Multi-Agent Patrolling with Reinforcement Learning[1]

Hugo Santana
Geber Ramalho
Universidade Federal de Pernambuco
Centro de Informática
Caixa Postal 7851, CEP 50732-970, Recife, Brazil
{hps, glr}@cin.ufpe.br

Vincent Corruble
Université Paris 6
Laboratoire d'Informatique de Paris VI
Boîte 169 – 4 Place Jussieu
75252 PARIS CEDEX 05
Vincent.Corruble@lip6.fr

Bohdana Ratitch
School of Computer Science
McGill University,
3480 University St.,
Montreal, Canada
H3A 2A7
bohdana@cs.mcgill.ca

Abstract

Patrolling tasks can be encountered in a variety of real-world domains, ranging from computer network administration and surveillance to computer wargame simulations. It is a complex multi-agent task, which usually requires agents to coordinate their decision-making in order to achieve optimal performance of the group as a whole. In this paper, we show how the patrolling task can be modeled as a reinforcement learning (RL) problem, allowing continuous and automatic adaptation of the agents' strategies to their environment. We demonstrate that an efficient cooperative behavior can be achieved by using RL methods, such as Q-Learning, to train individual agents. The proposed approach is totally distributed, which makes it computationally efficient. The empirical evaluation proves the effectiveness of our approach, as the results obtained are substantially better than the results available so far on this domain.

1. Introduction

Patrolling is literally "the act of walking or traveling around an area, at regular intervals, in order to protect or supervise it" [1]. Performing this patrolling task efficiently can be useful for various application domains where distributed surveillance, inspection or control is required. For instance, patrolling agents can be used for helping administrators in the surveillance of failures or specific situations in an Intranet [3], for detecting recently modified or new web pages to be indexed by search engines [6], for identifying objects or people in dangerous situations that should be rescued by robots [13], etc.

Despite its high potential utility, only recently this problem has been rigorously addressed. Our research group, in particular, has done pioneering work on this matter. Initially, in [10], we presented an original in-depth analysis of the patrolling task issues, and proposed different multi-agent-based solutions which were empirically evaluated on a simulator that we developed. Later, in [2], more sophisticated (multi-agent-based but non-adaptive) solutions were proposed and evaluated, and more complex instances of the problem were considered. More recently, colleagues addressed the problem of performing this task with real robots [15].

Although many of the previously developed solutions showed good empirical results, we noticed that for each proposed architecture, there were always some problem settings (e.g., a particular environment topology) on which it performed badly [2]. One of the reasons is that, for this domain, it is very difficult to design beforehand a general distributed strategy, since the task necessitates a fair amount of topology-dependent coordination between the agents' actions. Thus, adaptive machine learning techniques can be very useful in this respect. In fact, some of them have been previously used with success in other multi-agent domains, such as robotic soccer [16], as a way to automatically achieve coordination.

In this paper, we investigate the creation of adaptive agents that learn to patrol using reinforcement learning techniques [17]. The use of such techniques, in this case, is not straightforward. In order to use most of the RL algorithms, it is necessary to model this task as a Markov Decision Process (MDP), but many characteristics of this domain, such as the multi-agent setting, make it difficult to do so. One of the challenges, for example, lies in the definition of an appropriate model of the instantaneous rewards that would be conducive for achieving a high long-term collective performance. Such difficulties, and the ways to overcome them, are discussed in this work. Two RL-based agent architectures, using different

[1] Supported by the Brazilian-French project "capes-cofecub 371/01"

AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

communication schemes, are proposed, implemented, empirically evaluated and compared to non-adaptive architectures from our previous work. The optimality of these solutions is also analytically studied.

The remainder of this paper is structured as follows: Section 2 defines our task and revises previous work. Section 3 summarizes the reinforcement learning concepts. Section 4 presents a model of this task as a RL problem. Section 5 shows the experimental results, and Section 6 provides our conclusions and future work.

2. The Patrolling Task

In order to obtain a more general, yet precise, definition of the patrolling task, we adopted a more abstract representation of the terrain being patrolled: a graph, where the nodes represent specific locations and the edges represent possible paths, as shown in Fig. 2. This abstract representation can be easily mapped to many different domains, from terrains and maps to computer networks, for instance. Given a graph, the patrolling task studied in the remainder of this paper consists in continuously visiting its nodes.

