Multi-agent Patrolling in Dynamic Environments

Mehdi Othmani-Guibourg[1][2], Amal El Fallah-Seghrouchni[2], Jean-Loup Farges[1] and Maria Potop-Butucaru[2]
[1]ONERA, Toulouse, France
Email: {Mehdi.Othmani-Guibourg, Jean-Loup.Farges}@onera.fr
[2]Sorbonne Universités, UPMC Univ Paris 06
CNRS, UMR 7606, LIP6
F-75005, Paris, France
Email: {amal.elfallah, maria.potop-butucaru, mehdi.othmani}@lip6.fr

***Abstract*—For over a decade, the multi-agent patrolling task has received attention from the multi-agent community. A range of algorithms based on reactive and cognitive architectures has been developed. However, the existing patrolling-specific approaches regarding dynamic environment are still in preliminary stages. In this paper, we present a first study opening the multi-agent patrolling task to the assumption of varying environment. In order to accomplish this study we propose a formal model for dynamic environment grounded on the one hand on classical patrolling model and on the other hand on edge-markovian evolving graphs. An adaptation of two very different strategies of agent, Conscientious Reactive and Heuristic Pathfinder Cognitive Coordinated, to that environment is designed, implemented in a simulator and assessed. The results show the architecture implementing Heuristic Pathfinder Cognitive Coordinated strategy can patrol an area into dynamic environment more adequately than the one implementing the Conscientious Reactive strategy. Moreover the difference between the two strategies is larger in dynamic environment than in static environment.**

I. Introduction

To patrol is literally "the act of walking or travelling around an area, at regular intervals, in order to protect or supervise it" [1]. That area can be physical or abstract. Also, that task is by nature conveniently well-suited for being shared in space and time by several agents. There are a wide variety of problems that may be reformulated as particular multi-agent patrol task. As a concrete example, the task consisting in monitoring an area by a swarm of drones does face the problem of coordinating them all to patrol that area in order to detect the presence of intruders or any other event. The quality of strategy is evaluated by using different measures, each one measuring a specific property of distributions of visits generated by strategies. Informally, it is consensual that a good strategy is one that minimises the time lag between two passages on the same place and for all places.

Beyond drones patrolling an area, performing task efficiently can be useful for various application domains where distributed surveillance, inspection or control are required. For instance, agent patrolling could be useful for area surveillance of a forest by a swarm of drones in search of a start of fire or for detecting recently modified or new web pages to be indexed by search engines [6].

Just a few years ago, the new hypothesis of an open society of agent was proposed by Poulet [11] and evaluated to test if the multi-agent patrolling can be a robust modelling for the patrol task in attendance of some dynamic characteristics. In particular, it was shown in simulation that the leaving of one agent during the mission does not show loss of performance, the best scores being for the coordinated strategies.

Many tasks can be modelled as a problem of multi-agent patrolling in dynamic environment, namely a task where a pathway may be blocked as time goes by. Such a task could be the one where patrolling drones are blocked due to the presence of an enemy for example. As well, it could be the task where inspection vehicles in a town or in a factory has to share the road network with a large number of others vehicles. Depending on the activity of the other vehicle, the traffic of roads is either fluid or blocked. *The paper thereupon proposes an analysis of multi-agent patrolling in dynamic environments by introducing a model for such environments, adapting two ideal-type strategies and evaluating their performances according to that new assumption.*

Section II presents the state of the art on multi-agent patrolling and edge-markovian graph that is useful in order to understand proposed developments. Then in Section III, we show how modelling dynamic environments in a multi-agent patrolling problem, and we propose an extension of the selected multi-agent strategies enabling their use in the case of dynamic environment. In Section IV, the strategies adapted to dynamic environment will be analysed. Finally, Section V draws some conclusions and indicates directions for future work.

II. State of the Art

This section presents the background on multi-agent patrolling in static graph and on the dynamic model of environment we have used, which is the edges-markovian evolving graphs.

