Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Flávio Moreira de Oliveira (Ed.)

Advances in Artificial Intelligence

14th Brazilian Symposium
on Artificial Intelligence, SBIA'98
Porto Alegre, Brazil, November 4-6, 1998
Proceedings

Springer

Series Editors

Jaime G. Carbonell, Carnegie Mellon University, Pittsburgh, PA, USA
Jörg Siekmann, University of Saarland, Saarbrücken, Germany

Volume Editor

Flávio Moreira de Oliveira
Instituto de Informática - PUCRS
Av. Ipiranga 6681, prédio 30, bloc 4
90619-900 Porto Alegre - RS, Brazil
E-mail: flavio@kriti.inf.pucrs.br

Cataloging-in-Publication Data applied for

Die Deutsche Bibliothek - CIP-Einheitsaufnahme

Advances in artificial intelligence : proceedings / 14th Brazilian Symposium on Artificial Intelligence, SBIA '98, Porto Alegre, Brazil, November 4 - 6, 1998. Flávio Moreira de Oliveira (ed.). - Berlin ; Heidelberg ; New York ; Barcelona ; Budapest ; Hong Kong ; London ; Milan ; Paris ; Singapore ; Tokyo : Springer, 1998
(Lecture notes in computer science ; Vol. 1515 : Lecture notes in artificial intelligence)
ISBN 3-540-65190-X

CR Subject Classification (1998): I.2

ISBN 3-540-65190-X Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera ready by author
SPIN 10692710 06/3142 – 5 4 3 2 1 0 Printed on acid-free paper

Preface

The Brazilian Symposium on Artificial Intelligence (SBIA) has been organized by the Interest Group on Artificial Intelligence of the Brazilian Computer Society (SBC) since 1984. In order to promote research in Artificial Intelligence and scientific interaction among Brazilian AI researchers and practitioners, and with their counterparts worldwide, it is being organized as an international forum since 1993. The SBIA proceedings have been published by Springer-Verlag as a part of the Lecture Notes in Artificial Intelligence (LNAI) series since 1995.

The XIVth SBIA, held in 1998 at the PUCRS Campus in Porto Alegre, has maintained the international tradition and standards previously established: 61 papers were submitted and reviewed by an international program committee, from this number, 26 papers were accepted and are included in this volume.

Of course, organizing an event such as SBIA demands a lot of group effort. We would like to thank and congratulate all the program committee members, and the many reviewers, for their work in reviewing and commenting on the submitted papers. We would also like to thank the Pontifical Catholic University of Rio Grande do Sul, host of the XIV SBIA, and the institutions which sponsored it - CNPq, CAPES, BANRISUL, among others. Last but not least, we want to thank all the kind people of the Local Organizing Committee, whose work made the event possible.

Porto Alegre, November 1998

Flávio Moreira de Oliveira

Program Committee Members

*Flávio M. de Oliveira (Instituto de Informática - PUCRS, Brazil) ***
Antonio Carlos da Rocha Costa (Univ. Católica de Pelotas, Brazil)
Luis Otávio Alvares (Instituto de Informática - UFRGS, Brazil)
Dibio Borges (Univ. Federal de Goiania, Brazil)
Celso Kaestner (CEFET - PR, Brazil)
Tarcisio Pequeno (LIA - Univ. Federal do Ceará, Brazil)
Edson de Carvalho Filho (DI - UFPE, Brazil)
Ariadne Carvalho (Unicamp - Brazil)
Maria Carolina Monard (USP - São Carlos, Brazil)
Guilherme Bittencourt (LCMI - Univ. Federal de S. Catarina, Brazil)
Sandra Sandri (Inst. Nacional de Pesquisas Espaciais, Brazil)
Wagner Silva (UnB, Brazil)
Edilson Ferneda (Univ. Federal da Paraíba, Brazil)
Helder Coelho (Univ. de Lisboa, Portugal)
Yves Demazeau (Lab. Leibniz/IMAG - Grenoble, France)
Barbara Hayes-Roth (Stanford Univ. - USA)
Stefan Wrobel (GMD, Germany)
Manuela Veloso (Carnegie Mellon Univ. - EUA)
John Self (Leeds Univ. - UK)
Peter Ross (Edinburgh Univ. - UK)

Local Organizing Committee

Ana L.V. Leal, Gabriela Conceição, Iára Claudio, João L.T. da Silva, Lúcia M.M. Giraffa, Michael C. Mora, all the people at II-PUCRS, Ricardo Annes, Ricardo M. Bastos, Thais C. Webber, The PET Group.

