COOPERATIVE MULTI-ROBOT SYSTEM FOR INFRASTRUCTURE SECURITY TASKS

Erik Hernandez, Antonio Barrientos, Claudio Rossi and Jaime del Cerro
Center for Robotics and Automation, Technical University of Madrid, C/ José Gutierrez Abascal 2, Madrid, Spain

Keywords: Reinforcement learning, Robotics security, Infrastructure security tasks, Multi-robot systems.

Abstract: As a result of terrorist attacks in the last years, new efforts have raised trying to solve challenges related to security task automation using robotic platforms. In this paper we present the results of a cooperative multi-robot approach for infrastructure security applications at critical facilities. We formulate our problem using a Ms. Pac-Mac like environment. In this implementation, multiple robotic agents define policies with the objective to increase the number of explored states in a grid world. This is through the application of the off-policy learning algorithm from reinforcement learning area, known as *Q-learning*. We validate experimentally our approach with a group of agents learning a patrol task and we present results obtained in simulated environments.

1 INTRODUCTION

Recently, terrorist attacks around the world have showed environments vulnerability. As a result of these events new efforts have arisen in research to try to solve challenges related with security tasks automation. *Robotics Security* is a consequence of this research interest in security systems using mobile robots. Actually, "intelligent" buildings security depend upon Closed Circuit TeleVision for information gathering to identify anomalous behaviors. However, these solutions are inflexible and human limitations such as boredom, distraction or fatigue affect their performance, moreover, in some environments people must deal with dangerous conditions. Therefore, it is important to improve security elements utilized in these systems. Mobile robots characteristics make it suitable for this purpose. There are numerous advantages using mobile robots, i.e., it does not experiment human limitations. However, some tasks are too complex that a single robot cannot achieve good results. To overcome these challenges we can use *Multi-Robot Systems* which are defined as a set of homogeneous or heterogeneous robots interacting in the same environment using cooperative behaviors (Everett, 2003), (Oates et al., 2009), (Luo and Lin, 2009), (Roman-Ballesteros, 2006).

In this paper we present a collaborative multi-robot approach for infrastructure security applications at critical facilities such as nuclear power plants, urban transport installations and so forth. We have formulated our problem using a Ms. Pac-Mac like environment in which multiple agents define policies to explore states in a grid world, this adaptation differs to other works trying to solve multi-robot patrolling tasks (Chevaleyre, 2004), (Elmaliach et al., 2007). To solve this problem we have implemented an algorithm from reinforcement learning known as *Q-learning*. Reinforcement Learning yields machine learning systems which appeals to many researchers due to its generality. In these systems an agent learns how to achieve a goal by trial-and-error interactions with its environment. Our approach is different to numerous implementations on grid worlds of *Q-learning* works, mainly in the objective and in the way in this objective is reached. The rest of this paper is organized as follows. Section 2 introduces the problem. Section 3 shows the methodology implemented to solve the learning task. Section 4 gives the experimental result and their evaluation. Finally, section 5 concludes the paper.

2 PROBLEM DESCRIPTION

We have adapted Ms. Pac-Man game to represent infrastructure environments. Namely, in this game a human steers Ms. Pac-Man throughout a maze to eat dots while avoid getting caught by four ghosts. Despite our approach is based on this game, we have removed much of its complexity. First, we shape the

Hernandez E., Barrientos A., Rossi C. and del Cerro J..
COOPERATIVE MULTI-ROBOT SYSTEM FOR INFRASTRUCTURE SECURITY TASKS.
DOI: 10.5220/0003721403130316
In *Proceedings of the 4th International Conference on Agents and Artificial Intelligence* (ICAART-2012), pages 313-316
ISBN: 978-989-8425-96-6

maze structure into an infrastructure environment as a grid world with a set of paths where each cell represents a state. Characteristics such as corridors, T-junctions, interjections and L-turns remain. At each step time an agent perceives a state and selects one of four actions: North, West, South or East. That brings it to the next state. According to the orientation in which each agent reaches a state it has four possible actions to select. Second, we consider the dots distributed around the environment. The agents must eat all dots with fixed and variable values to begins a new episode. Finally, multiple agents that represent the ghosts are the last adaptation of the game.