A. Multi-agent patrolling in static graph

Multi-agent patrolling also called *multi-agent timed patrolling* by Sampaio [12] is a convenient model allowing to represent the patrolling task. Thereupon, it is necessary to have a rigorous definition of the multi-agent patrolling to characterise the strategies implemented by the agents, the evaluation criteria and the tools.

978-1-5386-0768-8/17/$31.00 ©2017 IEEE

1) Formal definition: Let a society of agents, A, in which all agents are able to move in the environment with the same mobility parameters. As stated by Machado [7], the representation of the terrain can be replaced by an undirected graph representing the possible paths.

Given such a graph, the patrolling task refers to continuously visiting all the graph nodes so as to minimise the time lag between two visits. The edges may have different associated travel or transit times corresponding to the time taken by an agent to travel at its nominal speed the real distance between nodes.

As shown by Chevaleyre [5], the graph representing the territory is referred to as $G = (V, E)$, where $V = \{1, .., n\}$ is the set of nodes and E which is included in the set of 2-element subsets of V is the set of edges of G. To each edge $\{i, j\}$ corresponds a parameter $c_{i,j}$ representing the time taken by an agent to move across the edge $\{i, j\}$. At the beginning of an instance of a patrolling task, agents are positioned on nodes of G. When the patrolling task starts, agents move simultaneously around the nodes and edges of the graph according to their predetermined strategy. At a given time an agent is either at a node i or travelling on edge $\{i, j\}$ in motion from i to j or from j to i for a duration lower than $c_{i,j}$ according to its position and heading on the edge. Each node has a dynamic variable named *idleness*, indicating the time elapsed since it has not been visited by an agent. The idleness of a node i at time t, $I(i, t)$, is defined [5] as being the amount of time elapsed since that node has received the visit of an agent. The idleness of all nodes at the beginning of the patrolling task is set to 0.

Each time an agent arrives at a node i, it shall decide, among the edges including $\{i\}$ which is $e_{i,j}$ the next edge to travel.

A common specialisation of this general model, is the one where a sample time is defined and where all $c_{i,j}$ are expressed as integer numbers of sample time. In that case, the formal model can be integrated step by step: agent positions and node idlenesses at time t can be computed from agent positions, node idlenesses and agent decisions at time $t-1$ with relevant heading. Early studies [7] consider a simpler case where all $c_{i,j}$ are equal to one. In that specific case, at each step an agent is necessarily at a node.

2) Strategies: A strategy of agent is an information processing method allowing each agents to take a decision each time it arrives at a node.

In the multi-agent patrolling problem, whatever the strategy considered, each agent intends actions based on their appropriated perceptions on the environment and on their knowledge about the idlenesses of nodes.

Ideally, we can distinguish two kinds of strategies: the reactive ones and the cognitive ones. According to Almeida et al. [8] reactive agents simply act based on their current perception while cognitive ones may pursue a goal. Reactive agents cannot, by definition, plan a path to distant nodes whereas cognitive agents can perceive to a depth of d ($d > 1$) of graph [7].

Besides the actual idleness defined in the Section II, agents make idleness estimates. Those estimates can be produced assuming different hypothesis. Two extreme hypothesis are:

- *Individual idleness*: each agent considers only its own visits to reset its estimated node idleness
- *Shared idleness*: all agents considers visits of all agents to reset estimated node idleness. In case of perfect instantaneous communication shared idleness corresponds to actual idleness.

In this work we adapt and assess two strategies: *Conscientious Reactive* (CR) and *Heuristic Pathfinder Cognitive Coordinated* (HPCC) representing each one an ideal-type for the reactive strategies and the cognitive ones, respectively.

According to Almeida et al. [9] Conscientious Reactive (CR) selects for each agent its next node to visit as the one with the highest individual idleness from its neighbourhood.

Cognitive Coordinated (CC) can perceive the whole graph and selects its next node to visit as the one with the highest shared idleness from the whole graph, according to suggestion given by a centralising coordinator [9]. This coordinator is responsible for avoiding that more than one agent chooses the same next node and it answers instantaneously to provide a next node. The *Floyd-Warshall Algorithm* computes a path between the current node of the agent and the selected node.

