Auction-based strategies for the open-system patrolling task

Cyril Poulet, Vincent Corruble, and Amal El Fallah Seghrouchni

LIP6, UPMC, Paris, France
`surname.name@lip6.fr`

Abstract. The Multi-Agent Patrolling task constitutes a challenging issue for the MAS field and has the potential to cover a variety of domains ranging from agent-based simulations to distributed system design.
Several techniques have been proposed in the last few years to address the basic multi-agent patrolling task. Recently, a variation of this task was proposed, in which agents can enter or leave the task at will: the patrolling task with an open system setting. A few centralized strategies were also described to address this new problem.
In this article, we propose to adapt to a dynamic population a completely decentralized strategy that was proposed for the original basic patrolling task: an auction-based strategy in which agents trade the nodes they have to visit. We describe and compare several entry and exit algorithms on various graph topologies and show the interest of basing these mechanisms on geographical proximity. Finally, we compare this strategy to the centralized ones on simulations with multiple variations in the population of agents, and show that it provides a strong stability and reactivity to changes in the population of agents.

1 Introduction

The patrolling problem is the formal modelling of the situation in which an area containing multiple points of interest need to be visited as often as possible by a team of people or agents. Two distinct variations have been studied in the past years: *adversarial patrolling*, in which the area needs to be guarded against intruders, and *timed patrolling*, where the task is to visit each point of interest as frequently and as regularly as possible. This is measured by metrics based on the temporal distribution of the visits, metrics that must be optimized.
This problem is a very interesting domain for studying coordination in Multi-Agent Systems (MAS): it is simple enough to be implemented easily, and various coordination strategies can be compared experimentally by using the chosen metrics. Indeed, the performances of the strategies are directly related to how well the agents are coordinated and share out the visits to all nodes between themselves. Yet the timed patrolling problem can model a wide variety of both real and artificial situations. In modern war-games, it can be important to patrol an area in order to detect changes in the environment. In real life, the problem can appear when agents (humans or robots) must perform repetitive maintenance or

preventive tasks (such as checking electric lines to prevent blackout). For these reasons, [7] proposed to use this problem as a benchmark for MAS.
Recently, a variation has been proposed for the timed patrolling problem: the open-system setting. In this new problem, agents may enter or leave the patrolling task at any time, thus requiring the system to adapt and reconfigure as the population of agents changes. This dynamicity allows for more complex applications, such as rescue scenarii in which a rescue agent finds a victim, and must leave the team patrolling the ruins to begin the rescue operations. Another example is when the agents performing the repetitive maintenance tasks begin or finish their shifts.
Several strategies have been proposed for the timed patrolling problem in closed-system setting, and some of them were adapted to the open-system setting. However, these adapted strategies are all centralized: they rely on a single coordinator that manages all the agents in the system. This limits scalability: as the population rises, the coordinator may become a bottleneck both in computational power (too many agents to manage) and in communication load (too many messages to and from the coordinator agent). It also proves to be very sensitive to the state of the communication network, which must be always available and reach every agent in the system. We propose here to distribute the strategy and adapt a decentralized, auction-based strategy proposed for the closed-system setting, the Flexible Bidder Agent from [6]. The auction mechanism provides here a stabilizing effect that is important to manage the dynamicity of the society. We propose several mechanisms to manage the entry or exit of agents in the system, and show that using geographical proximity provides very good results. Finally, we compare the adapted strategy to centralized ones, and show that it provides a high stability and a reactivity that are scalable.
In this paper, section 2 defines formally the patrolling task while section 3 reviews the related work. Section 4 shows how the patrolling strategy (namely auction-based strategy) could be adapted to the case of asynchronous communications. Then, it goes on to propose several "entry" and "exit" mechanisms associated with this new strategy. Section 5 presents our experimentation and the obtained results which are discussed in section 6. Section 7 concludes this paper and sketches some future work.

2 The Patrolling Problem

2.1 The patrolling task

In the patrolling problem, the environment is represented as a *graph*. Each node of this graph is a point of interest in the environment, and each edge links two neighbouring points. Building on the formal descriptions proposed in [3] and [8], we propose the following description of the multi-agent patrolling task.
The Multi-Agent Patrolling task is formally represented as a tuple $\langle G, S, M \rangle$, with G a graph, S a society of agents and M a set of metrics. The graph $G = \langle N, E \rangle$ is composed of a set of nodes N and the associated set of edges E. Each node $n_i \in N$ has a priority p_i. Each edge $e_j \in E$ has a length l_j representing the

