Simulation of Multi-robot Reinforcement Learning for Box-pushing Problem

Krešimir Kovač, Ivan Živković and Bojana Dalbelo Bašić
University of Zagreb/Faculty of Electrical Engineering and Computing
Department of Electronics, Microelectronics, Computer and Intelligent Systems, Zagreb, Croatia
{kresimir.kovac|ivan.zivkovic|bojana.dalbelo}@fer.hr

***Abstract*—The box-pushing problem represents a challenging domain for the study of object manipulation in a multi-robot environment. Our box-pushing problem is based on the *pusher-watcher* approach, involving two *pushers* robots that learn the best strategy for cooperatively moving an oversized elongated box to a specified goal and one *watcher* robot acting as the environment. This paper presents a solution to the box-pushing problem based on reinforcement learning in a multi-agent system. Within the framework of the paper, a simulator has been developed to carry out practical tests.**

I. Introduction

The box-pushing problem has its practical applications and also a theoretical significance. From a practical point of view, box pushing is prototypical problem for studying various tasks, requiring cooperation of a number of smaller robots moving larger objects that are widely applied in industry and also in space research. From a theoretical point of view it is variant on the canonical object manipulation. The task that we have given to our robots includes problems of orientation in space, cooperation on a common task, learning in a multi-agent environment [1]-[3] and other. We shall present our solution to the box-pushing problem that is based on reinforcement learning (RL), as well as a simulator developed to visualize and evaluate solutions. We believe that our solution enables robots to successfully solve the task and allows for further improvements and extensions.

The paper is organized as follows. Section II presents the theoretical base for the box-pushing problem. We review a number of preceding papers and solutions to this task. Section III describes experimental environment and all parameters of practical tests that were carried out. The proposed solution is presented in Section IV and discussed in Section V. The conclusion is given in Section VI.

II. Box-pushing Problem

The most general definition of the *box-pushing* problem is as follows: a number of agents (in this case, robots) have to manipulate a passive object (usually a box) to a predefined target position.

The proportions between the robots and the size of the box are such that it would be difficult, if not impossible, for a single robot to manipulate the box to the target position. In order to execute the task successfully, the robots are, therefore, forced to cooperate. There is a number of variants of this basic problem. They differ in the number and characteristics of robots, movability of the target etc. Thus, for instance, in [4] and [5] two six-legged Genghis-II robots, equipped with two frontal whiskers for detecting contact and pyroelectric sensors for detecting from moving IR sources, were engaged. The light source was set as the target and was fixed, although it can also be moved during the test. Robots pushed and turned the box towards the light, while maintaining a perpendicular contact with the box by means of whiskers. In the second example, in [6], a group of Pioneer 2-DX robots, equipped with a series of sonars and a colour camera, were engaged. One robot acted as the *watcher*, coordinating two other robots that were acting as *pushers* that were actually pushing the box towards the goal. Box-pushing problems involving more than one box and a single robot are also studied, like in [9]. Since we are primarily interested in the multi-agent aspect of the problem, we shall concentrate on the initial variants of the problem.

The box-pushing problem is suitable for a number or reasons. It is a classic example of a multi-agent system with simple cooperation and is easy for studying. Also, the basic idea is sufficiently simple and the aspects that are of interest to us can be clearly perceived, without spending too much attention on uninteresting points. Due to its simplicity, the box-pushing problem may be projected to a variety of other applications.

Since problems of multi-agent systems and cooperation and communication between agents, involving a small number of agents, may be well presented on the box-pushing task, the example is also appropriate for realization in the laboratory.

III. Experimental Environment

The proportions between the robots and the size of the box are such that they fulfil the definition of the *box-pushing* problem.

The test set-up has been limited to a square-plan room (5.6m by 5.6m), in which there is a box (2.7m long and 0.5m wide) and three robots (circular in plan, 15 cm radii). One robot acts as the *watcher*, while two remaining robots act as *pushers*.

