

Application of reinforcement learning to a mobile robot in reaching recharging station operation

L. Cragg and H. Hu

*Department of Computer Science, University of Essex, Colchester CO4 3SQ, U.K.
Email: lmcragg@essex.ac.uk, hhu@essex.ac.uk*

Abstract

Efficient control strategies for robot systems cannot always be developed by hand, especially when the robot system is operating in an unknown or uncertain environment. In this paper we show how Reinforcement Learning (RL) might be applied to improve the efficiency of a mobile robot in nuclear decommissioning characterisation, in particular allowing it to learn efficient routes back to a recharging station. We implement this learning functionality in a mobile agent (MA) environment. By doing so we can make use of the positive characteristics of MA mobility such as adaptability, fault tolerance and dynamic positioning of learning or control in a distributed system to supplement learning. Experimental results show how RL provides a more efficient method in this task than a non-AI control approach.

1. Introduction

If the task for a mobile robot is to navigate in an unknown or uncertain environment, it is difficult for a programmer to consider every eventuality when defining its behaviour. It is therefore beneficial to develop a mobile robot which can adapt to changes in the environment. One way to make a mobile robot more adaptable is to allow it to learn from experience. The Artificial Intelligence (AI) technique of RL allows a robot to learn from experience through interaction with an *environment*. The robot establishes the utility of taking an *action* from a range of actions when it finds itself in one of a range of environmental *states*. It learns which actions will lead over time to it realising optimum reward in pursuit of its goal.

In complex tasks such as search, rescue, space exploration or investigation in hazardous environments, mobile robots should be able to explore unknown environments and make decisions

when encountering unexpected situations. In this paper we examine the use of RL to aid a mobile robot in the nuclear decommissioning task of characterisation (i.e. the exploration of unknown or uncertain hazardous environments to assess and map levels of radiation so that safe and cost effective decontamination can subsequently be applied).

Many existing robot nuclear characterisation systems are tethered, single robot systems [1], [2] and [3]. To speed up mapping and increase system fault tolerance it might be more appropriate to use multiple robots in which wireless network technologies rather than tethers are employed (despite interference or noise caused by radiation) as demonstrated in [4] and [5]. While a wireless system may allow increased system flexibility and reduce inter-robot space conflict issues, a wireless robot is constrained by the need to periodically return to a recharging station to recharge its battery. We employ RL to enable a mobile robot to learn efficient paths back to a recharging station through an environment.

The rest of this paper is structured as follows: Section 2 briefly outlines some background material. Section 3 describes how RL was implemented in a MA environment. Section 4 describes experiments which examine the rate at which a RL implementation can learn a stable policy, and compare RL against non-AI control in a number of environments. Section 5 analyses the results of our experiments and Section 6 provides some conclusions.

2. Background/literature review

As previously mentioned, RL is a control strategy in which an agent embedded in an environment attempts to maximise total reward (or return) in pursuit of a goal. The agent at any time step is in a particular state relative to the environment and can take one of a number of actions within that state to reach its goal. When the agent performs an action it receives feedback in the form of a reward from the environment which indicates if this is a good or bad action to take in attempting to achieve the goal.

The value of taking an action or being in any state can be defined using value functions e.g. the *Action-Value Function (or Q-Value)* $Q^\pi(s,a)$, the expected return when starting from state s , taking action a , and then following policy π . The return can be defined as the sum of discounted rewards:

$$R_t = \sum_{k=0}^T \gamma^k r_{t+k+1}$$

where t is the current time step, γ is the discount rate (a value between 0 and 1), r is reward and T the total number of rewards (in which $T = \infty$ or $\gamma = 1$ but not both). The policy is a method by which an action is selected.

In this paper an ϵ -greedy policy is employed. In ϵ -greedy action selection, the action with highest estimated reward is chosen most of the time (greediest action). However occasionally with a small probability, ϵ , an action is selected at random. The aim of RL methods is to learn $Q^\pi(s,a)$ values so that the optimum action can be taken in any state. In this research we employ Q-learning, a temporal difference learning method which can learn from raw experience without a model of an environment's dynamics, and which updates estimates based in part on other learned estimates without waiting for a final outcome. It can be employed using simple algorithms which are not computationally expensive.