distance between the nodes linked by e_j. G may be static or evolve over time: nodes can become accessible or inaccessible, edges can become impracticable, priorities can change. The society of agents $S = \{a_i\}_{i \in N_S}$ is a set of size N_S of agents a_i. Each agent is defined by the sets of perceptions and actions that he has access to. The available perceptions are either **related to the environment**: internal time of the simulation (P_t), the agent's position (P_{Self}), the graph (P_G) around the agent up to a distance of d_G; or **related to the society of agents**: the other agents' positions (P_{Soc}) if they are under a distance (d_{Soc}) of the agent, and the communications (P_C). The available actions are either **on the environment**: visiting the node the agent is situated on (A_{visit}) and moving toward a destination (A_{GoTo}), or **on the society of agents**: sending a message (A_C) by broadcast or to a single recipient, in both cases within a distance of d_C of the sender.
The society S can be closed - the number of agents is constant over time - or open - the agents can join or leave the society at any time. Finally, M is a set of evaluation criteria based on the temporal distribution of the visits. The multi-agent patrolling task is then the objective given to the agents of S to visit each node of G repetitively in order to optimise the set of criteria in M. As we are primarily concerned with the dynamic aspect of S in this paper, we will only consider a static graph G. We also consider each node to be equally important, thus all priorities are the same. Finally, communications are not restricted: agents can communicate as often and as far as needed ($d_C = \infty$).

2.2 Metrics

The first proposed metrics for the patrolling problem in closed-system setting was the idleness criterion $Id_i(t)$ ([5]). If t^i_{visit} is the time of the last visit on node n_i, the instantaneous node idleness is $Id_i(t) = t - t^i_{visit}$. It can then be averaged over the graph (instantaneous graph idleness $Id_G(t)$) and over time (graph idleness $Id_G{}^{t_1}_{t_2}$). However, graph idleness is difficult to link directly to the events happening during the simulation. For this reason, [9] proposed interval-based metrics: the average interval and the Mean Square Interval. These criteria are defined as follow: with N the set of nodes in the graph, $Intervals\_of\_n_i$ the set of intervals between the visits of node n_i during the simulation, $Intervals\_on\_G$ the entire set of intervals of the simulation and $|I_{n_i}|$ is the length of an interval I_{n_i} of node n_i, the MSI is:

$$MSI = \sqrt{\frac{\sum^{\{n_i \in N\}} \sum^{\{I_{n_i} \in Intervals\_of\_n_i\}} |I_{n_i}|^2}{card(Intervals\_on\_G)}}$$

With $|N|$ the number of nodes in the graph and T_{max} the number of cycles in the simulation, the average interval I_{av} is:

$$I_{av} = \frac{\sum^{\{n_i \in N\}} \sum^{\{I_{n_i} \in Intervals\_of\_n_i\}} |I_n|}{card(Intervals\_on\_G)} = \frac{|N| \times T_{max}}{card(Intervals\_on\_G)}$$

Following [9], we can prove that their criteria can be related directly to the graph idleness by the relation $Id{G_0}^{T_{max}} = MSI^2/(2I_{av}) - 1/2$. For this reason, we propose to use the average interval and the MSI instead of graph idleness, since we think that they provide complementary information: how often the visits are in average for the first criterion, and how well the visits are spread over the graph for the second.
For the patrolling problem in open-system setting, [8] proposed additional criteria to measure performances during transitional phases:

- *stabilization time*: represents the time for the system to return to a stable phase. For this, [8] propose to calculate a mean over the stable phase following the transition on one of the criterion, then to compare short averages on a hundred cycles to this value, starting from the time at which the event occurred (change in the agents number) and stopping when the short average is less than 1% different of the stable value;
- *amplitude of variations*: measures the eventual loss of performances during the transitional phase. [8] propose to use the ratio between the maximum value of the instantaneous idleness during the transition and its average value during the stable phase showing the worst performances between the stable phase before the transition, and the stable phase after.

3 Related Work

Many different strategies have been proposed for the patrolling task with a closed-system setting (e.g. [5], [1]). In this paper we will only detail those which present state-of-the-art performances and will be used as references. For the centralized approaches, two strategies stand out: the Single-Cycle agent and the Heuristic Pathfinding Cognitive Coordinated agent.
The Single-Cycle agent (SC) was proposed in [3]. It uses two types of agents: a single coordinator agent, and as many performer agents as needed/wanted. The coordinator agent calculates the minimal cycle of the graph G in a TSP-like method (P_G with d_G=∞), then distributes evenly the performer agents around this single chosen path according to their starting point (P_{Soc} with d_{soc}=∞, A_C with d_C=∞). It is clear here that the more performers there are, the better the performances are. The performer agents only follow the chosen path, visiting the nodes as they cross them. Thus they only need to know their position (P_{Self}), the position of the next target (P_G with d_G=1) and how to go to and visit a node. However, to start the performer agents at the right moment, one of them is used as reference. The others must be able to detect its passage (P_{Soc} with d_{soc}=0).
As described in [8], in an open environment the agents are redistributed after each event (agents entering or leaving the patrolling task): agents are stopped various amounts of time in order to generate spaces for the new agents, or to fill the spaces left by the leaving agents. This is the role of the coordinator to calculate for each performer agent the time it needs to wait before setting off again, and to transmit it.