List of Reviewers

Alexandre Ribeiro
Ana Teresa de Castro Martins
Antonio Carlos da Rocha Costa
Ariadne Carvalho
Arnaldo Moura
Barbara Hayes-Roth
Bertilo F. Becker
Celso Kaestner
Dibio Borges
Doris Ferraz de Aragon
Edilson Ferneda
Edson de Carvalho Filho
Eloi F. Fritsch
Flávio M. de Oliveira
Flavio Soares Correa da Silva
Germano C. Vasconcelos
Guilherme Bittencourt
Hans-Jorg Schneebeli
Helder Coelho
Jacques Wainer
João Balsa
John Self
Lúcia M.M. Giraffa
Luis Antunes
Luis Moniz
Luis Otávio Alvares
Manuela Veloso
Mara Abel
Marcelino Pequeno
Maria Carolina Monard
Maria das Gracas Volpe Nunes
Mathias Kirsten
Michael C. Mora
Patrick Doyle
Paulo J. L. Adeodato
Peter Ross
Raul S. Wazlawick
Riverson Rios
Rosa M. viccari
Sandra Sandri
Solange Oliveira Rezende
Stefan Wrobel
Tarcisio Pequeno
Teresa B. Ludermir
Thomas F. Gordon
Vera L.S. de Lima
Wagner Silva
Wilson R. de Oliveira Jr.
Yves Demazeau

Goal-Directed Reinforcement Learning Using Variable Learning Rate

Arthur P. de S. Braga, Aluizio F. R. Araújo

Universidade de São Paulo, Departamento de Engenharia Elétrica,
Av. Dr. Carlos Botelho, 1465, 13560-250, São Carlos, SP, Brazil
{arthurp, aluizioa}@sel.eesc.sc.usp.br

Abstract. This paper proposes and implements a reinforcement learning algorithm for an agent that can learn to navigate in an indoor and initially unknown environment. The agent learns a trajectory between an initial and a goal state through interactions with the environment. The environmental knowledge is encoded in two surfaces: the reward and the penalty surfaces. The former deals primarily with planning to reach the goal whilst the latter deals mainly with reaction to avoid obstacles. Temporal difference learning is the chosen strategy to construct both surfaces. The proposed algorithm is tested for different environments and types of obstacles. The simulation results suggest that the agent is able to reach a target from any point within the environment, avoiding local minimum points. Furthermore, the agent can improve an initial solution, employing a variable learning rate, through multiple visits to the spatial positions.

1 Introduction

The motivation of this work is to propose and implement an agent that learns a task, without help from a tutor, in a limited and initially unknown environment. The agent is also capable of improving its performance to accomplish the task as time passes. Such a task is concluded when the agent reaches an initially unknown goal state. This problem is defined by Koening and Simmons [8] as a goal-directed reinforcement learning problem (GDRLP) and it can be divided in two stages. The goal-directed exploration problem (GDEP) involves the exploration of the state space to determine at least one viable path between the initial and the target states. The second phase uses the knowledge previously acquired to find the optimal or sub-optimal path.

Reinforcement learning (RL) is characterized by an agent that extracts knowledge from its environment through trial-and-error [6] [11]. In this research, learning is accomplished through a RL method that takes into consideration two distinct appraisals for the agent state: the reward and penalty evaluations. On one hand, a reward surface, modified whenever the agent reaches the target, indicates paths the agent can follow to reach the target. On the other hand, the penalty surface, modified whenever the agent reaches an obstacle, is set up in a way that the agent

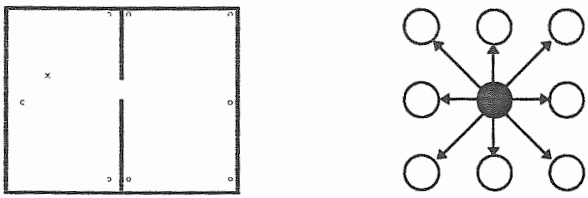
can get rid of obstacles on its way. Thus, the reward surface deals basically with planning [12] [14] whilst the penalty surface deals mainly with reactive behavior [4] [13]. The composition of the two surfaces guides the agent decisions avoiding the occurrence of any local minimum. The two surfaces are constructed employing temporal difference (TD) as the RL learning strategy.