0-7803-8271-4/04/$20.00 ©2004 IEEE

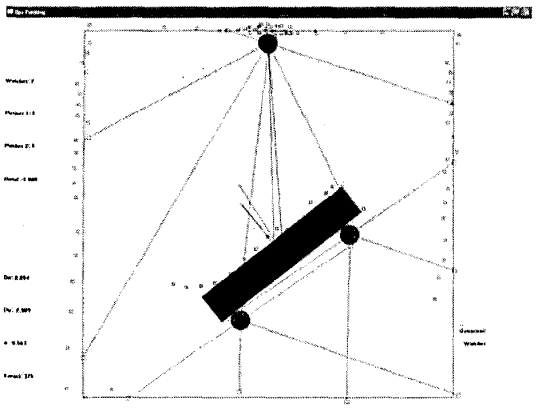

Figure 1. Plan of the simulator

We quote dimensions with measuring units, although the proportions (e.g. proportion between the size of the robots and the box) are very important, rather than the actual sizes. Actual dimensions help to visualize the test, particularly its realization in the physical world with actual robots.

Our solution is based on the AmigoBot type of robot, equipped with a series of eight sonars each, calibrated in the test to a ±10 cm tolerance. Thus, the noise of actual sonars has been simulated. It appears that the results do not practically depend upon the tolerance of the sonars if it is relatively small. The tests with and without sonar errors yielded approximately equal results.

At the beginning of each test, the three robots and the box are placed at their initial positions. The *watcher* robot is always positioned in the centre of the north wall, almost touching it. The box is placed at random in the south half of the room. The front side of the box is defined by the axis vector, which is perpendicular on the longer side of the box, and initially facing the *watcher*. The axis vector is rotated by a random angle between -90° and 90°. Thus, the front side of the box can be parallel to the north wall, parallel to either the east or the west wall, or it can be rotated between these two positions. *Pusher* robots are placed in the pushing position on the opposite side of the box. One robot is on the left side, called the *left pusher*, the second is on the right side and is called the *right pusher*.

Such initial configuration and the fact that the *watcher* is always in the same place do not reduce the complexity and the variability of individual tests.

The test is successfully completed if the centre of the box comes sufficiently close to the *watcher*, and is not successfully completed if the box touches a wall. The reason for this is that situations are possible, particularly at the beginning of learning, when robots get stuck pushing the box against a wall. Tests are, therefore, interrupted in order to eliminate the need for continuous human monitoring.

IV. DESCRIPTION OF SOLUTION

We have selected the variant with two pushers and a fixed target position, since our goal was to show and implement learning in a multi-agent environment applying reinforcement learning. The variant with two robots describes these aspects clearly enough.

Our solution is a combination of solutions in [4], [5] and [6]. The algorithm and the learning method is similar to the solution in [4] and [5]. However, we have introduced the division of labour (the *watcher* and the *pushers*) that is similar to the one in [6]. Also, we have significantly modified the role of the *watcher* by turning it into an immobile pushing target. Its task is to determine the orientation and distance of the box from the target – itself – and to transmit this information to the *pushers*.

The robot's sensors are limiting factors since, besides sonars, there are no other sufficiently robust devices for determining the location of the target. As previously stated, the *watcher* is positioned in the centre of the north wall, facing south. Its task is to determine the position and orientation of the box. The *watcher's* eight sonars are not even close to being sufficient for this task. We have enhanced its perception by cumulating sonar readings through time. The *watcher* periodically rotates, from left to right, screening with its sonars the entire room. 120 different readings are recorded for each 3° of the room. Thus, we obtain the picture of the entire room from the standpoint of the *watcher*. Based on the 120 sonar readings, the position and the orientation of the box are then determined, applying the following heuristic method: from all 120 readings, those that have "hit" the box are selected. A simple rule is applied: if it is not a wall, it is the box. The approximate value of a sonar reading "hitting" a wall is known in advance. If the sonar reads an obstacle that is closer to the anticipated position of the wall, it is clear that the obstacle is the box. The sonar readings reflecting from the box in most cases represent a series of sonar "hits" along the box. If from these readings we discern the line that represents the side of the box, the perpendicular line to this line shall represent the orientation of the box. Our method of determining this line is extremely simple, but yielding satisfactory results. Namely, the furthest and the closest sonar readings (from the *watcher*) hitting the box are taken into consideration. The connecting line is the target line. An exception is the situation when the box faces the *watcher* with its shorter side. This case is detected by testing the length of the obtained line. The position of the box is determined as the arithmetical mean of sonar readings that have "hit" the box.