It has been demonstrated that the SC strategy is the best possible on the I_{max} criterion. However, its flaws are obvious: on the first hand it is very sensible to the size of the graph (as finding a good TSP solution is NP-hard), even more if the graph is not static, and on the second hand changing the size of the population implies to partially stop the agents, thus causing important losses in the performances (see section 5.3).

The Heuristic Pathfinding Cognitive Coordinated agent (HPCC) was proposed in [5]. Again, it uses two types of agents: a single coordinator agent, and as many performer agents as needed/wanted. The role of the coordinator is to calculate, after each visit made by a performer on a node, the new target of the visiting performer agent using a combination of distance and expected idleness of the target. Once informed of its new target, the performer agent calculates the most interesting path to its new target, maximizing the idleness of the nodes visited along the path. This strategy is naturally adapted to the open environment. This strategy requires for all the performer agents and the coordinator to know the whole graph (P_G with d_G=∞) and to have a communication link to the coordinator at all time (A_C with d_C=∞).

Though the HPCC obtains the current best performances and has no obvious flaw, it is centralized, which can become a problem as the size of the task increases.

Various decentralized approaches have also been proposed. We can cite the Flexible Bidder agent in [6] which we will discuss in section 4, reinforcement learning approaches in [10] and the gravitational strategy of [9]. We can finally mention a few swarming algorithms: the Vertex-Ant Walk of [12], EVAP of [4] and CLInG in [11].

4 Adaptation of an auction-based strategy

This work is based on the Flexible Bidder Agent (FBA) strategy proposed in [6] for the patrolling problem in closed-system setting. We first propose to study its performances and limitations on stable phases when using asynchronous communication, then propose several entry and exit mechanisms for its adaptation to the open-system setting. It is worth mentioning that these mechanisms can be used for any auction based strategy, since they do not rely on the chosen auction protocol but only on the fact that the nodes are distributed among the agents.

4.1 Using asynchronous communication

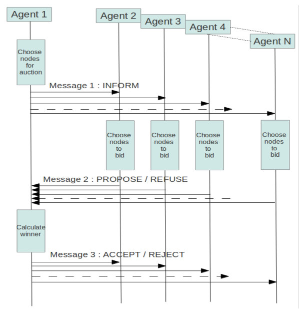
The Flexible Bidder Agent is formally described as:

$$FB\ Agent = \begin{cases} P = \langle P_{Self}, P_G(d_G = \infty), P_C \rangle \\ A = \langle A_{visit}, A_{GoTo}, A_C(d_C = \infty) \rangle \end{cases}$$

At the beginning of each simulation, the nodes of G are randomly distributed among the active agents. Each agent keeps track of its nodes, and estimates the instantaneous idleness of each one. This estimate is used to choose the next node to visit and the path to reach it, using the heuristic and pathfinding algorithms described in [1]. They can exchange nodes in order to decrease the length of the path between their nodes, trading one node for one node, one node for two or two nodes for two. The agents are selfish : only the exchanges that are beneficial to both agents are possible. In an auction, two roles appear : the initiator agent, which starts the auction by proposing nodes it want to trade, and ultimately chooses the exchange, and the participants, who propose nodes in exchange for the nodes proposed by the initiator. All agents can take the both role, but each auction has a single initiator. The auction protocol is the following (see fig. 1):

– The initiator agent (agent 1 in fig. 1) identifies 1 or 2 nodes that it wants to trade. They are those which add the most to the current path length of the agent. It then informs the other agents of the beginning of an auction via an INFORM message containing the nodes put to auction.
– The participants then choose to propose nodes in exchange (1 for 1, 1 for 2 or 2 for 2) via a PROPOSE message containing the chosen nodes, or refuse to participate in the auction (REFUSE message).
– Finally, the initiator agent reviews the propositions once it has received all the expected answers. It accepts the most interesting exchange (ACCEPT message) and rejects the others (REJECT message).

[6] made for this strategy the assumption that communication was synchronous. However, we consider that is is a very strong constraint, as it requires for each agent to be instantly available each time an auction is initiated. Asynchronous communication is a more realistic approach for multi-agent systems, as each agent is free to process the messages it receives whenever it is the most interesting or practical.
For this reason, we adapted this strategy for asynchronous communications: auctions can now be simultaneous, i.e. an agent can be both initiator and participant at the same time, in different auctions. Since auctions are in a single round (agents cannot go back on their previous offers), using asynchronous communications adds a new constraint : agents cannot propose a same node in more than one simultaneous auction (nodes are blocked until the end of the auction). This condition on nodes implies that not all possible exchanges are considered in a given auction, since some nodes are not available during its execution. Thus, whereas in the synchronous version the best possible exchange was always chosen, it is not the case in the asynchronous version. Furthermore, this best exchange may not be possible in the future, since agents only agree to mutually beneficial exchanges. This leads to an important loss in performances for this strategy. This can be seen in Fig.3 (see section 5 for the description of the maps), in which our results for the SC and HPCC strategies are similar to those of [6] (we did not study the GBLA strategy), but the loss of performances of the FBA strategy shows clearly the price of asynchronous communications. These new performances will be taken as reference for our study of transitional phases.