An agent is described as an *Heuristic agent* (H) [8] whether, in the decision-making process, it takes into account not only the normalised idleness but also the normalised *time to go* of a candidate goal node from its current position. Idleness and time to go are normalised by scaling them between 0 and 1.

Maximum idleness is attributed a zero normalised value whereas to the minimum idleness is attributed a value equals to one. Intermediary values are calculated by means of proportions as shown in Equation 1:

$$\begin{aligned} &\text{If } min_{v \in V}\{I(v,t)\} \neq max_{v \in V}\{I(v,t) \in V\}, \forall v_0 \in V \\ &\bar{I}(v_0,t) = \\ &\qquad \frac{I(v_0,t) - max_{v \in V}\{I(v,t)\}}{min_{v \in V}\{I(v,t)\} - max_{v \in V}\{I(v,t)\}} \end{aligned} \tag{1}$$

where $\bar{I}(v, t)$ is the normalised idleness.

Normalised time to go is calculated similarly [8]. For that purpose, to the minimum time to go is attributed a zero normalised value whereas to the maximum time to go is attributed a value equals to one. Intermediary values are calculated by means of proportions as shown in Equation 2:

$$\begin{aligned} &\forall d_0 \text{ a time to go,} \\ &\qquad \bar{d_0} = \frac{d_0 - min\{d\}}{max\{d\} - min\{d\}} \end{aligned} \tag{2}$$

where $max\{d\}$ and $min\{d\}$ are the maximum and the minimum time to go respectively, over all the graph.

Then a Heuristic agent at the position v_0 evaluates the nodes as shown in Equation 3:

$$\begin{aligned} &\forall r \in [0,1], \forall v \in V, val_r(v,t) = \\ &\qquad r \times \bar{I}(v,t)) + (1-r) \times \bar{d}(v_0,v)) \end{aligned} \tag{3}$$

where the weighting factor r has to be chosen by the strategy designer.

Minimising the node values according to that expression allows agents to visit nearby nodes with higher idleness first and foremost.

A *Pathfinder agent* (P) [8] is the one which takes into account the idleness of the nodes in-between the current location and the goal to compute the best path leading there. For that, it weights the edges as shown in Equation 4:

$$\begin{aligned} &\forall r \in [0,1], \forall e \in E : e = \{i,j\}, \\ &\quad c_r(e) = r \times I(j,t) + (1-r) \times \bar{c}_{i,j} \end{aligned} \tag{4}$$

In that case, it is normalised transit time of edge and not normalised time to go of path that is used to value edges.

Minimising the node values according to that expression allows agents as well to visit nearby nodes with higher idleness first and foremost.

3) Evaluation criteria: Sampaio et al. [12] introduced evaluation criteria, relevant to establish aggregation measures that are not based on idleness but on the intuitive concept of *interval between visits* to the node. In this class of evaluation criteria, the size of intervals between visits at each node is calculated by registering the value of idleness just before each visit by an agent. All intervals for all nodes are used to make an aggregated calculation. The two interval-based evaluation criteria we selected are the *Mean Interval* (MI) and the *Quadratic Mean Interval* (QMI), the mean and the root mean square, respectively, on all intervals between visits of a mission execution.

In order to better evaluate the contribution of each agent when the population size varies, these evaluation criteria are normalized by multiplying values by the number of agents [11].

4) SimPatrol: SimPatrol is an open-source simulator of multi-agent systems constructed strictly for the multi-agent patrolling task introduced by Moreira et al. [10]. It was initially developed at the Center of Computing (Centro de Informática in Portuguese) of the Universidade Federal de Pernambuco (UFPE). Thereafter, SimPatrol was modified [11] for the open system multi-agent patrolling task.

B. Edge-markovian evolving graph

Casteigts [2] presented a unified framework for dynamic graphs called *TVG* for *time-varying graphs*. From that unified framework, three kinds of time-varying graph are relevant for modeling of multi-agent patrolling from the most abstract to the most concrete.