Fig. 1 presents the Q-learning algorithm: α is the learning rate (a value between 0 and 1), which determines the rate at which Q-values are updated. γ is the discount rate, which causes future rewards to be worth less than immediate rewards, and \max_a is the maximum reward attainable in the state following the current one.

```

initialise  $Q^\pi(s,a)$  arbitrarily
repeat (for each episode):
  initialise  $s$ 
  repeat (for each step of episode):
    choose  $a$  from  $s$  using policy derived from  $Q$ 
    take action  $a$ , observe  $r, s'$ 
     $Q^\pi(s,a) \leftarrow Q^\pi(s,a) + \alpha [r + \gamma \max_{a'} Q^\pi(s,a') - Q^\pi(s,a)]$ 
     $s \leftarrow s'$ 
  until  $s$  is terminal

```

Fig. 1 Q-learning algorithm

Q-learning is proven to converge to an optimal action given that all state and action pairs are updated. By using an ϵ -greedy action selection mechanism all state and action pairs will be updated and the values of $Q^\pi(s,a)$ should converge given enough learning episodes. Q-learning and RL are described in more detail in [6].

3. System implementation

This section describes how RL was implemented in our system, as part of wider research into the use of MA mobility within multiple robot systems for nuclear decommissioning characterisation [7].

Our overall system is a MA based multi-robot environment [8]. A MA is a software entity which exists in an execution environment within which it has a clearly defined boundary. It can control to a large extent its own execution and interaction with other agents or computing components i.e. it is not purely reactive and it may be autonomous, asynchronous, dynamic and intelligent. A MA differs from a static software agent in the sense that it is mobile within its execution environment. When, as in this case, an agent's execution environment is composed of a number of distributed computers or robots, this allows it to move between computing nodes in order to perform tasks.

MAs have software engineering advantages over an amalgamation of other distributed computing architectures, can make our system more adaptable and robust [9], and allow us to dynamically position control or learning at the most appropriate location within a distributed environment [10]. By

encapsulating learning in a MA we can apply these characteristics to this functionality, independent of the learning ability.

Our MA's main body is a live() function in which its main activity is located. We implement learning in this part of the agent. A flow chart showing its behaviour is shown in Fig. 2. As can be seen, the RL MA can operate autonomously to learn robot behaviours or be instructed via a user control GUI. A learnt policy can then be applied either to a simulated robot or a real Pioneer robot [11] in our lab, as shown in Fig. 3.

Once the RL MA is initialised it is provided with data about the environment in which the robot is located i.e. position of obstacles, the recharging station and the robot. A model of the environment is constructed based upon this information, which also describes the current state of the environment, potential moves the robot can take, and the reward which would be derived from taking a particular move or getting to a goal state (i.e. the recharger). A multi-dimensional array is initialised to store Q-values for all actions in all states along with the Q-learning algorithm.

The agent can enter an autonomous control mode in which Q-learning algorithm parameters are set, and the algorithm is applied to the model of the environment to simulate a number of learning episodes and in so doing update the values in the Q-value array. Alternatively these processes can be controlled by the user control GUI. The robot environment is an idealised grid based environment containing a number of obstacles in which each grid square represents a state. The robot can take one of eight actions in each state allowing it to move between states i.e. (North (N), South (S), East (E), West (W), NE, NW, SE, SW).

The mobile robot can learn to move through its environment towards a recharging station located in the corner of the environment. Moves outside the environment or through an obstacle are considered illegal and not permitted; the aim of the robot is to find the shortest legal path back to the recharging station. Using this system we conducted a number of experiments to test the ability of the system to learn in a number of environments, these are described in the next section.