Intuitively, a good patrolling strategy is the one that, for each node, minimizes the time lag between two visits to the same node. However, to be more precise, our previous work suggests some *evaluation criteria* for patrolling strategies, using the notion of *idleness* [10]. Considering that a cycle is a single simulation step, the *instantaneous node idleness* for a node *n* at a cycle *t* is the number of cycles elapsed since the last visit before *t* (number of cycles *n* remained unvisited). The *instantaneous graph idleness* is the average instantaneous idleness over all nodes in a given cycle. Considering long-term performance criteria, the *average idleness* is the mean of the instantaneous graph idleness over a t-cycle simulation. In the same context, another measure is the *worst idleness*: the highest value of the instantaneous node idleness encountered during the simulation.

With reasonable strategies, performance with respect to these measures tends to improve as the number of agents grows. However, the lack of appropriate coordination may diminish the improvement expected from insertion of new agents. In order to assess coordination quality, we can measure the individual contribution of each agent by normalizing these criteria:

$$normalized\_value = absolute\_value \times \frac{number\_of\_agents}{number\_of\_nodes} \qquad (1)$$

2.1. Overview of Previous Work

Our first work on this task considered the problem of patrolling in non-weighted graphs (distance between adjacent nodes is one) [10]. The implemented solutions were simple, but covered several multi-agent architectures with varying parameters such as agent communication (allowed vs. forbidden), coordination scheme (central and explicit vs. emergent), agent perception (local vs. global), etc. Then, we considered the real distances between nodes, representing them as weights on the edges of the graph [2]. With this representation, we explored more sophisticated (non-adaptive) solutions that employed several heuristics based on node idleness values and path lengths, combined with negotiation mechanisms [11]. These studies improved the previous results, and we use them in this work as a baseline for evaluating our adaptive agents in Section 5. Recently, [15] addressed the problem of patrolling with robots, and challenges inherent to the complexity of the real world, such as the necessity for robots to recharge, were dealt with.

One of the lessons learned from these work is that the solutions were too sensitive to variations on problem settings [2]. This conclusion led us to look for machine learning techniques, such as reinforcement learning, in an attempt to build a more general solution. In the next section, we review the theory of reinforcement learning, and the current efforts on its use in other cooperative multi-agent domains.

3. Reinforcement Learning

Reinforcement learning is often characterized as the problem of "learning what to do (how to map situations to actions) so as to maximize a numerical reward signal" [17]. This framework is mostly defined over the theory of Markov Decision Processes (MDP), which we briefly review in the following sections.

3.1. Markov Decision Processes

Consider an agent that sequentially makes decisions in an environment. At each step, this agent chooses an *action* from a finite set *A*, based on a *state* signal from the environment. This state comes from a finite set *S*, and summarizes the present and past sensations in a way that all relevant information is retained [17]. The process' dynamics is described by two components: the state transition probability distribution, *P,* and the expected *immediate reward* function, *R*. Both of them are defined on triples *<s,a,s'>*, where *s* is the current state, *a* is the agent's action and *s'* is the next state. Both functions are assumed to satisfy the Markov property. In summary, a *finite Markov Decision Process*, is specified by a tuple <*S*, *A*, *P*, *R*>, of the elements defined before. In this formalism, the agent acts according to some policy π(s,a), representing the probability of choosing action a ∈ *A* at state s ∈ *S*. The main goal is to maximize a long-term performance criterion, called return, which is often defined as a sum of the discounted rewards. The agent

AAMAS'04, July 19-23, 2004, New York, New York, USA.
Copyright 2004 ACM 1-58113-864-4/04/0007...$5.00

then tries to learn an *optimal policy* π^*, which maximizes the expected return, called the value function, $V^{\pi}(s)$, as shown in (2), over the set of all possible policies:

$$V^{\pi}(s) = E_{\pi}\left\{\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1} \mid s_t = s\right\}, \qquad (2)$$

where γ is a *discount factor* $0 \leq \gamma < 1$, and r_t is the immediate reward received at step t.

Similarly, we can define the *action-value function*, $Q^{\pi}(s, a)$, as the expected return when starting from state s, performing action a, and then following π thereafter. If one can learn the optimal action-value function $Q^*(s, a)$, an optimal policy can be constructed greedily: for each state s, the best action a is the one that maximizes Q.

3.2. Q-Learning

Q-Learning [17] is a traditional RL algorithm for solving MDPs. Skipping technical details, it consists in iteratively computing the values for state-action pairs, online, using the following update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \cdot V(s') - Q(s,a)\right] \qquad (3)$$

where $V(s') = max_a Q(s', a)$, and α is a learning rate.