A *TVG* denoted $\mathcal{G}$ is a 5-uplet $(V, E, \mathcal{T}, \rho, \xi)$ where $V = 1, ..., n$ is the set of nodes, $E \subseteq V^2$ is the set of edges of G, $\mathcal{T} \subseteq \mathbb{T}$ is the *lifetime* of the system where the temporal domain $\mathbb{T}$ is generally assumed to be $\mathbb{N}$ for discrete-time systems or $\mathbb{R}^+$ for continuous-time systems, $\rho : E \times \mathcal{T} \to \{0,1\}$, called *presence* function, indicates whether a given edge is available at a given time and $\xi : E \times \mathcal{T} \to \mathbb{T}$, called *latency* function, indicates the time it takes to cross a given edge when starting at a given date. The latency of an edge could vary in time.

A *(discrete-time) random time-varying graph* [2] is a TGV whose lifetime is an interval of $\mathbb{N}$ and whose *sequence of characteristic graphs* $\mathcal{S}_\mathcal{G} = G_1, G_2, ...$ is such that every G_i is a Erdös and Rényi random graph [3]; that is, $\exists p : \forall e \in V^2, P[e \in E_{G_i}] = p$.

Clementi et al. [4] introduced *Edge-Markovian Evolving Graphs* which are discrete-time evolving graphs in which the presence of every edge follows an individual Markovian process. More precisely, the sequence of characteristic graph $S_\mathcal{G} = G_1, G_2, ...$ is such that

$$\forall e \in V^2, \exists\ p, q \in [0,1] : \begin{cases} P(e \in E_{G_{i+1}} | e \notin E_{G_i}) = p \\ P(e \notin E_{G_{i+1}} | e \in E_{G_i}) = q \end{cases} \tag{5}$$

where p and q are called *birth rate* and *death rate*, respectively. The probability that a given edge remains absent or present from G_i to G_{i+1} is obtained by complement of p and q.

III. Dynamic Environment for Patrolling

This section presents our contribution, thas is the modelling of dynamic environment and the subsequent adaptation of strategies.

A. Modelling of dynamic environment

In order to make the environment dynamic, we modelled it with an edge-markovian evolving graph as introduced in Section II, allowing the presence of edges at a time step t to be dependent on their presence at the previous time step $t-1$. This choice of modelling enables a generic model for dynamic environments and a relevant one for several kinds of scenarios. However, we made some changes to initial model.

First, unlike edge-markovian evolving graph presented in Section II, varying edges are not the set V^2 but the set E.

Then we simplify the function ξ into $\xi' : E \to \mathbb{T}$ representing now the time it takes to travel an edge. Indeed, in our model edges have lengths so that each unit of edge length is travelled in one cycle. ξ' refers thereupon to the number of cycles needed to cross an edge: $\xi(\{i,j\}) = c_{i,j}$.

Finally, we setted a vector (p_e, q_e) for every $e \in E$ instead of only one p and q as in the initial model of edge-markovian evolving graph; Equation 5 thereupon becomes:

$$\forall e \in V^2, \exists p_e, q_e \in [0,1], \begin{cases} P(e \in E_{G_{i+1}} | e \notin E_{G_i}) = p_e \\ P(e \notin E_{G_{i+1}} | e \in E_{G_i}) = q_e \end{cases} \tag{6}$$

According to properties of studied scenarios, $\forall e \in E, p_e$ and q_e the *birth rate* and *death rate* respectively, could be adjusted.

In our model, an edge-markovian evolving graph denoted $\mathcal{G}$ is then a 4-uplet $(G, \mathcal{T}, \rho, \xi')$ where G is the graph presented in Section II, $\mathcal{T}$ is the *lifetime* of the system, and ρ the *presence* function depending on $\{(p_e, q_e)\}_{e \in E}$.

Considering how environment may vary now implies new assumptions. As showed before, presence of edges becomes probabilistic. That must be taken into account by agents and implies new behavioural constraint assumptions on agents.

