of the best colonies deposit pheromones,
- allows ants to deposit pheromones on intermediate edges.

A subset of algorithms among all these possible variants will be experimentally studied in section V. All the subsequent algorithms use the technique proposed in [8] to evaluate the worst idleness of any consistent cyclic multi-agent patrolling strategy.

Another approach to enhance the search of a solution within the partition-based strategy space consists in using parallel algorithms. This is the subject of the next subsection.

### C. Multi-core algorithms

There are several ways to parallelize a population-based algorithm. One approach is to execute the common operations made successively on different individual solutions in the population within several threads that manipulates the same shared memory representing the population of solutions. Another approach is to execute several threads that

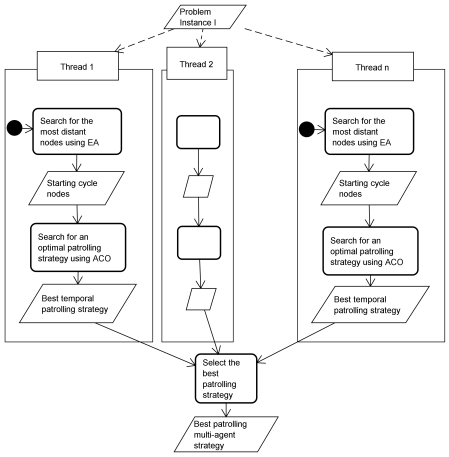



Fig. 6. Flowchart of a Multi-Core ACO/EA Algorithm

use their own local population of solutions and synchronizing them to obtain the best solution, as illustrated in Fig. 6.

We adopted this approach to build the multi-core variants from all the preceding single-core algorithms, both for its simplicity and for the limited number of messages exchanged between threads.

In next sections, the acronym $n$-*core* within the name of an algorithm indicates that $n$ cores were used. $n = 1$ refers to a single-core algorithm, $n > 1$ refers to a multi-core algorithm.

## V. Experimental Results

Multi-agent patrolling strategies were computed on 4 different graph topologies of various complexity (Fig. 7) with populations involving from 5 to 30 agents.

### A. Experimental Protocol

A subset of the possible hybrid algorithms described previously were compared experimentally. These algorithms are: *EA-AD+1-core GG-AA* [9], *EA-AD+1-core GG-AA-IE*, *EA-AD+1-core GG-BA-IE*, *EA-AD+1-core AG-AA-IE*, and *EA-AD+1-core AG-BA-IE* and *EA-AD+8-core GG-BA-IE*.

We conducted some preliminary experiments in order to determine the set of parameters for these hybrid EA/ACO algorithms that leads to a tradeoff between performance and computing time. The design parameters chosen for all the ACO algorithms are shown in table I.

TABLE I. Set of design parameters

| | |
|---|---|
| # Total Generated Solutions ($T_{Max}$) | 300 |
| # Colonies ($N_c$) | 128 |
| Pheromone Initialization Constant ($c$) | 0.01 |
| Pheromone Relative Importance ($\alpha$) | 1 |
| Visibility Relative Importance ($\beta$) | 1 |
| Evaporation Rate ($\rho$) | 0.2 |

For the multi-core algorithm *EA-AD+8-core GG-BA-IE*, each of the 8 threads uses 16 colonies, for a total of $8 \times 16 = 128$ colonies.

### B. Comparative results

Fig. 8 presents the performance results of single-cycle patrolling strategies and partition-based patrolling strategies. The single-cycle strategies were computed using an algorithm based on the heuristic Lin-Kernighan [10]. This algorithm is called *LKB* thereafter. The partition-based strategies were computed using the hybrid algorithms presented previously.

For a given instance of the problem (graph, number of agents, agent speeds and visit priorities), 10 patrolling strategies were computed. Thus, subsequent results represent the average worst idlenesses over the 10 runs. Also shown on these figures are the minimum worst idlenesses and the maximum worst idlenesses over these 10 runs.