The tests consist in simulating a point robot within an indoor environment. Initially the robot does not have any information about environment and about the goal state. Hence, the main objective of this work is to propose an algorithm that allows the robot to reach the target from any position within the environment through an efficient path. The tests involve obstacles with different levels of complexity. The results suggest that the target is reached from any position within the environment and the chosen path is improved during the second stage of the problem solution. Despite the final trajectory is better than that one initially generated, the final path is not necessary the very best one. Moreover, the improvement takes very long to be reached.

This paper is structured as follows. The next section presents the task of this work. Section 3 introduces the RL algorithm used. The exploration stage is treated in Section 4 and the improvement of the resulting paths is discussed in Section 5. Finally, Section 6 presents the conclusions.

2 Navigation in an Indoor and Initially Unknown Environment

The proposed environment is based on the physical structure of our laboratory: a room divided into two compartments with a single passage between them (Figure 1.a). In this figure, the circles represent initial training states and the "X" represents a particular target. After the learning phase, the robot should reach the target from any point within the environment. The algorithm was also tested for different shapes of obstacles immersed into a single-room in order to evaluate how well the algorithm generalizes its achievements. These tests are based on the experiments proposed by Donnart and Meyer [5] and Millán [9]. The different environments are the following: a free room, a room with a barrier, a room with an U obstacle and a labyrinth.



(a) (b)

Fig. 1. (a) Sketch of the chosen environment. (b) A particular state (in black) and its neighborhood: circles represent states and arrows are possible actions

All simulated environments are taken as a discrete and limited space state in which each state has eight neighbor states (Figure 1.b). An agent action is a movement from a given state to one of its neighbors.

3 Using TD to Acquire Knowledge about the Environment

The navigation problem can be viewed as a decision tree in which each node corresponds to a possible state s. The search in this tree is heuristically guided by the evaluations for any possible action a the agent can take from the state s. The agent has to learn how to evaluate the actions from scratch in autonomous way using only environmental feedback. Hence, RL is a suitable approach to solve GDRLP.

In RL, the agent receives a reinforcement signal r from its environment that quantifies the instantaneous contribution of a particular action in the state s to reach the goal state. Then, the evaluation of a state s under a policy π, denoted by $V^{\pi}(s)$, estimates the total expected return starting the process from such a state and following that mentioned policy [11]. The learning process takes into consideration three types of situations: free states, obstacle states, and goal state. A free state is defined as a state in which the agent does not reach the goal (goal state) or an obstacle (obstacle state). Thus, Araújo and Braga [1] proposed the following definition for the reinforcement signal:

$$r(s_t, a, s_{t+1}) = \begin{cases} +1, \text{ goal state;} \\ -1, \text{ obstacle state;} \\ 0, \text{ free state.} \end{cases} \quad (1)$$

This representation is derived from the two most common types of representations found in the literature: the goal-reward representation [10] and the action-penalty representation [3]. The former indicates the states to guide the agent towards the goal. Consequently, it is a very sparse representation because, very often, there is only a single target. The latter is less sparse than the first representation because the occurrence of obstacles is much more common than the presence of goals. Even though, this representation does not direct the agent towards the target.

The learning strategy for solving this GDRLP is the temporal difference (TD) as used by Barto *et al.* [2]. This technique is characterized for being incremental and for declining a model of world. The chosen learning rule is the following [11]:

$$\Delta V_t^{\pi}(s_t) = \alpha\left[r_{t+1} + \gamma V_t^{\pi}(s_{t+1}) - V_t^{\pi}(s_t)\right] \quad (2)$$

where: $V^{\pi}(s_t)$ is the evaluation of the state s; $\Delta V^{\pi}(s_t)$ is the updating value of the evaluation of the state s_t; s_t is the state at time t; r_{t+1} is the feedback from the environment at time $t+1$; α is the learning rate ($0 \leq \alpha \leq 1$); γ is the discount factor ($0 \leq \gamma \leq 1$).