Mathematically, let $N : V \rightarrow \mathcal{P}(V)$ denoting the function returning, in static environment, the adjacent nodes of the current one v_0. In dynamic environment this function depends now on time, that's why it can be rewritten as $N' : V \times \mathcal{T} \rightarrow \mathcal{P}(V)$ such as $\{v_{i_0}, v_{i_1}, ...\} = N(v_0, t) \subseteq N(v_0)$.

Second, in the case in which an agent becomes blocked due to the inaccessibility of all adjacent nodes, it shall wait the next cycle that one of its adjacent nodes becomes accessible again.

Lastly, if an agent is crossing an edge, in order to remain consistent with our model which does not treat the agent disappearance for now, this edge shall not be deactivated as long as the agent is crossing it. Let $\sigma : E \times \mathcal{T} \rightarrow \{0, 1\}$ denotes the boolean function which sets whether any agent a from the agent set A is crossing the edge $e \in E$ at time $i \in \mathcal{T}$ such that:

$$\forall e \in E, \forall i \in \mathcal{T}, \sigma(e, i) = \begin{cases} 0 \text{ if } \forall a \in A, a \text{ is not} \\ \quad \text{crossing } e \text{ at time } i, \\ \\ 1 \text{ else} \end{cases} \quad (7)$$

Then, Equation 6 becomes:

$$\forall e \in E, \exists\, p_e, q_e \in [0, 1], \begin{cases} P(e \in E_{G_{i+1}} | e \notin E_{G_i}) = p_e \\ P(e \notin E_{G_{i+1}} | e \in E_{G_i} \cap \sigma(e, i) = 0) = q_e \\ P(e \notin E_{G_{i+1}} | e \in E_{G_i} \cap \sigma(e, i) = 1) = 0 \end{cases} \quad (8)$$

Equation 8 is initialised by $E_{G_1} = E$.

B. Adaptation of strategies

Let consider a perpetual changing environment. Some edges can appear or disappear at each time step t.

For that purpose changes were applied to studied strategies with the aim of satisfying this new need. In dynamic environment, for the purpose of complying with their strategy, while moving toward their goal node, at each time agents are on a node, they must check if the next adjacent node to visit is reachable. If it is not, either they run a new stage of goal node selection under their strategy or they wait.

Regarding the strategy CR, it has almost the same procedure over a varying graph than over a static one. Every time an agent chooses a new node to visit, the nodes located in its neighbourhood that are unreachable due to absence of edge leading there, are excluded from its current decision-making; selection of next adjacent node to visit coincide with selection of next goal node. Thereafter, in dynamic environment decision-making of a Conscientious Reactive agent is performed considering N' instead of N and in consequence it depends also on time because of the time-varying nature of some edges.

Adaptation of the strategy Heuristic Pathfinder Cognitive Coordinated (HPCC) is a few more complicated because agents have a plan of nodes to visit. Thereupon, if during a cycle t the next node that an agent has to visit according to its plan is not in $N'(v_0, t)$, then it cancels its present plan to request the coordinator for a new one. The coordinator computes and provides a new goal node and a new plan by assuming that the current graph will not change throughout plan.

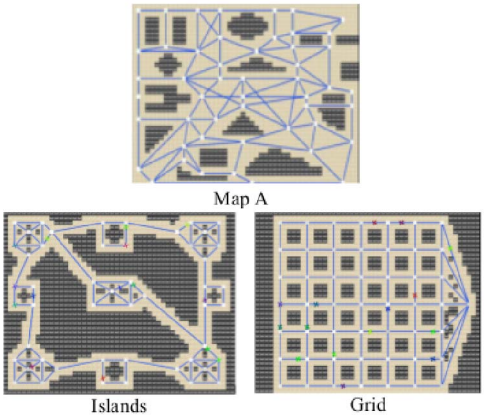

Fig. 1. Graphs used during assessment.

IV. Assessment

To evaluate the two strategies, CR the reactive one and HPCC the cognitive one, we used SimPatrol [10] presented in Section II.