*1) Comparative results between single-core algorithms:* One may first notice that all these algorithms manage to generate strategies whose efficiency depends on the number of involved agents. Generally, the more agents, the smaller worst idleness. One may also quickly observe that all the hybrid variants are equivalent. This highlights how difficult it may be for a given problem to devise ways based on the use of pheromones that allow cooperative ants to communicate relevant information between them so that good solutions rapidly emerge.

Let us now compare hybrid algorithms with *LKB*. The hybrid algorithms perform at best equivalently to *LKB* when all the agents have the same speeds and no visit priorities on nodes have been specified. They outperform *LKB* when some different agent speeds and visit priorities have been defined. *LKB* is based on a heuristic that is commonly used to search for optimal tours in TSP instances. These results clearly emphasize the limits of this heuristic approach that does not take into account the hypothesis on agent speeds and visit priorities.

*2) Comparative results with the multi-core algorithm:* As shown in all the figures, the overall quality or the variance of the solutions obtained by single-core hybrid algorithms can be enhanced by multi-core algorithms. This can be explained by the fact that in the single-core variants, all the colonies of the ACO algorithm exploits the same set of starting nodes computed by the EA. In the multi-core variants, two ACO algorithms executed on different threads may exploit possibly different starting nodes. Thus more promising regions of the search space can be exploited in these multi-core algorithms.

## VI. Concluding remarks and future works

The problem of multi-agent patrolling where agents may move at different speeds and visit priorities on nodes may be specified has been addressed in this paper. Two classes of patrolling strategies were especially studied: the single-cycle strategies and the partition-based strategies. Heuristic-based and hybrid algorithms generating such strategies were experimentally compared on different graph topologies, agent populations, with or without the same agent speeds and visit priorities on nodes. Experimental results show that: the heuristic-based algorithm only generates efficient strategies

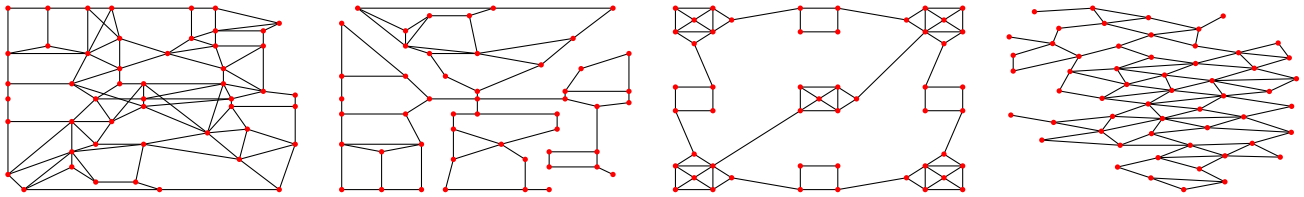



Fig. 7. Graph Topologies. Numbers within parentheses represent respectively the number of nodes, the number of edges and the degree of the graph. Graph degree denotes the maximum degree of the graph nodes. The degree of a node is the number of its incident edges.

when agents move at the same speeds and no visit priorities have been defined; all single-core variants are equivalent; multi-core hybrid algorithms may improve overall quality or reduce variance of the solutions obtained by single-core algorithms.

Several research direction can be explored to find better solutions to this complex multi-agent problem by using meta-heuristics. Future works include: enhancing the search of the single-core hybrid algorithms by devising new ways of communicating reliable information by the use of pheromones or other bio-inspired mechanisms, designing more sophisticated parallel hybrid algorithms, proposing algorithms for generating cyclic strategies, and generating problem instances whose optimal patrolling strategies are known.

## REFERENCES

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

[2] B. Bošanský, V. Lisý, M. Jakob, and M. Pechoucek. Computing Time-Dependent Policies for Patrolling Games with Mobile Targets. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2011.

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

[4] Y. Chevaleyre. *Combinatorial Optimization and Theoretical Computer Science*, chapter The Patrolling Problem: theoretical and experimental results. Wiley, 2007.

[5] A.X. Jiang, Z. Yin, C. Zhang, M. Tambe, and S. Kraus. Game-theoretic Randomization for Security Patrolling with Dynamic Execution Uncertainty. In *International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2013.