4. Experiments

This section examines experiments, which we conducted with the RL system described previously. Two sets of experiments were conducted:

- (1) *RL rate* – to assess the rate at which a stable policy could be learnt using RL in a variety of robot environments.
- (2) *RL vs. non-AI control* – to compare RL against a non-AI control method.

A more detailed examination of set-ups employed for these two types of experiment is discussed in the following sections.

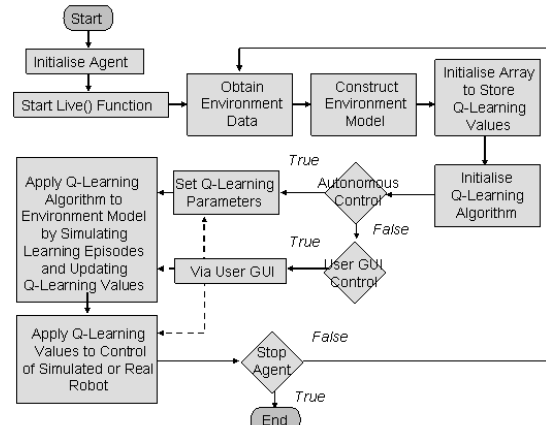


Fig. 2 Flow chart of a RL MA

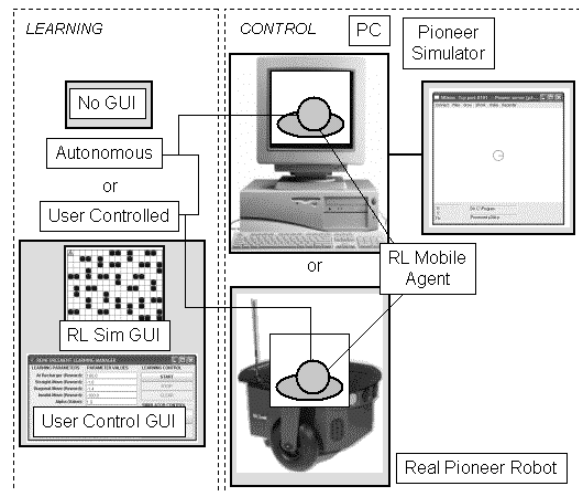


Fig. 3 RL based MA/mobile robot system

4.1. Reinforcement Learning Rate

A number of environments were selected to test the RL method as shown in Fig. 4 in which the triangle at the top left corner is the recharging station, a column is an obstacle, and the icon at the bottom right corner is the robot.

The aim of the RL rate experiments is to assess the rate at which a stable policy can be learnt using RL in a variety of robot environments. In these experiments the aim of RL is to guide the robot to

the recharging station using the shortest route. In order to direct the RL learning process the following rewards were used: +100 if the robot reaches the battery recharging station, -1.0 if the robot makes a straight move (N, S, E, W), and -1.4 if the robot makes a diagonal move (NE, NW, SE, SW). These rewards encourage the robot to move to the recharging station in the least number of valid moves to achieve a maximum reward.

In the Q-learning algorithm α was set to 0.1 to encourage learning to occur slowly, γ was set to 0.9 so that future reward has more value than immediate reward, and ϵ in the ϵ -greedy algorithm was set to 0.001 to encourage exploration on average only once every 1000 moves, so that the robot utilises maximum reward actions more often.

The experiment set-up for the RL Rate experiments is shown in Fig. 5. In this set-up a RL MA is located on a PC on which it runs a number of simulations of the robot environments. By running a large number of simulations in which the robot starts in a variety of different locations, the RL MA can learn the optimum path the robot should follow given any start point in the environment. 50000 episodes (an episode being a successful run of the robot reaching the recharging station) were conducted for each of the five environments shown in Fig. 4.

The average positive and negative rewards obtained over these episodes were recorded; these would show how often the robot successfully reached the recharging station and obtained (+100 reward) relative to the number of moves made to reach the recharging station (-1 or -1.4 reward).

It was found that after about 40000 episodes a stable average reward for each environment was obtained. The results from these experiments are shown in Section 5.1.