The tasks that each agent can perform are: patrol, homing and avoiding. *Patrol* causes the agent to move throughout the environment following a path by taking decision to choose actions. *Homing* causes the agent to be resting to charge batteries energy. Finally, *Avoiding* causes the agent to avoid obstacles or other robots. Each agent switches its internal task between *Patrol* and *Homing* considering battery life value. Therefore, the learning task consist of finding a mapping from states to actions for cooperative patrol where each agent learns how to select the action with the highest value. Each agent implements a set of adaptation rules. Individually each agent has a map, whereas from a group perspective, the rules are triggered by pheromone like communication among robots, so that each time step they inform others the state explored (Khamis et al., 2006).

3 REINFORCEMENT LEARNING

A reinforcement learning model consist of a discrete set of environment states, S; a discrete set of actions, A; and a set of scalar reinforcement signal. In this model, an agent learns a mapping from situations to actions by trial-and-error interactions with the environment to achieve a goal. This environment must be at least partially observable. At each time step $t \in T$ each agent receives a current state indication $s_t \in S$ of the environment, then it chooses an action $a_t \in A$ to generate an output which changes the state of the environment to $s_{t+1} \in S$ and the value of this state transition is indicated to the agent through an scalar r_t known as *reward* (Sutton and Barto, 1998). A *reward* defines the goal in a reinforcement learning problem. It maps each perceived state or state-action pair to a single numerical value that indicates the intrinsic desirability of that state. An important *reward* property is known as *Markov Property*. A *reward* with this property must include immediate sensation and retain all relevant information from the past (Puterman, 1994). Thus, the agent learns to perform actions that maximize the sum of the rewards received when starting from some initial state and proceeding to a terminal one. The reward function must be necessarily unalterable by the agent and it only serves as a basic for altering a *policy* π. In this implementation mainly dots values have been used as reward. A *policy* is a mapping from each state, $s \in S$, and action, $a \in A(s)$, to the probability $\pi(s,a)$ of taking action a when in state s. An *stationary policy*, $\pi : S \rightarrow \Pi(A)$ defines a probability distribution over actions. A *policy* is the core of the agent since it defines which action must be performed at each state. Thus, the objective of reinforcement learning is to develop a agent with a behavior policy to choose actions that tend to increase the long-run sum of values of the *reward*.

We have implemented an off-policy temporal difference algorithm known as *Q-learning* which learns directly from raw experience without a model of the environment and updates estimations based in part on other learned estimations without waiting for a final outcome. A model consist of the state transition probability function $T(s,a,s')$ and the reinforcement function $R(s,a)$. However, reinforcement learning is concerned with how to obtain an optimal policy when such a model is not know in advanced (Watkins and Dayan, 1992). The objective of *Q-learning* is to learn the action-value function Q applying the rule $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r + \gamma \max_a Q(s_{t+1},a) - Q(s_t,a_t)]$, where $< s_t, a_t, r, s_{t+1} >$ is an experience tuple. If each action is executed in each state an infinite number of times on an infinite time run and α is decayed appropriately, the *Qvalues* will converge with probability 1 to their optimal values Q^* (Kaelbling et al., 1996). An action-value function for policy π defines the value of taking action a in state s under policy π, denoted by $Q^\pi(s,a)$, as the expected reward starting from s, taking action a,and thereafter following policy π.

The general form of the *Q-learning* algorithm consist of nine step, as describe below.

```
1 Initialize Q(s,a) arbitrarily
2  Repeat (for each episode):
3   Initialize s
4   Repeat (for each step of episode):
5    Choose a from s using policy derived from Q
6    Take action a, observe r, s'
7    Apply equation X
8    s <- s'
9   Until s is terminal
```

In this implementation an episode terminates when all dots are eaten and each step of episode consist of choosing an action a when in state s, updates $Q(s,a)$ and go to s'.

4 EXPERIMENTS AND RESULT

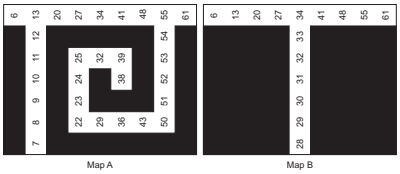

Figure 1: Maps Utilized in Our Implementation.

We have used the maps depicted in figure 1 to describe important aspects of our approach.