Based on the orientation of the box, the state of the environment is determined applying the heuristic method, i.e. should the box be rotated to the *left* or to the *right*, or should it be pushed *straight*. The information whether any particular position of the box is better or worse than the previous is determined based on the position and the orientation of the box and serves as the credit or punishment to the *pushers* in their reinforcement learning. This information the *watcher* transmits to the *pushers*.

Two pushers rotate the box in the way that one pushes, while the second stands still. Two robots translate the box straight by pushing it simultaneously.

The *watcher* robot during reinforcement learning takes the role of the environment. *Pushers*, as reinforcement

learning agents, map the state into actions ("push" or "stop"). Changes from one state to the other are infrequent. Therefore, similar to the n-armed bandit problem [1], each state is learned separately. The appropriate function learning method for such non-stationary problems is the *exponential, recency-weighted average*

$$Q_{t+1} = Q_t + \alpha[\, r_{t+1} - Q_t] \qquad (1)$$

where, α is the learning step, $0 < \alpha \leq 1$, r_{t+1} is (k+1)st reward and Q_{t+1} is weighted average of past rewards. By developing this recursive function, it may be easily concluded that the credits awarded earlier gradually loose their significance in relation to those awarded more recently:

$$Q_{t+1} = \alpha\, r_t + (1-\alpha)\,\alpha\, r_{t-1} + (1-\alpha)^2\,\alpha\, r_{t-2} + \ldots. + (1-\alpha)^{k-1}\,\alpha\, r_1 + (1-\alpha)^k\, Q_0$$

where Q_0 is initial estimate. This function learning method is appropriate here primarily because, in multi-agent systems, during the early learning steps, it is not known whether the awarded credit is the consequence of one particular agent or some other agent. When learning stabilizes, i.e. when the value function comes closer to the actual value, the obtained credits are much more accurate and make for most of the value function. It is important to note that, in environments with higher uncertainty and noise, credits awarded in later steps are sometimes inaccurate as well, and it is good to set a lower value of parameter α so that a larger number of credit values awarded later would have more significance. In other words, the higher parameter α is, the lesser is the number of later credits which are significant in the approximation of the value function, while the remaining credits are negligible.

While learning, *pushers* approximate the value function applying a simple 3×2 table. Each robot has its own table and learns and acts completely independently from the other. None of the *pusher* robots has any knowledge on the state of the other.

Table 1. Table of value function of the left *pusher*

action/state	straight	left	right
push	3.4	1.1	0.5
stop	-1.5	3.6	-0.2

Table 1 shows an example of a well learned value function for the left *pusher*. The learning algorithm modifies the table according to formula (1). Actions are selected according to the ε-greedy policy [1]. The action having a higher value in the table for the state at a given moment is selected, but there is probability ε that an action with a lower value in the table is selected. The probability is the highest at the beginning and decreases with time. This enables better exploration at the beginning and much better knowledge usability during the advanced learning phase. Initially, both tables are filled with random values. At the end of learning, both tables should have, for the state *straight,* a higher value for the action *push.* The left *pusher* should have for the state *left* a higher value for the action *stop*, and for the state *right* a higher value for the action *push.*

Table 2. Table of value function of the right *pusher*

action/state	straight	left	right
push	11	2.9	2
stop	0.1	-1	5.1

A well learned value function table of the right *pusher* should resemble Table 2. For the state *straight*, values are the same as for the left *pusher*, while for the remaining two states the values are symmetrical. In other words, when the box is being pushed to the left, the left *pusher* stands still while the right *pusher* pushes. When the box is being pushed to the right, the situation is the opposite. When pushing the box straight, robots push it together.

In addition to learning actions, *pushers* are tasked, based on the readings from their sonars, to position themselves after each completed action perpendicular to the box. The task is to remain in a position suitable for pushing. Equalling the readings of the front sonars on the left and right sides by rotation carries out this task.

V. Discussion

The *watcher* robot can transmit to the *pushers* three different state values: *straight*, *left* or *right*. *Pusher* robots may react in the four different modes: both push, both do not move, the left pushes while the right does not move, and vice versa. The only thing that the robots need to learn is to map correctly the three different states to four different reactions. They are capable of achieving this after an average of 15 consecutive tests in the simulator. It is important to note that the learning is distributed and independent. Since the state space and the number of possible actions are so small, it would be possible to determine the behaviour of the *pushers* in advance. The solution involving learning can be further expanded on a larger number of agents or higher complexity of the task. The more complex the task is, it becomes more difficult to determine the optimal behaviour of agents. Therefore, machine learning arises as the best and, at times, the only solution.