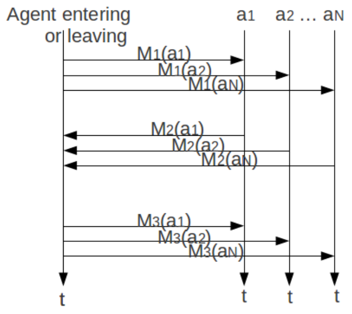


Fig. 1. Auction protocol.

Fig. 2. Entry and Exit protocol.

4.2 Entry mechanisms

We now present four entry mechanisms that allow an agent to enter a running patrolling task. The general protocol can be seen in Fig.2. The questions that were considered were : is it better to reallocate node by node, or by groups of nodes ? Is it better to favour physical proximity over more egalitarian methods ? Is it better than a random reallocation on the chosen metrics ? The specifics of the mechanisms are as follow, with N_S being the size of S *before* the arrival of a new agent, and $|a_i|$ the number of nodes of agent a_i:

- **Random mechanism**: Each agent gives randomly some of its nodes to the new agent. M_1 is a simple message signalling the arrival of the new agent. Each active agent a_i selects randomly a fraction of $|a_i|/(N_S+1)$ of its group of nodes, and sends in $M_2(a_i)$ the list of selected nodes and the estimated current idleness of each node. The new agent then adds all the received nodes to its own list of nodes, and sends acknowledge messages to all agents $(M_3(a_i) = ACK)$. When receiving the ACK message, the agents remove the selected nodes from their list;
- **Worst Nodes mechanism**: Each agent gives the nodes that are the most costly to manage. The mechanism is very similar to the Random mechanism,

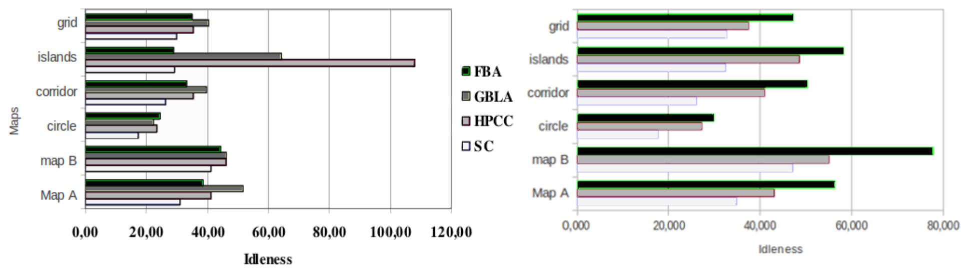


Fig. 3. Average Graph Idleness for various strategies, 5 agents. Left: results from [6]. Right: our results.

only the selection of nodes differs. Here, each agent compares the distances added by each of its node to the total distance it has to walk to visit all them, and selects the $|a_i|/(N_S+1)$ nodes that adds the biggest distances;
- **Worst Group of nodes mechanism**: Each agent gives the group of nodes that is the most costly to manage. Again, this mechanism only differs from the Random mechanism by the selection of nodes. Each agents calculates the group of $|a_i|/(N_S+1)$ nodes that adds the biggest distance to the total distance it has to walk to visit all its nodes, by comparing the distance with and without each group of nodes;
- **Proximity mechanism** Each agent close to the new agent give some of its nodes. With a distance d_{prox}: M_1 signals the arrival of the new agent, and its position. Each active agent a_i then sends at most $|a_i|/2$ nodes that are at a distance smaller than d_{prox} of the advertised position. If it has no eligible node, it sends the single node minimising the distance to the advertised position, and signals it as a second choice. Each node is sent with its estimated instantaneous idleness. Upon reception of the M_2 messages, the new agent adds all the first-choice nodes to its list of nodes. If not enough nodes were sent as first choice, it completes its list by choosing in the second-choice nodes those minimising the distance to its newly acquired nodes. Finally, it sends to each other agents an ACK message if their nodes were accepted, a REFUSE message if not. This allows the agents to remove (or not) the selected nodes from their own list.
  In our experiments, we tested d_{prox} as k times the average length of an edge in G, with k = 1, 2 or 3.

The random mechanism has been chosen as a control mechanism, whereas the others will allow us to understand if it is better to create the best group of nodes (by proximity) or to improve the existing groups by removing the worst nodes in them.

4.3 Exit mechanisms

We here describe five exit mechanisms that allow an agent to quit a patrolling task by distributing its nodes to the other agents. The general protocol is also represented in Fig.2, and the underlying questions are the same.