This algorithm is guaranteed to converge to the optimal Q function in the limit, under the standard stochastic approximation conditions. Note that a priori knowledge about the process' dynamics is not necessary.

3.3. Semi-Markov Decision Processes

Conventional MDPs do not involve temporal abstraction or temporally extended actions [18]: the unitary action taken at time t affects only the state and reward at time $t+1$. *Semi-*M*arkov Decision Processes* (SMDPs) can be seen as a more general version of the MDP formalism, where actions may take variable amount of time. There are extensions of MDP algorithms, such as Q-Learning, which can also solve SMDPs in tractable time. SMDPs are more suitable for modeling our patrolling task, as will be shown on next sections.

3.4. Cooperative Multi-Agent Reinforcement Learning

As the MDP theory only deals with the single-agent case, one can try to consider the whole multi-agent system as a centralized single-agent, but this solution is intractable as it has an exponentially large number of actions. In contrast to this approach, many solutions use RL agents as independent learners [11], however, having no guarantees of achieving a satisfactory global behavior. The issue of creating cooperative agents that learn to coordinate their actions through RL has been studied in the current literature [5][9][7][19]. Although the problem of finding a globally optimal solution for a cooperative group of agents with partial information is known to be intractable in theory [4], many proposed approaches have achieved good results in practice.

The main difficulty is that, when a group of agents is solving a common task in a distributed manner, if each agent tries to optimize its own rewards, this does not necessarily leads to a globally optimal solution. In this context, [19] investigates the design of a more collective intelligence through the use of different utility functions. In the *Wonderful Life Utility* (WLU), the agents optimize a private utility that is aligned with the global utility. In other words, the utility functions for each agent assure that they do not work in cross-purposes. This approach was implemented in the past for some tasks by including penalties when more than one agent competed for the same reward. This utility has been compared to others, such as the *selfish utility* (SU), where each agent tries to maximize its own utility, and to the *team game utility* (TG), where the global performance is given to each agent as a reward. The TG seems to suffer from very poor learnability, as each agent's actions contribute little individually to the global reward/penalty received.

From the communication point of view, distributed RL approaches can be classified into three major groups, according to the information available to each agent [11]. In Black-Box systems, the agents do not know about the actions of other agents, the effect of which is perceived as part of the environmental stochasticity. In White-Box systems, also known as joint action learners, each agent knows about the actions of all other agents. Finally, in Gray-Box systems, agents can communicate their actions, or their intentions for future actions. In this paper, we consider Black-Box and Gray-Box architectures, with the latter exhibiting a superior performance.

4. Patrolling Task as a Reinforcement Learning Problem

Some characteristics of this task can cause a huge impact on the difficulty of building the MDP model. Because of this, we discuss our model incrementally, from simpler to more complex instances. While we discuss it, we introduce the main aspects of our approach.

4.1. The Simplest Single-Agent Case

Consider a *simplified* patrolling task, in which there is only a *single-agent*, and the terrain abstraction consists of a *graph without weights* (unitary distance between nodes). For this task to be considered an MDP, it is necessary to define the tuple <*S, A, P, R*>, as stated in section 3.1.

A natural design for the action space A is a set of actions that let the agent navigate between adjacent nodes in the graph. In other words, with one action, the agent can traverse one edge on the terrain abstraction.

Although it is not necessary to define *P* for a model-free algorithm like Q-Learning, the state transition probabilities can be easily obtained, as with a single agent our environment is completely deterministic.

To properly define a reward function, *R*, we should take into account the evaluation criteria that we would like to optimize. The agents created in this work were designed to optimize a single criterion: the *average idleness criterion*, which turned out to be very natural for the RL setting. Being *I(t)* the instantaneous idleness at cycle *t*, then, the average idleness, *M(T),* after a T-cycle simulation, is represented as follows:

$$M(T) = \frac{\sum_{t=0}^{T} I(t)}{T} \tag{4}$$

Let us denote by Pos(*t*) the node visited by the agent at cycle *t* and by Φ(Pos(t),t) the idleness of this node at cycle *t.* Considering that the idleness of each of the *n* nodes increases by one at each cycle of the simulation (if the node is not visited), the agent can easily calculate the value of *I(t)* with the following recurrence relation:

$$I(0) = InitialIdleness\,; \qquad I(t) = I(t-1) + \frac{n - \Phi(Pos(t),t)}{n} \tag{5}$$