4.2. RL vs. non-AI control

RL vs. non-AI control experiments were conducted in the same experimental environments used in the RL rate experiments. The aim of the RL vs. non-AI control experiments is to compare RL against a non-AI control method. As previously described, using RL the aim is to control the robot to move back to the recharging station using the shortest path possible. In order to direct the RL learning process, the same values for rewards, α , γ and ϵ were used as in the RL rate experiments.

Non-AI control in these experiments was implemented using a simple reactive control approach. Under the reactive control approach the

robot attempts to find the shortest path back to the recharging station. It knows the position of the recharging station relative to its current location and attempts to take moves towards it. If it encounters an obstacle in its path it takes a move in a random direction before attempting once more to move towards the recharging station.

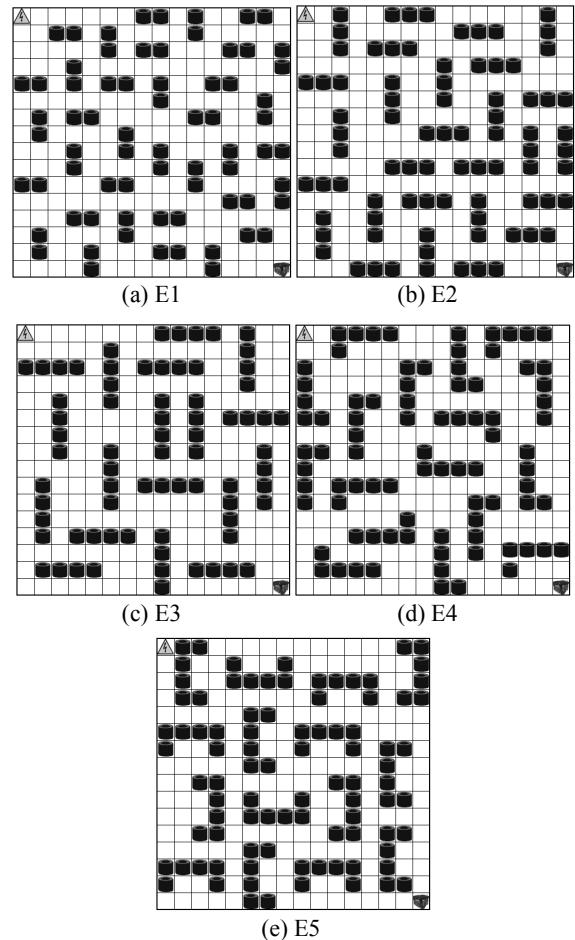


Fig. 4 Five Environment settings used in experiments

The experimental set-up for these experiments is the same as for the RL rate experiments as shown in Fig. 5. In addition to the RL MA a non-AI MA is also located on the PC and runs a number of simulations. By running a large number of simulations in which the robot starts in a variety of different starting locations the RL and non-AI MAs can be used to assess the overall effectiveness of both control methods to control a robot back to the recharging station.

The average positive and negative reward obtained by the RL and non-AI MAs in controlling the robot over 10 trials (1 trial = 10000 moves) was

obtained to assess the relative percentage of positive to negative rewards obtained. As the non-AI method does not employ rewards, the type of move made and number of goals reached were recorded in order to calculate what the reward would be if measured with the RL implementation. 10000 moves for a trial acted as a timeout in the event that non-AI control failed to find a path to the charging station for the robot, and was also the figure at which the average positive reward obtained from both RL and non-AI control became stable. The results for these experiments are shown in Section 5.2.