The agent learns the navigation task accumulating knowledge into two surfaces: a reward and a penalty one. The reward surface does not explicitly map obstacles, it

is modified following the identification of the goal in the environment. The penalty surface holds information that indicates the proximity between the agent and any obstacle. The composition of both surfaces guides the robot towards the target. The learning algorithm is shown in Figure 2.

```
Initialize with zeros all states in both surfaces;
Repeat
    Take the agent to the initial position;
    Repeat
        Select an action a according to the policy π;
        Execute the action a and take the agent to its new state s_{t+1};
        If the new state is an obstacle
            Update the penalty evaluation for s_t according to (2).
        Store the current state in the current trajectory;
    While the agent does not find an obstacle or a target;
    If the new state is a target
        Update the reward evaluation for the whole memorized trajectory
        according to (2);
        Delete the last trajectory from the agent memory.
While the agent does not find the target for a pre-determined number of times.
```

Fig. 2. The learning algorithm in procedural English.

The agent policy π is inspired in the potential field methods [7]:

- In 85% of the attempts the algorithm chooses the action with larger combined value (reward + penalty);
- In 10% of the attempts the algorithm chooses the action that takes the agent to the opposite state with the largest penalty;
- In 5% of the attempts the algorithm chooses the action with the largest reward.

The timing to update each one of the surfaces is different. The penalty surface is modified whenever the agent changes its state and the algorithm takes into consideration only the current state and its successor. The changes follow every single visit to a particular state. The reward surface is updated whenever the target is encountered. The modification involves all the states the agent visited to reach the goal. This trajectory is stored by the agent while it tries to reach the goal.

The algorithm above was implemented in MATLAB® where the graphic user interface (GUI) permitted to visualize some behaviors of the agent. The two next sections give the details about the results obtained in the two stages of GDRLP.

4 Solving GDEP

The objective of the first stage of GDRLP is to explore the workspace to generate trajectories, not necessarily the shortest ways, between the initial states and the goal state. The algorithm is tested in five different environments: layout of our laboratory, free room, room with barrier, room with U obstacle and labyrinth.

The learning stage generates surfaces as illustrated in Figures 3.a and 3.b. The reward surface assigns values between zero and one to each visited state in the environment. These values represent the proximity to the goal state. The penalty surface maps the obstacles positions that the agent tried to visit. The penalty evaluations range from -1 to zero and such values represent the proximity to obstacles. Note that the reward surface is smoother than the penalty one.

Figure 3.c shows a number of trajectories obtained after initial exploration of the first environment. Within the explored area, the trajectory generated by the agent is a viable connection between an initial point and the target but, generally, it is not the best path. The results shown in Figure 3.c illustrate that certain parts of the trajectories are repeated for different starting points. Thus, when leaving from a nontrained initial point, the agent moves along the state space to reach a point that had been visited during the training stage. From that point onwards, the robot follows the trajectory to which that point belongs, the agent stops its movement after reaching the goal.

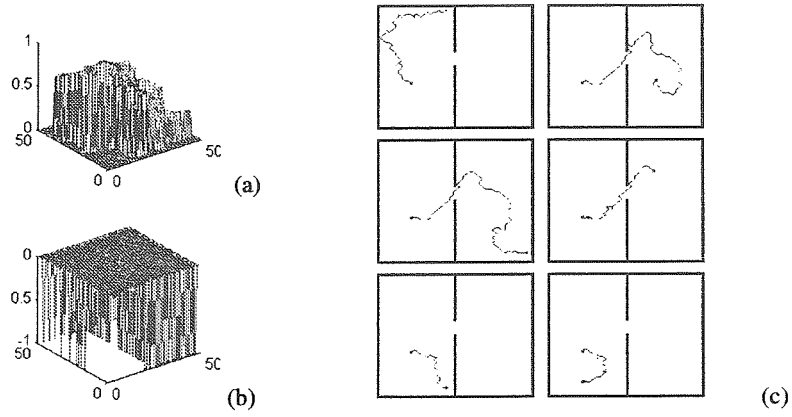


Fig. 3. (a) A reward surface learned by the agent. (b) A penalty surface learned by the agent. (c) Trajectories obtained with the TD algorithm: the left-hand side shows initial trained points and the right-hand side initial nontrained points. For all trajectories the goal state is located at the center of the left-hand side part of the environment.