- **Random mechanism**: The quitting agent gives randomly its nodes to the other agents. M_1 signals that an agent wants to quit the task. Via $M_2(a_i)$, the agent a_i signals that it is ready to receive new nodes. After receiving the M_2 messages, the quitting agent distributes evenly and randomly its nodes to the N_S-1 agents remaining ($|a_{quit}|/(N_S-1)$ nodes by agents) and sends to each agent the message $M_3(a_i)$ containing the list of nodes assigned to a_i and their estimated idleness.
- **Best Nodes mechanism**: Agents choose the nodes that interest them the most in the nodes of the quitting agent. The quitting agent sends in M_1 its list of nodes to the other agents. Each agent then orders the list by increasing

cost of adding each node to its own nodes, and sends the preference list and the associated costs via M_2. The quitting agent divides up the nodes evenly between the remaining agents by order of cost: the smallest cost is found, and if the agent that sent the (node, cost) pair has not been assigned enough nodes, the node is assigned to it. The (node, cost) pair is discarded, and the process is iterated until all nodes have been assigned. $M_3(a_i)$ then brings to a_i the list of nodes it was assigned, and the estimated idlenesses.

– **Best Group of nodes mechanism**: Agents choose the group of nodes that interest them the most in the nodes of the quitting agent. This mechanism is similar to the previous mechanism, but the agents send via M_2 preferences over groups of nodes. These groups are of size $|a_i|/(N_S - 1)$, and form a partition of the original list of nodes: the agent calculates the group of nodes minimizing the added cost, removes them from the list and iterate. We decided to make a partition to avoid computing all possible groups of nodes, which would be intractable or very costly. The quitting agent then assigns the group with the smallest cost,then the group with the smallest cost and which nodes are all not assigned, until no eligible group remains. The remaining nodes are assigned randomly between the agents that were not assigned enough nodes.
– **Best Proposed Group of nodes mechanism**: this mechanism has been derived from the previous one: here the quitting agent pre-calculates groups of $|a_{quit}| \; /(N_S - 1)$ nodes by geographical proximity, and sends the list of groups in M_1. Each agent then orders this list and sends the ordered list and associated costs in M_2. The attribution of the groups is as in the previous mechanism.
– **Proximity mechanism** with a distance d_{prox}: Agents chooses nodes that are close to them in the nodes of the quitting agent. This mechanism is very similar to the best nodes mechanism, but here the agents only order the nodes that are at a distance smaller than d_{prox} from one of their own nodes. a_{quit} then assigns the nodes by smallest cost. However, the preference lists are not complete, so when no cost remains, any non assigned node is assigned by geographical proximity to the nodes already assigned. With this mechanism, the distribution is not even between the remaining agents.

Again, the random mechanism is here as a control mechanism. The other mechanisms will allow us to see if it is interesting to redistribute nodes by group or one by one. The "best proposed group" mechanism is presented as an attempt to limit the cost of calculating the best groups: only the quitting agent calculates the partition.

5 Experimentations

We used the Simpatrol simulator initiated at CIn, UFPE (Recife, Brazil) ([7]). We chose as environments the maps described in [5], which present a variety of topologies: : a random map strongly connected (50 nodes, 106 edges): map A, a random map loosely connected (50/69): map B, a circle (50/50), a corridor

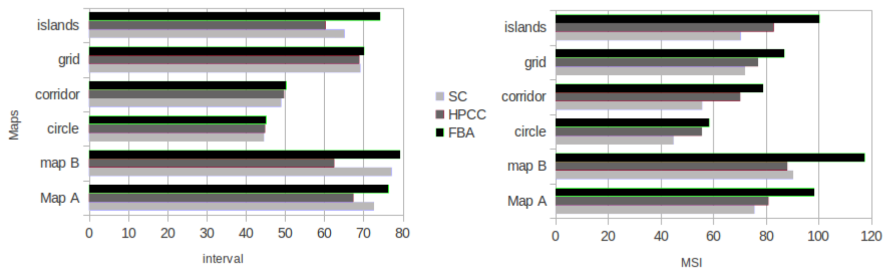


Fig. 4. Compared performances over the various graph topologies, 5 agents.

(49/50), a grid (50/90) and a map of 9 islands strongly connected inside and loosely connected between them (50/84). We will here present the experiments with a society of 5 agents, but they hold for up to 15 agents.
30 simulations were calculated from random start positions and a length of 3000 cycles, first with a closed system of 5 agents, then 4. In a second time, we ran open-system scenarios: 4→5 agents to compare entry mechanisms, and 5→4 to compare exit mechanisms. In these scenarios, the agent enter or leaves at cycle 1000, when the system has stabilized. The two stable periods (0→1000 and 2000→3000) can then be compared to the experiments with closed societies. We present in Fig.4 the stable performances of SC, HPCC and FBA for $N_S = 5$, on the average interval and average MSI criteria. Finally, we compared HPCC, SC and FBA on a larger scale scenario of 20000 cycles, with an open system which size evolves between 2 and 12 agents.
In the remainder of this work, we will call “proximity k” the proximity mechanism with a distance d_{prox} equal to k times the average length of an edge in G.