From equation (4), it can be seen that if the sum of the instantaneous idleness is minimized, consequently, the average idleness is also minimized. A plausible reinforcement (actually, a punishment), would be the instantaneous idleness of the graph *I(t)* after each agent's action, calculated using (5). Then, if the return is defined by a sum of *discounted* rewards, such a model would not be optimal (as equation 4 is not discounted over time), but would yield an approximation. Actually, to seek for optimal performance with this reward function, we should consider RL methods that optimize the value function relative to the average expected reward per-time-step, such as R-Learning [17]. However, we do not deal with such methods in this paper, for two main reasons. First, their use would not be a reasonable starting point for our task, as there is still very little experience with these methods in the community. Second, although using *I(t)* as a reward function is useful enough for the single-agent case, it makes a huge assumption that would make it difficult to generalize to the multi-agent case. Namely, it assumes that the agent has a complete model of the whole environment (it encapsulates the model of the idleness of the whole graph). In the single-agent case, this works, because the environment is only modified by the agent itself. In the multi-agent case, communication between the agents or with a central information point would be necessary to form such a reward signal. We show now how a more general reward function can be obtained.

Expanding Eq. (5), and considering that *InitialIdleness* is zero, a closed-form expression can be given by:

$$I(t) = \frac{(t \times n) - \sum_{i=0}^{t} \Phi(Pos(i),i)}{n} \tag{6}$$

Substituting (6) in (4), *M(T)* is given by:

$$M(T) = \frac{\frac{n \times (T + T^2)}{2} - \sum_{i=0}^{T} \left((T-i) \times \Phi(Pos(i),i)\right)}{n \times T} \tag{7}$$

In Eq. (7), we notice that the only term that depends on the agent's policy is:

$$\sum_{i=0}^{T} \left((T-i) \times \Phi(Pos(i),i)\right) \tag{8}$$

Thus, if the agent can learn a policy that maximizes (8), such policy is optimal for the average idleness criterion. If we use the function Φ(Pos(t),t) as a reward function, the policy learned by Q-Learning would be the one that maximizes the sum of the discounted rewards:

$$\sum_{i=0}^{T} \left(\gamma^{i} \times \Phi(Pos(i),i)\right) \tag{9}$$

The difference between (8) and (9) is that (8) is discounted over an arithmetic progression, while (9) is discounted over a geometric progression. By using an appropriate value for γ, we can approximate (8), achieving a policy that is optimal in many cases. Furthermore, this reward function is entirely local, depending only on the idleness of the node currently being visited by the agent, and it does not need to assume anything about the rest of the environment.

At first sight, it could seem contradictory to minimize the average idleness of the graph by maximizing the idleness values of the visited nodes. But intuitively it means that, if the agent's reward is the idleness of the node that it visits, it will try to visit the nodes with highest idleness, thus, decreasing the global idleness, as shown.

Having defined *A*, *P* and *R*, we are left with the definition of the state space *S* to obtain a complete MDP model. We would like our state representation to be such that the resulting model has the Markov property, so that the choice of the best next node to visit could be made only based on the current state and independently of the whole history of positions of the agent. Past positions are important for future decisions only in that they impact the idleness of the nodes on the graph. Thus the state of the environment can be fully captured, with the Markov assumption, by the *current position* of the agent and the *idleness* (the number of cycles) of each node on the map. However, as the idleness values are not bounded, this state space is not finite. One possible solution is to group idleness values into a finite set of values, but that would still lead to a great number of states. If *k* is the number of considered levels and *n* is the number of nodes in the

AAMAS'04, July 19-23, 2004, New York, New York, USA.

graph, there are $n \; x \; k^n$ possible states, which grows exponentially in the number of nodes. This issue becomes even more problematic with real-valued distances between nodes. We discuss it in the next sections.

4.2. Considering the Real Distance

In the formulation of the patrolling task, as presented in the previous section, we considered that every edge on the graph had a length of one. Intuitively, the problem with the case where edges may have arbitrary lengths is that, if the agent is trying to maximize rewards over time, it should avoid actions taking longer time intervals, since during such periods it does not receive any rewards (the idleness of the graph only grows).