A. Scenarios

Six different graphs were proposed by Almeida et al. [9] as a benchmark for the patrolling problem, which were used by Poulet [11]. In our study, we used three among them: the maps *Islands*, *Grid* and *Map A*, as shown on the Fig. 1. In each map we tested the strategies CR and HPCC, with a value of 0.5 for r, into closed system over population size of 1, 5, 10, 15 and 25 agents and for each population size we selected 30 random starts. For each setting, each strategy was tested for a total of 3000 cycles. In each map we defined approximately between 10 and 15 % of dynamic edges to model a dynamic environment. We chose this limit because experiments showed us beyond that value of variability, agents become paralysed. Practically it corresponds to 9 dynamic edges out of 84 for the topology *Island*, 10 out of 90 for *Grid* and 10 out of 106 for *Map A*.

Due to its original implementation complexity, adaptation of SimPatrol for dynamic environments have scaled up very poorly, in particular for the coordinated strategy HPCC that is more greedy strategy. For example, an instance of simulation with a population size of 25 HPCC agents may take approximately up to 8 hours regardless of the features of the used computer. This problem certainly stems from an incompressible overhead inherent to the way by which SimPatrol was implemented. This is the reason why we carried out simulation runs only upon three maps, that is the less complex maps among the six ones.

Lastly, with regard to evaluation criteria, we used normalised *MI* and *QMI*.

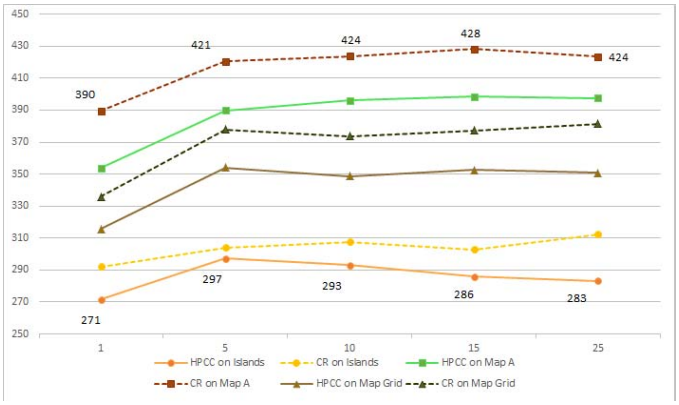

Fig. 2. Normalised MI in abscissa of the two strategies for the three maps and different numbers of agents in ordinate in static environment.

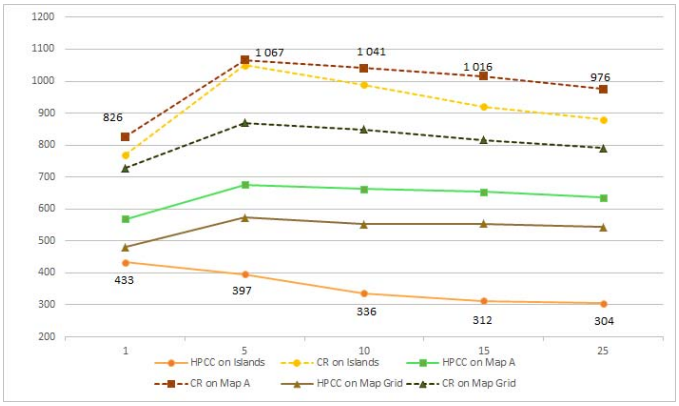

Fig. 3. Normalised QMI in abscissa of the two strategies for the three maps and different numbers of agents in ordinate in static environment.

B. Results

Fig. 2 and Fig. 3 show the normalised *MI* and *QMI*, respectively, of the strategies CR and HPCC for the three maps in static environment.

As previously stated by Poulet [11], the Fig. 2 shows that for each map the strategy HPCC has better performances than the CR one for the evaluation criterion normalised *MI*. As showed by the Fig. 3 it should be noted that the results of the evaluation criteria *QMI* is better for the strategy HPCC run on all maps than for the strategy CR run on the same maps. This result stems from the nature of *QMI* which is a measure of dispersion. HPCC being a coordinated strategy and CR a decentralised and reactive one, not surprisingly HPCC agents are better distributed over the places to visit of the map and more efficient than CR agents.