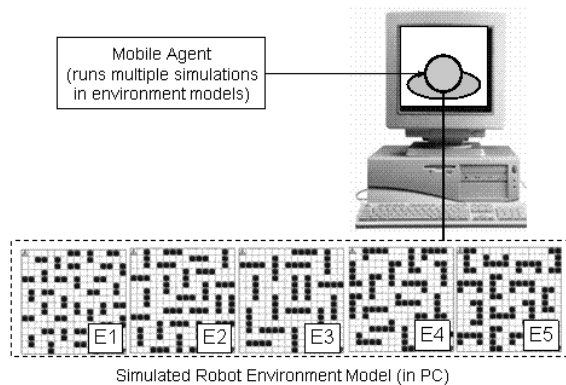


Fig. 5 Experiment set-up

5. Results and analysis

5.1 RL rate

RL rate experiment results are shown in Fig. 6, which include results for all environments from 0-50000 episodes. It shows that using this RL setup and environments, a stable policy can be learnt by approximately the 40000 episode mark, and that the most rapid learning occurs in the first 1000 episodes as shown by a rise in positive reward as an overall percentage of reward from 0% to 49-50%.

Learning continues to rise, but at a slower rate over the next 9000 episodes up to the 10000 episode mark from 49-50% to 79-82% for positive reward. Finally, over the final 40000 episodes up to the 50000 episode mark the learning rate continues at a slower rate and becomes stable at an 85-87% level. From Fig. 6 it can be seen that there is not a large difference between the learning rates or overall positive reward obtained in each of the five environments despite the difference in environment structure.

5.2. RL vs. non-AI control

The results for the RL vs. non-AI control

experiments can be seen in Fig. 7. As can be seen, RL control outperforms non-AI control in each of the five environments. There is little difference in the overall performance of RL control in each of the five environments and it maintains a higher degree of positive reward relative to non-AI control.

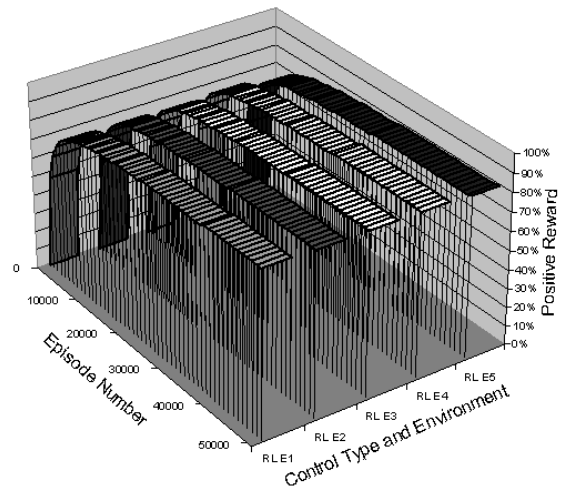


Fig. 6 RL rate experiment results 0-50000 episodes

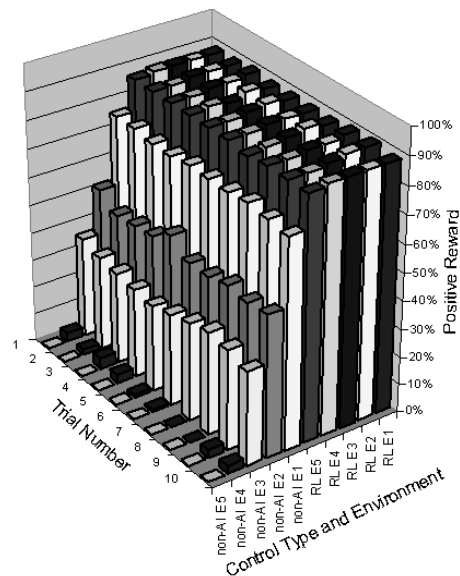


Fig. 7 RL vs. non-AI control experiment results

In some of the environments the non-AI control performance is very low, or fails completely because the robot cannot reach the recharging station because it becomes stuck in local minima and cannot escape e.g. non-AI E5 and non-AI E4.

While non-AI control can produce results that allow the robot to reach the recharging station

successfully in some environments, RL control learns a path which is on average shorter. In other words, RL control provides a more efficient as well as more reliable means for directing the mobile robot back to the recharging station. Fig. 8 shows some paths taken using RL and non-AI control from seven random starting positions in environment E3. It demonstrates that when using non-AI control the robot always attempts to head directly towards the recharging station even if this means attempting to move through obstacles. This results in a larger average number of moves, and in some situations an inability to move round an obstacle. This is because it cannot learn a new strategy.