The following example illustrates the problem: Consider the nodes A, B and C in Fig. 1. Suppose that an agent is at 1 and has to decide between the path (A, B, C) (non dotted) or the path (A, C, B) (dotted), considering the utility of each path as the return estimated using Q-Learning, for example. Using the last reward function defined in section 4.1, considering that all idleness values were zero initially, and $\gamma = 0.8$, we would have: Reward (A,B,C) $= 0 + \gamma \cdot 6 + \gamma^2 \cdot 12 = 12.5$, and Reward (A,C,B) $= 0 + \gamma \cdot 10 + \gamma^2 \cdot 16 = 18.2$. The agent would, though, choose the path (A, C, B), but if we use equation (7) to calculate the average idleness (considering that the function Φ is zero while on an edge), we see that the path (A, B, C) is actually better. In fact, this path takes only 12 cycles to be executed, while the dotted one takes 16, and the agent does not take this into account.

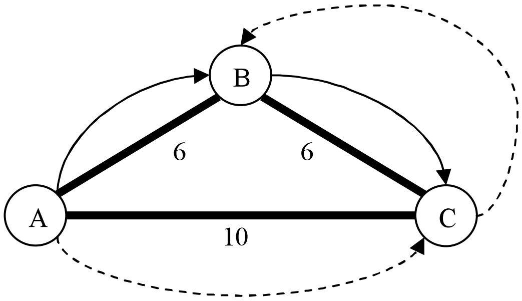

Fig. 1. Weighted graph for patrolling. Agent (at A) has to choose path (A,B,C) or (A,C,B)

Fortunately, the Semi-Markov Decision Process formalism [18] already deals with this issue. By using a discrete-time finite SMDP, the γ factor can be discounted over all time steps, instead of being discounted only at the times of the actions' final outcomes. In this case, for our previous example, the new value function, calculated using equation (2), would be: Reward(A,B,C) $= 0 + \gamma^6 \cdot 6 + \gamma^{12} \cdot 12 = 2.4$, and Reward(A,C,B) $= 0 + \gamma^{10} \cdot 10 + \gamma^{16} \cdot 16 = 1.52$, thus yielding a correct policy.

If the function Φ(Pos(t), t) is redefined to be the idleness of the visited node at time *t*, if the agent visited any node, and zero otherwise, all equations defined in section 4.1 remain unchanged, and the model developed earlier naturally generalizes to weighted graphs.

4.3. The Multi-Agent Case

Our approach for extending the model presented in the previous sections to the multi-agent case is based on the concept of independent learners [11]: we try to solve our global (collective) optimization problem by solving local (individual) optimization ones. In the rest of this section, we describe how this can be done.

The action set, *A,* can remain the same as in the single-agent case. However, there are some problems regarding the rest of the items.

By the same arguments that showed that an optimal policy with respect to the average idleness criterion is the one that maximizes equation (8), we can generalize this result to the multi-agent case: if *G* is the number of agents and $Pos_j(t)$ is a function that tells the position (node) of a specific agent j, $1 \leq j \leq G$, we want now to maximize:

$$\sum_{j=1}^{G}\left(\sum_{i=0}^{T}\left((T-i)\times\Phi(Pos_j(i),i)\right)\right) \qquad (10)$$

Let us consider that each agent of the group uses the same reward function as in the single-agent case: $\Phi(Pos_j(t),t)$, that is, the idleness of the nodes they visit. Consequently, each of them will try to independently maximize an internal summation from (10), in the same way as shown in section 4.1.

However, as the value of *Φ*, for any node, can be affected potentially by the policy of any agent *j*, this approach is selfish (each agent will do no effort to help the other agents to maximize theirs rewards) and even that we had a Markovian representation for each agent, it would not guarantee a globally optimal solution. Recalling from section 3.4, this result is equivalent to the *selfish utility.* We can also adapt this reward model with the concept of *wonderful life utility* in the same way as [8], by giving penalties when agents compete for idleness (rewards) on the same node.

Although the model presented in this section has no guarantees of global optimality, it is perfectly well-suited for a distributed implementation: the only feedback that each agent receives is the idleness of the node currently being visited. This reward is completely local, and can be implemented by a very simple scheme of flag communication: for every node that the agent visits, it places a flag containing the number of the cycle that it has been there, and this flag can be seen by other agents.

Analyzing now the state space *S*, as the single-agent case was already intractable, the same can be said for the multi-agent case. With that in mind, we decided to

AAMAS'04, July 19-23, 2004, New York, New York, USA.

represent partial information on the state of our agents. In particular, we represent a few characteristics of its surroundings, analyzing empirically the effect of each new characteristic added to the state vector, as will be detailed later. Although this is a simplification, it has some advantages. First, this formulation is cheaper in terms of computational complexity. Second, it is more realistic than the one proposed to the single-agent case, in which the agent needs to have access to the state of the whole world, which is not possible in most practical instances of patrolling. Third, it is well-suited for a distributed implementation, since each agent can only observe its immediate surroundings.