Fig. 4 and Fig. 5 show the normalised *MI* and *QMI*, respectively, of the CR and HPCC strategies for the three maps in dynamic environment.

Not surprisingly the HPCC strategy is approximately stable and it behaves like its corresponding static version showed on Fig. 2 and Fig. 3 due to the presence of the coordinator which perceives all the graph. Also HPCC is better than CR strategy for normalised *MI* and *QMI*.

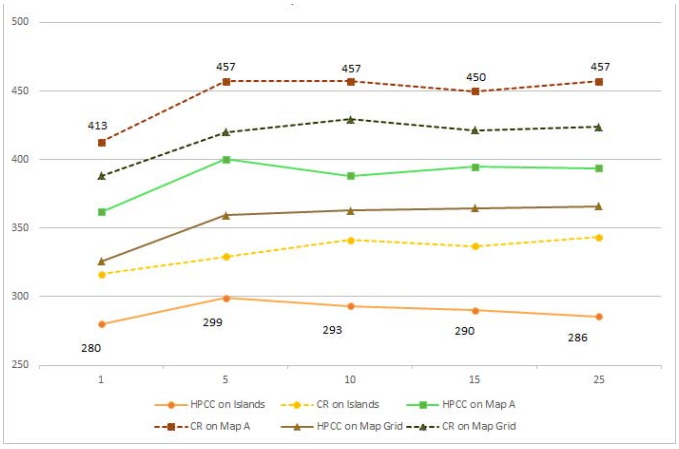

Fig. 4. Normalised MI in abscissa of the two strategies for the three maps and different numbers of agents in ordinate in dynamic environment.

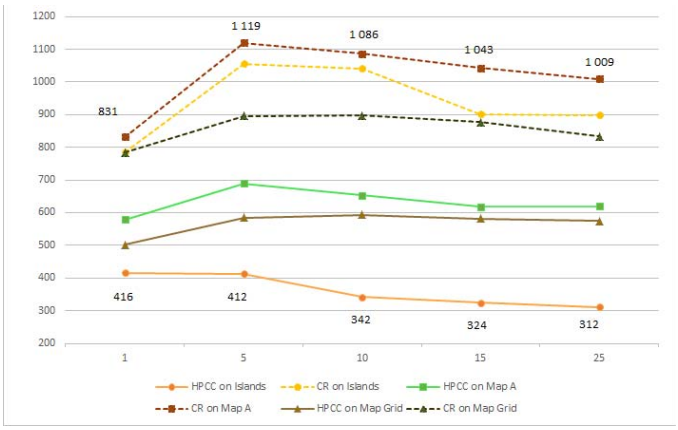

Fig. 5. Normalised QMI in abscissa of the two strategies for the three maps and different numbers of agents in ordinate in dynamic environment.

On the map *A*, CR in dynamic environment shows slightly worse performances for normalised *MI* and approximately the same one for normalised *QMI* than in static environment. For maps *Islands* and *Grid* the strategy CR is worse in dynamic environment for the normalised *MI* than in static one and approximately equals for the normalised *QMI*. On average over the set of populations, the performances of the strategy CR in dynamic environment decrease - i.e. the time increases - with respect to the static one by 7% for the map *A*, 10% for the maps *Islands* and 13% for the map *Grid*. Lastly, for all maps considered together, the performances are globally better in static environment than in dynamic one for the CR agents.

In dynamic environment, the strategy HPCC has approximately the same performances for all the maps than in static one, regarding the normalised *MI* as well as the normalised *QMI* except for the map *Grid* where the evaluation criteria in static environment are slightly better than in dynamic environment.