[6] H. Kitano. RoboCup Rescue : A Grand Challenge for Multi-Agent Systems. In *4th International Conference on Multi Agent Systems*, pages 5–12, 2000.

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

[8] F. Lauri, J.C. Créput, and A. Koukam. The Multi-Agent Patrolling Problem – Theoretical Results about Cyclic Strategies. In *12th International Conference on Practical Applications of Agents and Multi-Agent Systems*, 2014.

[9] F. Lauri and A. Koukam. A Two-Step Evolutionary and ACO Approach for Solving the Multi-Agent Patrolling Problem. In *IEEE World Congress on Computational Intelligence*, 2008.

[10] S. Lin and B.W. Kernighan. An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, 21:498–516, 1973.

[11] A. Machado, A. Almeida, and *al*. Multi-Agent Movement Coordination in Patrolling. In *3rd International Conference on Computer and Game*, 2002.

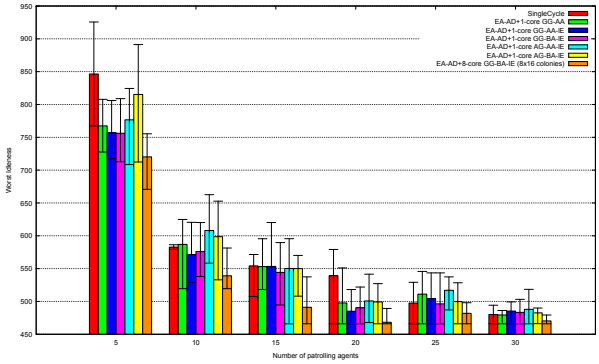
[12] A. Machado, G. Ramalho, and *al*. Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures. In *3rd International Workshop on Multi-Agent Based Simulation*, pages 155–170, 2002.

[13] J.-S. Marier, C. Besse, , and B. Chaib-draa. A Markov Model for Multiagent Patrolling in Continuous Time. In *International Conference on Neural Information Processing: Part II*, pages 648–656, 2009.

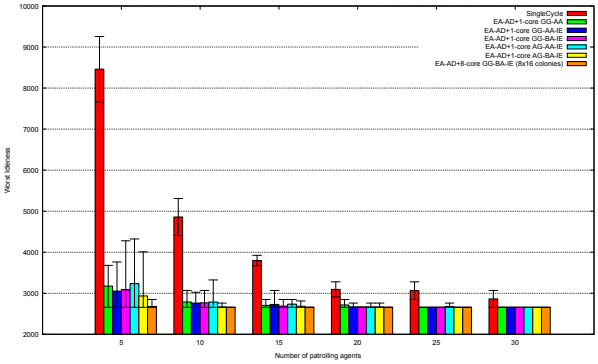
[14] C. Poulet, V. Corruble, A.E.F. Seghrouchni, and G. Ramalho. The Open System Setting in Timed MultiAgent Patrolling. In *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, 2011.

[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.

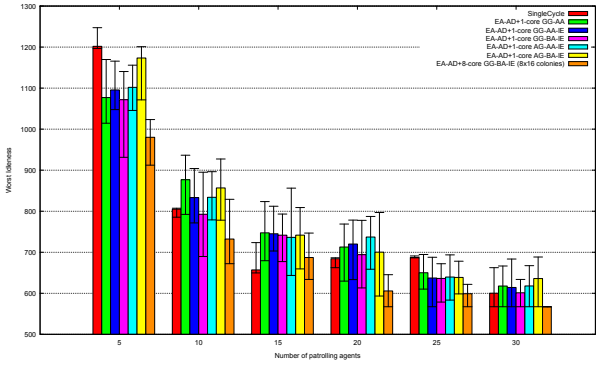
[16] H. Santana, G. Ramalho, and *al*. Multi-Agent Patrolling with Reinforcement Learning. In *3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1122–1129, 2004.