Considering the state transition probabilities P on the multi-agent case, now each agent faces a non-deterministic environment, if it does not know the actions that the other agents will perform. Worse than that, since all agents are adapting their behavior simultaneously, P will be a *non-stationary* distribution. This problem can make our attempt to learn by reinforcement useless, since we would not have a MDP or a SMDP. In order to diminish the side effects of this non-determinism, we investigate architectures using different schemes of communication, which we detail in the next section.

4.4. Agents in more Details

Two agents were modeled in this work, using the different communication schemes explained in section 3.4: a) a Black-Box Learner Agent (BBLA), which communicates only by placing a flag in each node visited. This flag can be seen by the other agents and considered in theirs state space representation. b) a Gray-Box Learner Agent (GBLA), which communicates also by flags, but can communicate their intentions of actions. Including these intentions in the state representation of each agent, the non-stationarity of the environment is diminished.

We did several experiments in order to achieve a satisfactory state representation, doing our best effort to balance the complexity of the state representation with the performance of the agent. After these experiments, the set of possible states for the BBLA was defined to be a vector of the following characteristics (consider m as the maximum node connectivity, and n the number of nodes): i) the node in which the agent is - n possible values; ii) the edge from which it came from, informing the agent about its past actions – m values; iii) the neighbor node which has the highest idleness – m values; iv) the neighbor node which has the lowest idleness – m values.

The set of possible states for the GBLA is the same as the BBLA, plus some information that comes from the communication of their intentions: v) the adjacent nodes which are intended to be visited by other agents – 2^m possible values. To implement this, after each decision step, each agent broadcasts a message containing information about the node it plans to visit. When another agent receives a message containing a neighbor node, it includes this information in its state. If communication is expensive, these messages do not need to be broadcasted to all agents, but only to those that are adjacent to the node informed in the message. With this representation, each BBLA has about 6.250 possible states for the Maps A and B shown on Fig. 2, and each GBLA about 200.000 possible states.

5. Experimental Results

A simulator was developed to evaluate patrolling strategies, and different test instances (maps) were created; two of them are shown in Fig. 2. Map A has few obstacles (highly connected). Map B has the same number of nodes, but fewer edges.

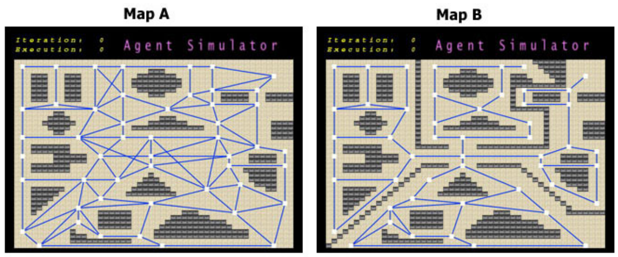

Fig. 2 - Maps A and B from our simulator. Black blocks in the maps represent obstacles, edges represent possible paths

Our experimental results are divided in two main steps. First, we do some preliminary experiments comparing WLU and SU rewards, with a fixed architecture (BBLA). Second, having defined a proper reward model, we evaluate the BBLA and GBLA architectures, comparing them with non-adaptive architectures. The team game utility (TG) was not compared, as initial experiments showed very poor results and no convergence on training.

5.1. Preliminary Results: Comparison of Reward Models

We measured the average idleness for a population of 10 agents, after 6 millions training iterations (with estimated parameters), using non-weighted graphs for simplification. The results (see Fig. 3) show that SU perform better than the WLU, particularly due to instability of WLU performance. Although surprising, as both BBLA and GBLA include little information about other agents in its state, we did not expect significant improvements in the asymptotic performance of WLU. However, we will investigate in the future the use of WLU with a more complete state representation. It is

AAMAS'04, July 19-23, 2004, New York, New York, USA.

interesting to notice also that GBLA_WLU performs worse than BBLA_WLU, showing that there is no point in using WLU with GBLA, as the communication of intentions allows agents to naturally learn to deviate from potential conflicts with other agents if these conflicts are not rewarding for them.

As the results yet did not justify the use of WLU, the experiments on next section were all done using SU, and compared the gain from communicating intentions (GBLA to BBLA) and adaptive possibilities (BBLA and GBLA to previous non-adaptive work).