In contrast the paths for RL control show that RL can learn to find a path which navigates the robot around obstacles in order to reach the recharging station rather than attempting to move it through obstacles.

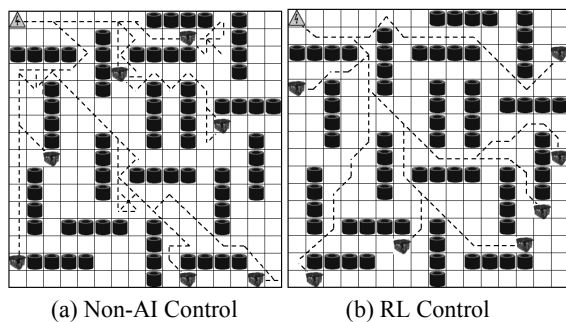


Fig. 8 Comparison of control from random start paths

6. Conclusions

In this paper, we have examined the use of the RL technique to provide a mobile robot with the capability to operate autonomously in hazardous environments in general, autonomously charging its batteries in particular. We have implemented the robot control architecture in a mobile agent environment, which provides software engineering advantages in comparison to an amalgamation of other distributed computing architectures. The proposed system is more adaptable and robust and allows a mobile robot to dynamically position control or learning to reach the most appropriate location within the environment.

Experimental results show the learning characteristics of RL control in five different simulated environments. It has been shown that RL can be used for a mobile robot to learn efficient control policies in a range of environments of

varying complexity.

A comparison of the RL approach against a non-AI control method shows that RL provides a more efficient and safer method for a mobile robot to return to a recharging station. The RL method can be used to successfully guide the mobile robot back to the recharging station even in environments where the reactive method would have failed to find a path. This justifies the implementation of the AI technique in our wider multiple robot architecture.

References

- [1]. US Department of Energy, Houdini II Remotely Operated Vehicle System – Innovative Technology Summary Report DOE/EM-0495, 2002.
- [2]. Pacific Northwest National Laboratory, Technology Deployment Fact Sheet - Andros Robot for Canyon Disposition Initiative Remote Characterization, 2002.
- [3]. Willis, W.D., Anderson, M.O., McKay, M.D., The Remote Underwater Characterization System, Proc. of the American Nuclear Societies 8th Topical Meeting on Robotics and Remote Systems, 1999.
- [4]. Anderson M. et al., Demonstration of the Robotic Gamma Locating and Isotopic Identification Device, Proc. of the American Nuclear Soc. Spectrum, 2002.
- [5]. US Department of Energy, Mobile Automated Characterisation System – Innovative Technology Summary Report DOE/EM-0413, 2002.
- [6]. Sutton R. and Barto A.G., Reinforcement Learning: An Introduction, MIT Press, 1998.
- [7]. Cragg L. and Hu H., Application of MAs to Robust Tele-operation of Internet Robots in Nuclear Decommissioning, Proceedings of IEEE International Conference on Industrial Technology–ICIT’03, pages 1214-1219, Maribor, Slovenia, 10-12 Dec. 2003.
- [8]. Cragg L. and Hu H., Application of Mobile Agents in Multiple Online Robots, Proceedings of e-ENGDET 2004 (the 4th Int. Conf. on e-ENGINEERING & Digital Enterprise Tech.), Leeds, England, 1-3 Sept. 2004.
- [9]. Cragg L. and Hu H., Implementing ALLIANCE in Networked Robots using Mobile Agents, Proc. of IAV 2004 5th IFAC Symposium on Intelligent Autonomous Vehicles, Instituto Superior Tecnico, Lisbon, Portugal, 5th-7th July, 2004.
- [10]. Cragg L. and Hu H., Implementing Multiple Robot Architectures using Mobile Agents, Proc. of Control 2004, 6th-9th September, Bath, England, 2004.
- [11]. ActivMedia Robotics, Software, Documentation and Tech. Support, <http://robots.activmedia.com/>, 2005.