Figures 4.c and 4.f show agent paths in one-room environment with and without a barrier. The corresponding reward surfaces for visited states in both cases (Figures 4.a and 4.d) are similar, even so the penalty surfaces are significantly different. Figure 4.e shows many more details about the obstacles than Figure 4.b because in the first one the agent had to work further to find a viable path.

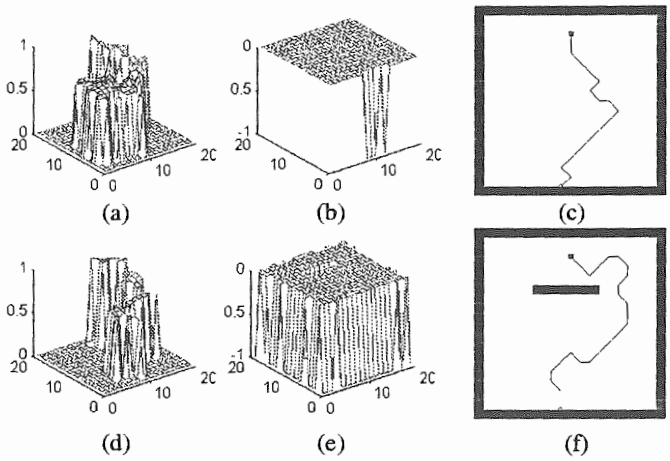


Fig. 4. Reward surface, penalty surface and obtained trajectory on a agent in a: (a)-(c) free single room; (d)-(f) single room with a barrier. The target is at the uppermost point of the path.

More complex environments are also tested, as presented in Figures 5.a and 5.b. Note that the agent has successfully reached the target in both cases.

Fig. 5. Two paths found by the agent considering: (a) An U obstacle with the initial position inside the U, (b) A labyrinth with the initial position in the right lowest corner of the figure.

Based on the proposed algorithm, the agent produces trajectories to reach the goal. However, if the evaluations are not updated, the trajectories remain be the same ones, and no decrease in the number of visited states occurs. A mechanism to suitably modify the reward surface to accomplish the second stage of GDRLP is presented in the next section.

5 Solving the Second Stage of GDRLP

At this point, the agent already has coded into its two surfaces the information that allows it to navigate appropriately along the areas visited during the exploration

stage. As mentioned before, the evaluation of each state is updated only once. If that constraint is lifted, the multiple visitation originates "isles" in the reward surface (Figure 6.a). As a consequence, the agent does not have suitable reward evaluation in the area between isles then the agent is arrested in a loop as sketched in Figure 6.b. The penalty surface, with its local update scheme, does not have significant change in this extra training.

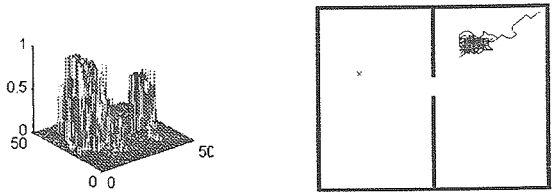


Fig. 6. (a) "Isles" generated in the reward surface. (b) An agent get stuck due to multiple visitation during the training stage.

The trajectory pursued by the agent during the training stage (Figure 7.b) explains the occurrence of isles in the reward surface. A particular state, visited a number of times in a same trajectory, has its evaluation modified proportionally to the number of visits. Thus, very often this evaluation becomes incoherent with those in the neighborhood presenting significant differences between them. Hence, one can conclude that the modifications in the reward surface should become smother in the current visit than the change in the previous visit in order to avoid surfaces as showed in Figure 7.a.

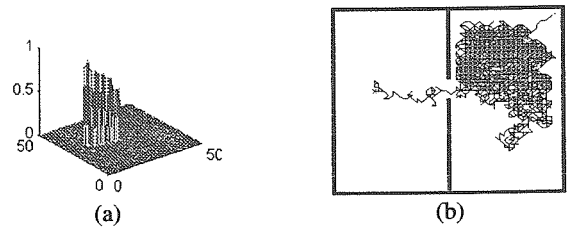


Fig. 7. (a) The reward surface for learning with multiple visitation. (b) The corresponding trajectory to reach the goal.