V. Conclusion and Perspectives

We have presented a model and empirical results about the newly introduced assumption of a dynamic environment in

the multi-agent patrolling problem. First, we have recalled the formal model of the multi-agent patrolling. Then, we have defined the model used to represent dynamic environment, which is the edge-markovian graphs, where edges appear and disappear in a probabilistic way. After adapting that model of time-varying graphs to the standard model of multi-agent patrolling, we have adapted CR and HPCC strategies and assessed them with SimPatrol.

CR, the reactive strategy, is the worst one for dynamic environments. Indeed, it is not cognitive, leading to the fact that it cannot reason over all the graph entailing and pursuit of a goal is thereupon impossible. HPCC, conversely, has turned out to be rather stable in the context of dynamic environment according to the first results. Its coordinator and more accurately its centralised nature, allow HPCC agents to quickly adapt their behaviour by reasoning - through the coordinator - on the whole graph. However this aspect needs many computing resources so that it scales up poorly. Moreover, for the motivating domain of patrolling drones in an hostile environment the exchange of information between the coordinator and the agents may not be possible.

In the future it would be relevant to study level of decentralisation from which performances become poor. As stated in Section II, CR and HPCC have been ideal-type strategies, and the interval between CR the most decentralised strategy and HPCC the most centralised, is a continuum of strategies.

Considering the problems experienced during assessment about the scalability of SimPatrol, we are currently creating a new simulator coded in Python which will be slighter than SimPatrol. This new simulator should solve scalability problems experienced during assessment. It will allow as well, the generation of a large amount of data very quickly about multi-agent patrolling task in the perspective to use machine learning methods in order to improve strategies created so far.

REFERENCES

[1] Abate, Frank R.: The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers. *Oxford Univ*. Press. 1996

[2] A. Casteigts, P. Flocchini, W. Quattrociocchi, N. Santoro. Time-varying graphs and dynamic networks. In *International Journal of Parallel, Emergent and Distributed Systems*, volume **27**, number **5**, 2012.

[3] A. Chaintreau, A. Mtibaa, L. Massoulie, and C. Diot. The diameter of opportunistic mobile networks. *Communications Surveys & Tutorials*, 10(3):74–88, 2008.

[4] A. Clementi, C. Macci, A. Monti, F. Pasquale, and R. Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of 27th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 213–222, 2008.

[5] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. (IAT 2004).*

[6] Cho J., Garcia-Molina, H.: Synchronizing a database to Improve Freshness. In *Proceedings of 2000 ACM International Conference on Management of Data (SIGMOD)*, May 2000.

[7] Machado A., Ramalho G., Zucker JD., Drogoul A. (2003) Multi-agent Patrolling: An Empirical Analysis of Alternative Architectures. In: Simão Sichman J., Bousquet F., Davidsson P. (eds) *Multi-Agent-Based Simulation II. MABS 2002. Lecture Notes in Computer Science*, vol 2581. Springer, Berlin, Heidelberg

[8] AL Almeida, PM Castro, TR Menezes et GL Ramalho. Combining idleness and distance to design heuristic agents for the patrolling task. In *II Brazilian Workshop in Games and Digital Entertainment*, pages 33–40, 2003.

[9] Almeida A., Ramalho G., Santana H., Tedesco P., Menezes T., Corruble V., Chevaleyre Y. (2004). Recent Advances on Multi-Agent Patrolling. In *Advances in Artificial Intelligence–SBIA 2004*, pages 126–138, 2004.

[10] D.H. Moreira, L.J. da Silva Filho, P.R. Tedesco et G.L. Ramalho. SimPatrol : Establishing a Testbed for Multi-agent Patrolling. 2008.

[11] C. Poulet. Coordination dans les systèmes multi-agents : Le problème de la patrouille en système ouvert. *Thèse de doctorat de l'université Pierre et Marie Curie*.

[12] P.A. Sampaio, G. Ramalho et P. Tedesco. The Gravitational Strategy for the Timed Patrolling. In *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI) 2010*, volume 1, pages 113–120. IEEE, 2010.

[13] Robert W. Floyd. Algorithm 97: Shortest path. In *Communications of the ACM CACM* Volume 5 Issue 6, June 1962 Page 345