5.2. Best Agents Performance Comparison

Our agents were implemented using a tabular implementation of Q-Learning, with each Q-Table learned individually, and the ε-Greedy method for balancing exploration and exploitation [17]. Some preliminary experiments were done to achieve a satisfactory set of values for each of the learning parameters, such as the decay for the learning rate, discount factor and exploration probability. The final values of these parameters are shown on Table 1.

Learning Rate	Discount Factor	Exploration Probability
$\left(2+\frac{\#visits(s,a)}{15}\right)^{-1}$	0.9	10%

Table 1. Parameters estimated for training phase

For each population size (2, 5, 10, 15, 25 homogeneous agents), we simulated 6 million cycles for training, with all agents in the population learning simultaneously, and 23.000 cycles after the training, with no learning or exploration, for evaluation. This process was done for the BBLA and repeated exactly for the GBLA. This result is compared with the same experiment done with the best previous architectures: the Conscientious reactive [10] and the Heuristic Coordinator [2]. Fig. 4 shows the results for the average idleness, normalized according to (1). These results were run on Map B from Fig. 2.

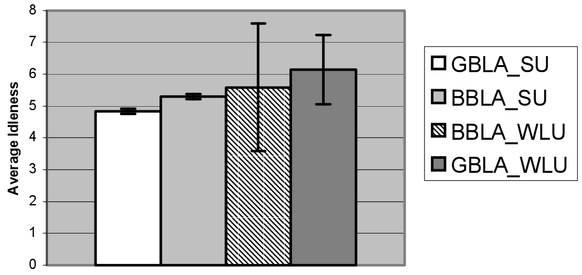

Fig 3. Comparison of Reward Models

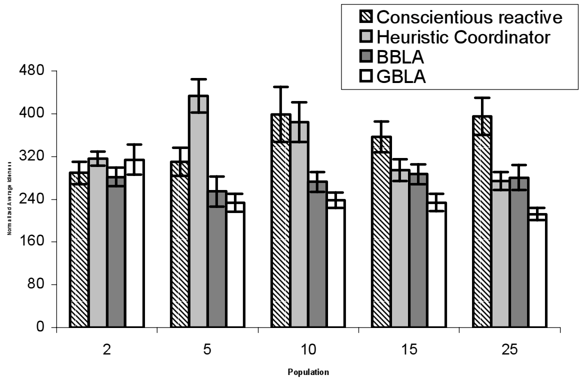

Fig 4. Average idleness comparison, normalized according to equation (1)

5.3. Discussion

Analyzing the experimental results from the last section, it can be seen that both adaptive solutions are superior to the other solutions in most of the experiments, in terms of the average idleness criterion. The GBLA can reach a performance that is more than 35% better than the best non-adaptive, for a population of 10 agents, and performs always better for populations of more than 2 agents. This is a clear example of an emerging coordination, which can be stated when observing the agents on our simulator: we can see that they are able to subdivide the graph into sub-regions, and each agent becomes responsible for one of these regions (bigger regions sometimes are patrolled by more than one agent). The BBLA also showed performance better or equal to the best non-adaptive technique in all cases, but it relatively decreases performance as the population grows and the noise from other agent's actions is augmented. As the reward model used was developed to optimize the average idleness criterion, both architectures showed worse results than the non-adaptive techniques when compared on the *worst idleness* criterion, which was already expected.

Observing the agents behavior on the simulator, we can see the consequences of our selfish formulation of the learning task: sometimes, an agent "invades" an area that is already being patrolled by another agent, seeking a better area for itself, at the price of a loss in global performance. We can observe also that when there is a small population (1 or 2 agents), although they are able to learn coherent paths, the fact that they have only local information in theirs state representation (thus not satisfying the Markov property), makes them "forget" to visit some areas for a while, decreasing performance.

Besides the good results, these architectures have two attractive characteristics: first, they are distributed, differentiating them from the Heuristic Coordinator one,

AAMAS'04, July 19-23, 2004, New York, New York, USA.

which is centralized and assumes full accessibility to the whole world. Second, they have the ability to learn and adapt continuously to the environment, a desirable characteristic in many patrolling instances.