In sum, the constrain of updating just one time the evaluation of each state guarantees the solution of GDEP. However, this restriction does not allow the agent to alter the learned trajectory in a way to get a better solution. The possible solution is to adopt a variable learning rate in the algorithm.

The learning rate determines the size of the step to update the evaluations during the training stage. It is adopted a variable learning rate per state that is initially high and diminishes as function of the number of visits. The decreasing of the learning rate should be abrupt to avoid the occurrence of the isles in the reward surface. The expression used for the learning rate to update the reward surface is:

$$\alpha(s) = \alpha_2 \left(\frac{\alpha_1}{\alpha_2} \right)^{\frac{k(s)-1}{k(s)}} \tag{3}$$

where[1]: $\alpha(s)$ is the learning rate of state s; α_1 is the minimum value of all learning rates ($\alpha_1 = 0.1$); α_2 is the maximum value of all learning rates ($\alpha_2 = 0.9998$); k(s) is the number of visits to state s in trials that reached the goal.

The choice of an action is based on a composition of the reward and the penalty. Thus, the policy [2] becomes:

- In 60% of the attempts the algorithm chooses the action with the largest combined value (reward + penalty);
- In 10% of the attempts the algorithm chooses the action that takes the agent to the opposite state with the largest penalty;
- In 5% of the attempts the algorithm chooses the action with the largest reward;
- In 25% of the attempts the algorithm chooses the action randomly.
- In case of tie the choice is random.

The first three components of the policy provides the agent with a tendency to follow the growth of reward evaluation and to avoid the growth of penalty evaluation. The fourth component, however, adds a random component to the agent behavior to allow variations in the learned paths. The randomness allows to find shorter trajectories as long as the number of training cycles[3] increases.

During the training cycles, the penalty surface changes slightly, i.e., the learning basically occurs in the reward surface. Following the increase in the number of training cycles, the reward surface maps more states and becomes smoother (Figure 8).

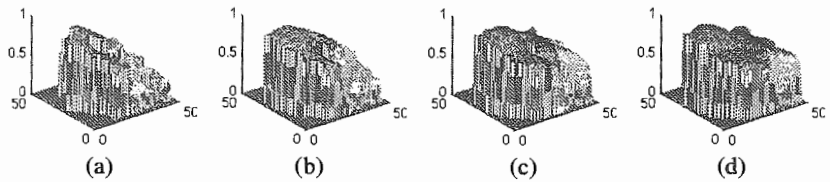


Fig. 8. A reward surface after: (a) 1 training cycle, (b) 100 training cycles, (c) 1000 training cycles and (d) 5574 training cycles.

The trajectories in Figure 9 exemplify the improvement of the agent performance (Figure 8). During these tests, the policy becomes that one explained in Section 3.

---

[1] The parameters adopted in the learning rate calculus are chosen heuristically.

[2] The distribution used in the policy tries an tradeoff between a determinist and a random behavior and it was selected heuristically.

[3] In simulations, a training cycle corresponds to the random selection and training of the eight initial points and a final state as in Figure 1.a.

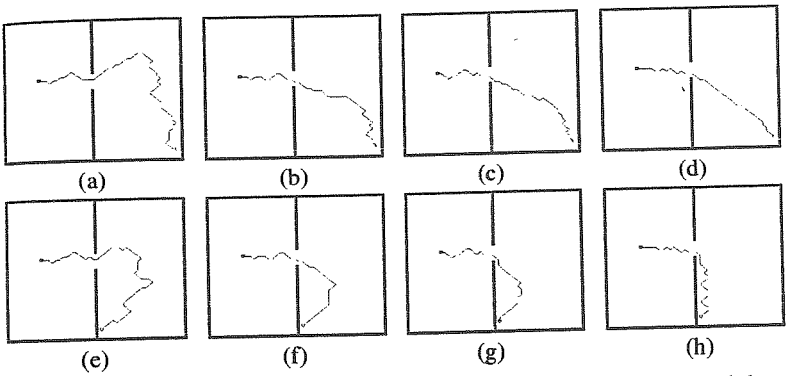


Fig. 9. (a)-(d) Improvement of the agent performance between an initial point and the goal state and (e)-(h) Improvement of the agent performance between an initial point and the goal after 1, 100, 1000 and 5574 training cycles. The final point is placed at the left-hand side of the room.