The drawback of our solution is the oversignificant role of the *watcher*. Unless it carries out its task accurately, the entire procedure shall fail. This drawback seems more justifiable if we consider the following. Approaching our solution from the aspect of reinforcement learning, we note that *pusher* robots represent agents that learn, while the *watcher* robot represents the environment. The *watcher* transmits to the *pushers* the information on the state of the environment, awarding them credits or punishments. Based on the state of the environment, *pushers* change by their actions (pushing the box) the state of the environment. Clearly, the *pushers* are not capable of learning well nor executing the task unless they receive accurate information on the environment, and the environment is the *watcher*.

The state *straight* occurs relatively seldom. Therefore, is somewhat more difficult for the *pushers* to learn the right actions for this state. This drawback has no major consequences, since this state is promptly exited if the wrong action is selected. In the next state, the box will return to the right path. In the worst case, the robots will push the box left-and-right, which at the end results is a linear pushing path.

The credit assignment problem does not arise due to the reduced state space and the fact that changes in the state occur relatively infrequently. However, this problem is quite pronounced in multi-agent box-pushing implementations supported by learning [5].

Improvements are possible in the algorithm of the *watcher*. The part of the algorithm that determines sonar readings of the box might be better implemented by means of a neural network. In a way, this is a classification problem (classification of sonar readings into two classes) and, therefore, suitable for this method. The gain is adaptability. New configurations of the environment would be automatically learned without the need for manual modification of the algorithm.

VI. CONCLUSION

This paper has addressed the box-pushing problem involving three robots equipped with a series of eight sonars each. Our solution is based on the *pusher-watcher* approach where two *pushers* robots learn the best strategy for cooperatively moving an oversized elongated box to a specified goal and one *watcher* robot acting as the environment.

The *pusher* robots see an overlimited picture of the world and are incapable, using only their own sensors, to fulfil the task. Most importantly, they are incapable of perceiving the position of the goal. The communication between *pushers* and *watcher* is, therefore, essential.

Our solution is based on the reinforcement learning executed by *pushers* independently one from the other. The goal is achieved after an average of 15 consecutive tests in the simulator.

The solution to the box-pushing problem proposed here serves as a base for future laboratory experiments that may be further extended by increasing the number of *pusher* robots, placing the robots and boxes in a more complex set-up with obstacles and enabling the *watcher* to move.

REFERENCES

[1] R. S. Sutton, A. G. Barto, "Reinforcement Learning, An Introduction", MIT Press, 1998.

[2] J. Ferber, "Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence", Addison-Wesley, 1999.

[3] P. Stone, "Layered Learning in Multiagent System", The MIT Press, 2000.

[4] M.J. Matarić, M. Nilsson, K.T. Simsarian, "Cooperative Multi-robot Box-pushing", Proceedings, *IROS-95,* Pittsburgh, PA, 1995, 556-561.

[5] K.T. Simsarian, M.J. Matarić, "Learning to Cooperate Using Two Six-Legged Mobile Robots", *Proceedings of the Third European Workshop of Learning Robots,* Heraklion, Crete, Greece, Apr 28-29, 1995.

[6] B.P. Gerkey, M.J. Matarić, "Pusher-watcher: An Approach to Fault-tolerant Tightly-coupled Robot Coordination", *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2002),* Washington, DC, May 11 - 15, 2002, 464-469.

[7] M.J. Matarić, "Using Communication to Reduce Locality in Multi-robot Learning", *Journal of Experimental and Theoretical Artificial Intelligence,* special issue on Learning in DAI Systems, Gerhard Weiss, ed., 10(3), Jul-Sep, 1998, 357-369.

[8] M.J. Matarić, "Reinforcement Learning in the Multi-robot Domain", *Autonomous Robots,* 4(1), Mar 1997, 73-83.

[9] S. Mahadevan, J. Connell, "Automatic Programming of Behavior-based Robots Using Reinforcement Learning", *Artificial Intelligence* , vol. 55, Nos. 2-3, pp. 311-365, June, 1992.