6. Conclusions and Future Work

This work investigated several alternatives for the use of reinforcement learning on the multi-agent patrolling task and represents the first successful effort in applying an adaptive strategy in this context. The applied RL algorithm, Q-learning, was operating in an unusual context, namely in a non-stationary environment, caused by the simultaneous and independent adaptation of all the agents. Notwithstanding this, no unstable behavior was observed and the learning process converged to very good solutions. The approach required very little intervention into the design of the coordination strategy, as the coordinated behavior emerged mainly automatically as the result of collective learning. The distributed nature of this approach made it computationally very efficient and thus appealing for real-time applications. We gave a detailed treatment of the issues involved in modeling the patrolling task in the RL framework and the analysis of the optimality of its solutions, in particular for the average idleness criterion. The insights presented can be equally valuable for other problems with similar properties. This work constitutes a valuable positive case study for similar applications of standard RL techniques in multi-agent systems.

In the future, we plan to experiment with other designs of the reward function and state representation to investigate whether an added complexity and communication would result in significant increase of performance. To take full advantage of adaptation capabilities, we also intend to experiment in more dynamic environments, where the graph topology changes and regions have different patrolling priorities, possibly changing over time. Finally, we are developing an architecture, aiming to achieve a globally optimal performance, through explicit coordination of the agent's actions, using an extension to the Coordinated Reinforcement Learning method [7].

References

1. Abate, F. R.: The Oxford Dictionary and Thesaurus: The Ultimate Language Reference for American Readers. Oxford Univ. Press. 1996
2. Almeida, A. Patrulhamento Multi-Agente em Grafos com Pesos. Dissertação de Mestrado. Universidade Federal de Pernambuco, 2003.
3. Andrade, R. de C., Macedo, H. T., Ramalho, G. L., and Ferraz, C. A. G.: Distributed Mobile Autonomous Agents in Network Management. Proceedings of International Conference on Parallel and Distributed Processing Techniques and Applications, 2001
4. Bernstein, D., Zilberstein, S., Immerman, N. The Complexity of Decentralized Control of Markov Decision Processes. In Proceedings. of the 16th Conf. on Uncertainty in Artificial Intelligence, 2000.
5. Chalkiadakis, G., Boutilier, C. Coordination in Multiagent Reinforcement Learning: A Bayesian Approach. Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-03), pp. 709-716, Melbourne, 2003.
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. Guestrin, C., Lagoudakis, M., Parr, R. Coordinated reinforcement learning. Proceedings of the 19th International Conference on Machine Learning (ICML-02), pages 227-234, Sydney, Australia, July 2002.
8. Hoen, P.J., Bohte, S.M. COllective INtelligence with Sequences of Actions: Coordinating actions in Multi-Agent Systems. European Conference in Machine Learning (ECML), 2003.
9. Lauer, M. Riedmiller, M. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In Proceedings of International Conference on Machine Learning, ICML '00, pp. 535-542, Stanford, CA, 2000.
10. Machado, A., Ramalho, G., Zucker, J-D., Drogoul, A. Multi-Agent Patrolling: an Empirical Analysis of Alternative Architectures. In: 3rd. International Workshop on Multiagent Based Simulation (MABS'02), Bologna, 2002.
11. Menezes T., Tedesco P., Ramalho G. Negociação em sistemas Multi-Agente para Patrulhamento. Technical Report from Universidade Federal de Pernambuco, Brasil, (2004).
12. Riedmiller, M. Merke, A. Using Machine Learning Techniques in Complex Multi-Agent Domains. In Perspectives on Adaptivity and Learning (2002), LNCS, Springer.
13. RoboCup Rescue home page: http://www.r.cs.kobe-u.ac.jp/robocup-rescue/, accessed in 2002.
14. Russell, S. J., Norvig, P. Artificial Intelligence: A Modern Approach. Prentice Hall (1995) 61, 31-52.
15. Sempé, F., Drogoul, A. Adaptive Patrol for a Group of Robots. International Conference on Intelligent Robots and Systems, 2003.
16. Stone, P. Layered Learning in Multi-Agent Systems. PhD thesis, Computer Science Department, School of Computer Science, Carnegie Mellon University, December 1998
17. Sutton, R., Barto, A. (1998) Reinforcement Learning: An Introduction. Cambridge, MA.
18. Sutton, R., Precup, D., & Singh, S. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Articial Intelligence, 112, 181-211.
19. Tumer, K., Agogino, A., Wolpert, D. Learning sequences of actions in collectives of autonomous agents. In Autonomous Agents & Multiagent Systems, pages 378–385, part 1. ACM press, 2002.

AAMAS'04, July 19-23, 2004, New York, New York, USA.