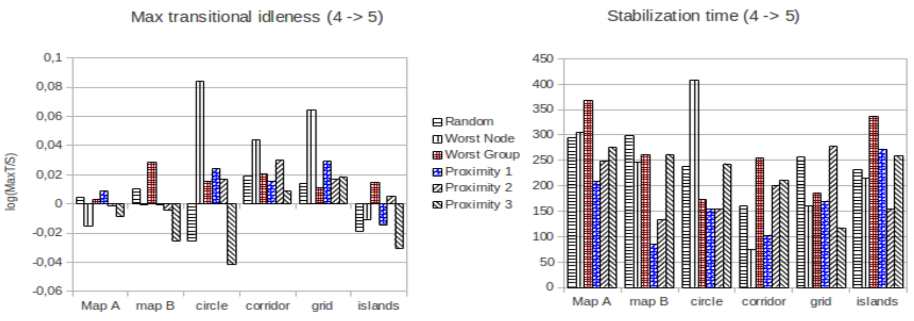


Fig. 5. Compared performances over the various graph topologies, entry mechanisms.

5.1 Entry mechanisms

A first result of our experiments is that every entry mechanism tested showed performances statistically similar to closed-system experiments on both stable phases and on every topology: within a 2% margin for the interval criterion, and 6% for the MSI. The only exception is the proximity 1, which showed a gain in performances of 10 to 15% on the interval criterion on the map A, map B and islands topologies.
Studying the transitional phase, we can see in Fig.5 that the maximum transitional instantaneous idleness criterion (i.e. the ratio between the maximum value of the instantaneous idleness and the idleness in stable phase with 4 agents, see section 2.2) clearly favours the proximity 3 mechanism, followed by the random mechanism, then by proximity 1 and 2. The worst nodes and worst group mechanisms perform worse than all the other mechanisms on this criterion. On the stabilization time criterion, the proximity mechanism is clearly quicker to stabilize, with more regularity for k=1. Here again, the worst nodes and worst group mechanisms do not show clearly better performances than the random mechanism.
Finally, a last criterion we study is the number of successful transactions needed to return to a stable phase. In Fig.6.a, proximity 1 is clearly the mechanism that minimizes this number of transactions, followed by the worst nodes and proximity 2 mechanisms. This indicates that the initial distribution is better for these mechanisms than for the others, and less nodes need to be redistributed before stabilization.

5.2 Exit mechanisms

Like the entry mechanisms, the exit mechanisms also lead to stable phases showing statistically similar performances to the closed-system experiments, with the same margins on both criteria. During the transitional phase, we can see in Fig.7 that for the maximum transitional instantaneous idleness criterion, the best group and best proposed group mechanisms show the best performances, followed by the proximity 2 mechanism. On the stabilization time criterion, the Best Proposed Group shows the best overall performances, but with a very bad

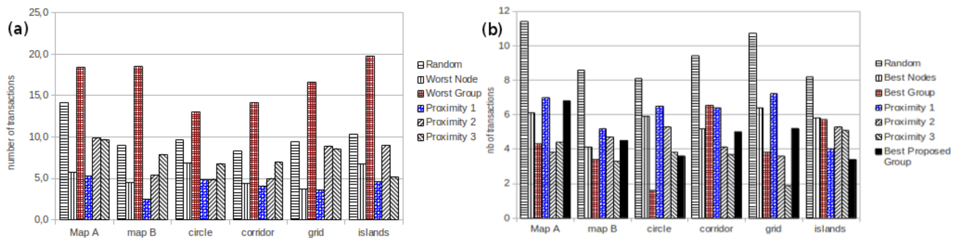


Fig. 6. Number of successful transactions in transitional phase, entry (a) and exit (b) mechanisms.

performance on the map A topology. On the meantime, the proximity 1 mechanism presents the second best overall performances, and is very regular with respect to the various topologies.
Finally, Fig.6.b allows us to see that the proximity 2, 3, best group and best proposed group mechanisms require less transactions to stabilize than the other mechanisms, but are difficult to rank. However, proximity 3 seems to provide the overall smallest number of transactions.

5.3 multi-variation scenario

Once the entry and exit mechanisms compared, we confront the modified FBA to the leading centralized strategies (SC and HPCC) presented in [8]. For this, we chose proximity 1 as entry mechanism, and proximity 2 as exit mechanism. We generated a scenario of 20000 cycles, on the map A topology. The system has 5 agents at the beginning, then each 100 cycles there is a 0.1 probability that an event happens. An event touches 1 to 4 agents (with a decreasing probability), and can be the entry or exit of these agents in the system. The probability of the type of event depends on the number of agents in the system: the more agents there are, the more probable it is that the event is an exit.
We first studied the performances on stable phases through the long simulations. Results show that on each stable phase, we have the ranking between the strategies and the orders of magnitude of the performances that were presented in Fig.3 right. We also evaluated for each strategy the drift in performances on the interval and MSI criteria on various stable phases of the scenario: the performances on each stable phase is less than 2% different from expected on the average interval criterion, and less than 10% on the MSI criterion.
Studying the transitions during the scenario, Fig.8 shows the stabilization times and maximum transitional idlenesses of the successive transitional phases. On the stabilization time, Fig.8 shows clearly that HPCC > FBA > SC. We calculated that SC spends 22% of the total duration of the scenario in transitional phases, FBA 14.5% and HPCC 8.7%. On the maximum transitional idleness, Fig.8 shows that FBA is the strategy causing the less disturbances in the performances, followed closely by HPCC. On the contrary, SC causes high losses in performances at each transitional phase.