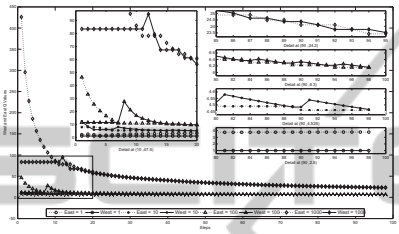

Figure 2: Evolution of West and East *Qvalues* in State 34 T-junction of Map B with four Different Initial Values.

To initialize the *QValues* we select values so that north is chosen the first time a state is explored, then west or east if north has been selected and south as the last option. Since the algorithm performance changes with the multiplier of initial values, we have executed experiments with four multipliers $1x$, $10x$, $100x$ and $1000x$. Figure 2 shows the first 100 steps of the algorithm in state 34 of map B. As it can be seen in the details, the multiplier affects the frequency of the selection of west or east actions. The bigger the frequency, the better the performance of the algorithm. Nevertheless, when a different action is selected every step, the performance decreases. Based on this results we have selected the multiplier $100x$.

To calculate the value of γ we have executed experiments of one episode with values from 0.1 to 0.9 and increments of 0.1 using map A. The results show that the average of $\gamma = 0.9$ gets the shortest path, however, it performs a high exploration in the first episode, in contrast with $\gamma = 0.1$ in which explorations take a number of steps similar to the shortest path. Based on these results, we have selected $\gamma = 0.1$ which indicates that the agents are concerned with maximizing immediate rewards.

Taking this indication into account, figures 3 and 4 show the evolution of the reward, $\gamma \max_a Q(s_{t+1}, a)$ and *Qvalues* for west and east options in states 13 and 34 of map A and B, respectively. Both states are T-junctions, however, when each state is reached with east orientation, the number of next states for each one is different, i.e., 1 in west direction and 22 in east

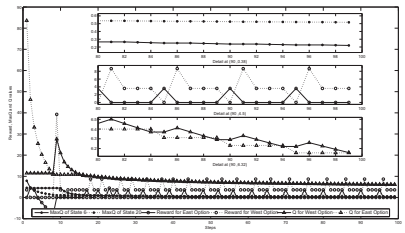

Figure 3: First 100 Steps of the Evolution of r, $\gamma \max_a Q(s_{t+1}, a)$ and *Qvalues* in State 13 T-junction of Map A.

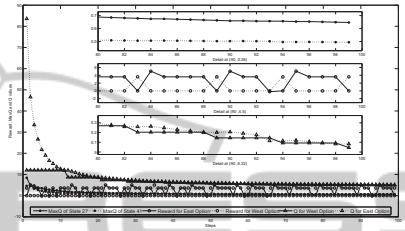

Figure 4: First 100 Steps of the Evolution of r, $\gamma \max_a Q(s_{t+1}, a)$ and *Qvalues* in State 34 T-junction of Map B.

direction for state 13 of map A and 4 in both directions for state 34 of map B. As can be seen in the details of $\gamma \max_a Q(s_{t+1}, a)$, in figures 3 and 4, despite that the number of next states of each state is not the same, the difference does not affect proportionally the values of $\gamma \max_a Q(s_{t+1}, a)$. This is a consequence of γ value, the only appreciable difference is the reward that every 5 steps the agent receives when it explores state 13 in these conditions.

The value of the shaped reinforcement signal r used in our approach is the result of the expression $r = dot(s') + incrementalDot(s') + \lambda$, where λ is the relation between the number of eaten dots and the number of explored states. Finally, the value of the learning rate α that we have used in all experiments in this work is subject to the rule, $\alpha(Q(s,a)) = \frac{1}{N(s,a)}$, where $N(s,a)$ is the number of times that the action a has been selected when in state s.

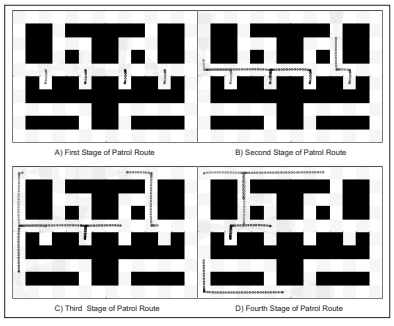

Figure 5: A Team of Four Khepera III Robots Patrolling.