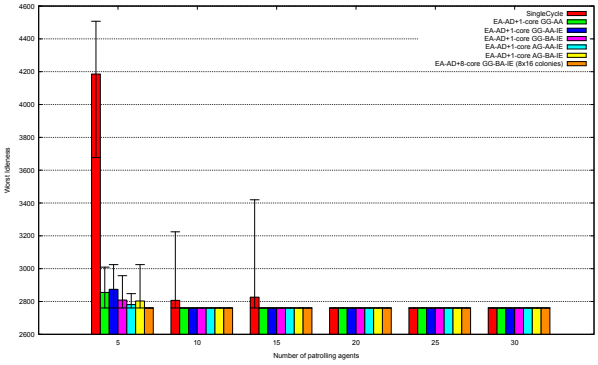

**a)** Results on "Map A" with same visit priorities and same agent speeds



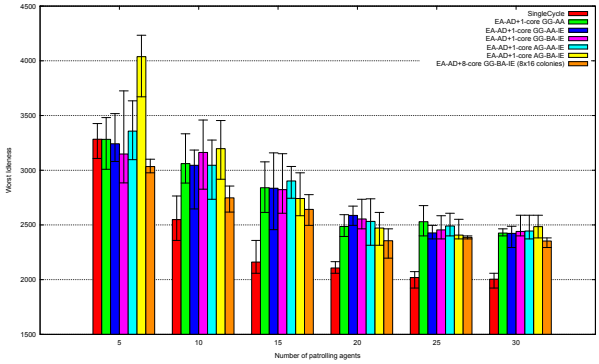

**b)** Results on "Map A" with some different visit priorities and agent speeds



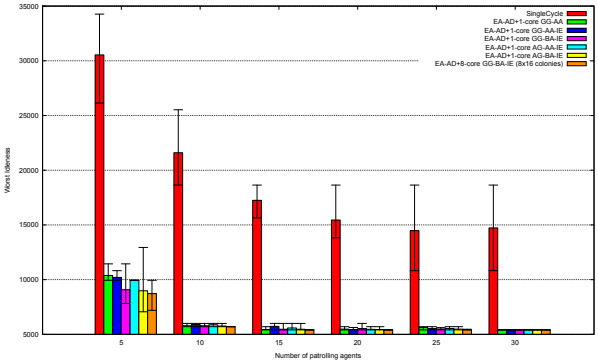

**c)** Results on "Map B" with same visit priorities and same agent speeds



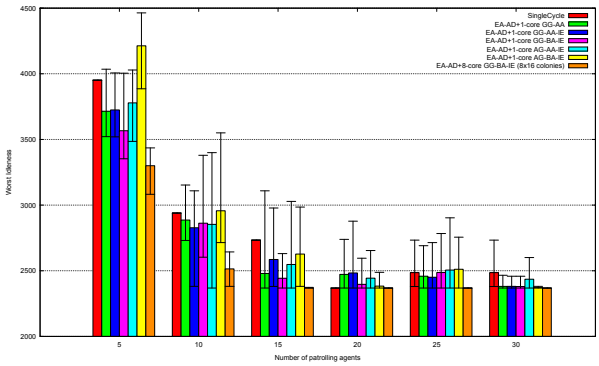

**d)** Results on "Map B" with some different visit priorities and agent speeds



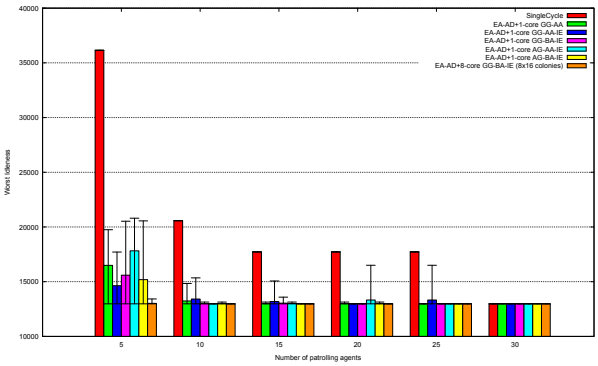

**e)** Results on "Island" with same visit priorities and same agent speeds




**f)** Results on "Island" with some different visit priorities and agent speeds




**g)** Results on "TownN56" with same visit priorities and same agent speeds




**h)** Results on "TownN56" with some different visit priorities and agent speeds

Fig. 8. Comparative Results