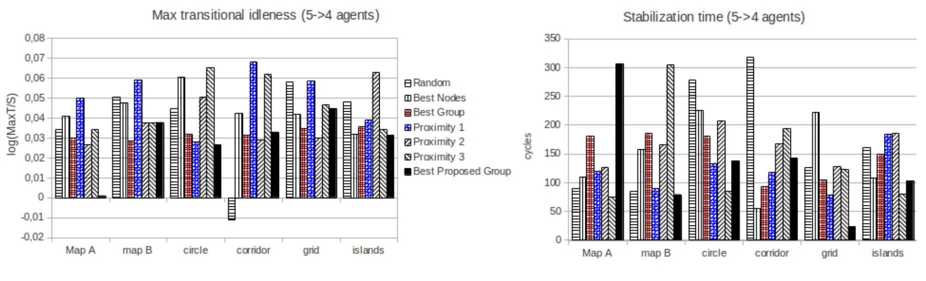


Fig. 7. Compared performances over the various graph topologies, exit mechanisms.

6 Discussions

A first point we would like to discuss is the loss in performances of the FBA strategy on stable phases due to the use of asynchronous communications. We provided in Fig.4 its performances as a reference for further experiments. As is explained in section 4.1, it is due to our choice to keep the auction protocol presented in [6], which was only possible if agents did not offer the same node in concurrent auctions. This restrains the number of possible bid, and thus the quality of the auction. As can be seen in Fig.4, the heuristic and pathfinding techniques allow each agent to keep a good average interval. However, the loss of quality in the auction mechanism leads to a situation where each agent as a few bad nodes (geographically far from the others) which could not be traded before stabilization, causing a higher MSI, and thus a higher graph idleness (see section 2.2 for the relation between the metrics). This choice was made in order to provide a first reference in decentralized strategies for the patrolling task with the open-system setting, and allow comparisons in future works.
On the entry mechanisms, it is interesting to point out that both the worst node and worst group mechanisms show poor performances both on the stabilization time and the maximum instantaneous idleness (Fig.5). This can be explained by the fact that the new agent becomes a "worst case" agent, with nodes that are far from him and scattered across the graph. It thus takes time to visit them all, or to trade them for more interesting ones. On the contrary, the proximity mechanism assigns to the new agent nodes that are spatially close to it, and thus easy to reach. It also allows the system to stabilize in fewer transactions (Fig.6.a).
Although exit mechanisms are more difficult to rank (see Fig.7), some observations can be made. First, the best proposed group mechanism performs in average as well as the best group mechanism on both criteria. It is then interesting to prefer the first over the second one, as it is less costly in computational power (one agent calculates the groups, instead of all agents but one). A second observation is that the proximity mechanism performs better than the best

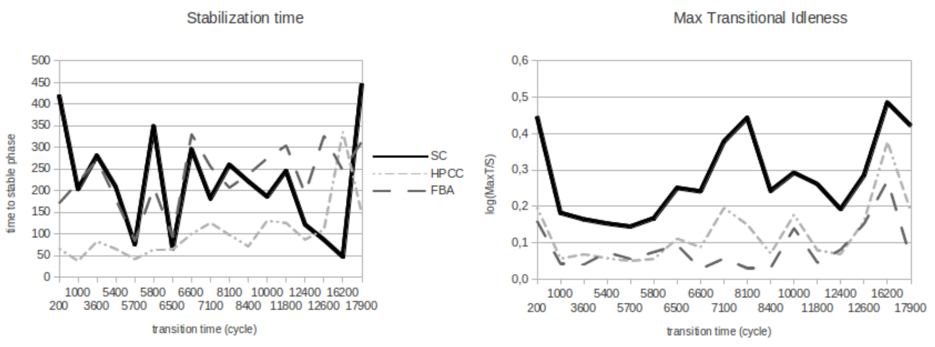


Fig. 8. Compared transitional performances during the 20000 cycles scenario.