Once we get a suitable performance with an agent, we use a team of four agents with different initial position. Now each one must communicate to indicate its current state and the next one, as a result of this cooperation, each agent defines a policy that allows it to patrol in different areas of the environment and design a special division of labor in which negotiation is not needed.

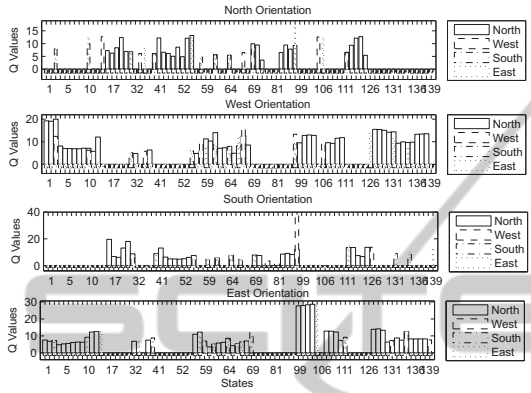

Figure 6: Behavior Policy of Agent A_0.

Figure 6 shows the policy of one agent. As it can be seen, almost all states are explored at least once, however, after 2000 episodes, an agent defines a behavior with preference to patrol on the left upper area of map. Due to the lack of space the policies of the other agents are not showed, however the simulation depicted in figure 5 shows the four agents patrolling in the same environment. Even though the results we have presented consider four agents, our approach is not restricted to this number.

5 CONCLUSIONS

In this paper, we presented a cooperative multi-robot approach for security applications at critical facilities in which a team of agents define policies to patrol, using the algorithm of reinforcement learning known as *Q-leaning*. The initial *Qvalues* that we selected, allow each agent to move in a suitable way instead of in a random manner, as some works of multi-robot implementations in multi-robot foraging or search and rescue literature have proposed. Therefore, the agents do not need to wait until a learning phase terminates to patrol. It is worth noting that in *Q-learning* literature, almost all implementations in grid worlds describe solutions as undiscounted task in which the goal positions are defined and agents receive delayed reward. In contrast, in our approach we do not define goal positions but task goal. Indeed, as it can be seen in section 4 a value near undiscounted factor does not work or its performance is worse than the smallest value. Moreover, these implementations seem to be off-line process instead of on-line as we claim in our implementation. It is on-line due to the fact that an agent can explore the environment meanwhile it finds its optimal behavior policy. Finally, we validated our approach in a simulation environment to show the ability of the method to define patrol paths.

ACKNOWLEDGEMENTS

This work has been supported by the Robotics and Cybernetics Research Group at Technique University of Madrid (Spain), and funded under the projects "ROTOS: Multi-Robot system for outdoor infrastructures protection", sponsored by Spain Ministry of Education and Science (DPI2010-17998), and the project ROBOCITY 2030 Project, sponsored by the Community of Madrid (S-0505/DPI/000235).

REFERENCES

Chevaleyre, Y. (2004). Theoretical analysis of the multi-agent patrolling problem. In *Intelligent Agent Technology, 2004.(IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on*, pages 302–308. IEEE.

Elmaliach, Y., Agmon, N., and Kaminka, G. (2007). Multi-robot area patrol under frequency constraints. *Annals of Mathematics and Artificial Intelligence*.

Everett, H. (2003). Robotic security systems. *Instrumentation & Measurement Magazine, IEEE*.

Kaelbling, L., Littman, M., and Moore, A. (1996). Reinforcement learning: A survey. *Arxiv preprint cs/9605103*.

Khamis, A., Kamel, M., and Salichs, M. (2006). Cooperation: concepts and general typology. In *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, volume 2, pages 1499–1505. IEEE.

Luo, R. and Lin, T. (2009). Multisensor based security robot system for intelligent building. *Robotics and Autonomous Systems*.

Oates, R., Milford, M., and Wyeth, G. (2009). The implementation of a novel, bio-inspired, robotic security system. *, 2009. ICRA'09.*

Puterman, M. (1994). *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Roman-Ballesteros, I. (2006). A framework for cooperative multi-robot surveillance tasks. *Robotics and Electronics*.

Sutton, R. and Barto, A. (1998). *Reinforcement learning: An introduction*, volume 116. Cambridge Univ Press.

Watkins, C. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3):279–292.