nodes mechanism. This is due to the “even distribution” constraint (each agent gives the same amount of nodes) being lifted in the proximity mechanism, thus avoiding the scattering of the given nodes. Finally, it is worth noting that although the time scale of stabilization is the same for an entry or an exit event, the exit event causes less disturbances on the criterion of the maximum transitional idleness. This is due in part to the fact that the disturbances of an exit event is partly absorbed in the overall loss of performances due to having less agents in the system.
Finally, we would like to discuss the results of the 20000 cycles scenario. Due to the previous results and the choice of the map A topology, the chosen entry and exit mechanisms were respectively the proximity 1 and the best proposed nodes mechanisms. As evoked in section 5.3, a first interesting result is that although the auction protocol has limitations we described earlier, the performances hold on every stable phases through the whole scenario. This suggests that the various changes in the population of agents have a regulating effect on these limitations by mixing the nodes and thus allowing new auctions during the whole duration of the scenario. Another result that can be seen on Fig.8 is that the FBA strategy is faster to stabilize than the SC strategy, and that it causes very small disturbances during the transitions. On the contrary, the SC is very sensitive to changes in the population of agents, to the point of showing very poor performances when these changes are frequent (for example, see Fig.**??** between cycles 5000 and 9000).
In our opinion, this stability in the performances on stable phases and ability to reconfigure quickly and without disturbing the system are strong assets to the FBA strategy. Depending on the characteristics one requires of the system, they make of the FBA a good alternative to the SC strategy. The fact that the FBA is a completely decentralized strategy can also make it an interesting challenger to the HPCC strategy if one needs to avoid bottlenecks. We also think that by limiting the number of awaited bids in each auction, the FBA could easily be scaled up to bigger graphs, whereas the HPCC coordinator would encounter a communication bottleneck. Using time limits to auctions could also allow us to lift the constraint of perfect communication, giving the FBA another advantage on the HPCC strategy. As an indication, we calculated the communication load (the number of sent messages) for each strategy: HPCC and FBA sent an average 32000 messages during the 20000 cycles scenario (all to or from the coordinator in the HPCC strategy), against 300 for the SC strategy.

7 Conclusion

In this paper, we presented the adaptation of an existing auction-based strategy - the Flexible Bidder Agent ([6]) - designed for the patrolling task with a closed-system setting and synchronous communications to an open-system setting and asynchronous communications. We compared several entry and exit mechanisms, and chose the best of them to design a strategy fit for long scenarios. Finally, we compared this strategy to state-of-the-art centralized strategies, the Single

Cycle and the Heuristic Pathfinding Cognitive Coordinated ([8]). We showed that although showing poorer performances on stable phases (especially on the MSI criterion), the adapted FBA showed a great ability to adapt to changes in the population of agents, with a fast stabilization and little loss of performances during the transitional phases. This lead us to recommend this strategy as a good and decentralized alternative to the centralized strategies, that could furthermore be easily scaled up to bigger graphs.
In future work, we plan to improve this approach using cooperative auctions : we believe that having non-selfish agents could be a way to get around the need to block nodes during auctions, by allowing them to be put to non-beneficial auctions later on. We will try to demonstrate that auction-based strategies can achieve a level of performances similar to HPCC, without being centralized.

References

1. A. Almeida, P. Castro, T. Menezes, and G. Ramalho. Combining idleness and distance to design heuristic agents for the patrolling task. *II Brazilian Workshop in Games and Digital Entertainment*, 2003.
2. A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, and Y. Chevaleyre. Recent advances on multi-agent patrolling. *Advances in Artificial Intelligence–SBIA 2004*, 2004.
3. Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. *Proc. of the Intelligent Agent Technology, IEEE/WIC/ACM Int. Conf.*, 2004.
4. H. Chu, A. Glad, O. Simonin, F. Sempe, A. Drogoul, and F. Charpillet. Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation. *ICTAI'07 IEEE Int. Conf. on Tools with Artificial Intelligence*, 2007.
5. A. Machado, G. Ramalho, J. Zucker, A. Drogoul. Multi-agent patrolling: an empirical analysis of alternative architectures. *Lecture notes in computer science*,2003.
6. T. Menezes, P. Tedesco, and G. Ramalho. Negotiator agents for the patrolling task. *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, 2006.
7. D. Moreira, G. Ramalho, and P. Tedesco. Simpatrol - towards the establishment of multi-agent patrolling as a benchmark for multi-agent systems. *ICAART*, 2009.
8. C. Poulet, V. Corruble, A. El Fallah Seghrouchni, and G. Ramalho. The open system setting in timed multiagent patrolling. *WI-IAT'11, Proc. of Web Intelligence - Intelligent Agent Technologies 2011*, 2011.
9. G. Sampaio, G. Ramalho, and P. Tedesco. A technique inspired by the law of gravitation for the timed multi-agent patrolling. *22nd IEEE Int. Conf. on Tools with Artificial Intelligence (ICTAI)*, 2010.
10. H. Santana, G. Ramalho, V. Corruble, and B. Ratitch. Multi-agent patrolling with reinforcement learning. *AAMAS-2004 - Proc. of 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*, 2004.
11. F. Semp. *Auto-organisation d'une collectivit de robots: application l'activit de patrouille en prsence de perturbations.* PhD thesis, Universit Paris VI, 2004.
12. I. Wagner, M. Lindenbaum, and A. Bruckstein. Distributed covering by ant-robots using evaporating traces. robotics and automation. *IEEE Transactions on Robotics and Automation*, 1999.