Figure 10 shows that after ten thousand training cycles the trajectories in other environments are also better then before, however they are not the shortest paths.

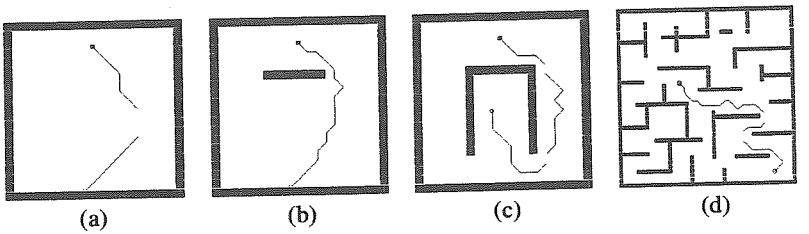


Fig. 10. (a)-(d) Trajectories followed by the agent after 10000 stages in the environments presented in Figures 4 and 5.

6 Conclusions

This paper proposed an agent that uses reinforcement learning methods to guarantee its autonomy to navigate in its state space. The reinforcement signal from the environment is encoded into the reward and penalty surfaces to endow the agent with the ability to plan and to behave reactively. The agent solves GDRLP in which an exploration stage finds a solution to the proposed task and after that the agent performance is improved following further training.

The agent performance is assessed using three out of four evaluations methods proposed by Wyatt *et al.* [15]. The internal estimates of the performance, embodied in the two surfaces, are able to represent the action repercussions. The qualitative analysis of the robot behavior shows that it improves its performance with further training. Finally, the quantitative external analysis of performance results in shorter paths since when the robot finds its goal for the first time.

The agent solves GDRLP, even so the time to reach a sub-optimal solution is quite long. It is necessary a great amount of training cycles to generate trajectories better than the early ones. The authors are working for learning strategies that dynamically modify the policy, so the randomness in agent behavior decreases with the increase of visitations to states of the environment.

References

1. Araújo, A.F.R., Braga, A.P.S.: Reward-Penalty Reinforcement Learning Scheme for Planning and Reactive Behavior. In Proceedings of IEEE International Conference on Systems, Man, and Cybernetics (1998).
2. Barto, A. G., Sutton, R. S. and Anderson, C. W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Transactions on SMC. V. 3, N. 5, pp: 834-846. (1983).
3. Barto, A. G.; Bradtke, S. J. and Singh, S. P.: Learning to act using real-time dynamic programming. Artificial Intelligence. V. 73, N. 1, pp: 81-138. (1995).
4. Brooks, R. A.: A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation. V. RA-2, N. 1, pp: 14-23. (1986).
5. Donnart, J. -Y. and Meyer, J. -A. Learning reactive and planning rules in a motivationally autonomous animat. IEEE Transactions on SMC, V. 26, N. 3, pp: 381-395. (1996).
6. Kaelbling, L. P., Littman, M. L. and Moore, A. W. Reinforcement learning: A survey. Journal of Artificial Intelligence Research, V. 4, pp: 237-285. (1996).
7. Khatib, O. Real-time obstacle avoidance for manipulators and mobile robots. In Proceedings of IEEE International Conference on Robotics Automation. St. Louis, MO. pp: 500-505. (1995).
8. Koenig, S. and Simmons, R. G. The effect of representation and knowledge on goal-directed exploration with reinforcement learning algorithms. Machine Learning, V. 22, pp: 227-250. (1996).
9. Millán, J. del R. Rapid, safe, and incremental learning of navigation strategies. IEEE Transactions on SMC, V. 26, pp: 408-420. (1996).
10. Sutton, R. S. Integrated architectures for learning, planning and reacting based on approximating dynamic programming. In Proceedings of International Conference on Machine Learning. pp: 216-224. (1990).
11. Sutton, R.S. and Barto, A. An Introduction to Reinforcement Learning. Cambridge, MA: MIT Press, Bradford Books. (1998).
12. Thrun, S., Moeller, K. and Linden, A. Planning with an Adaptive World Model. In Advances in Neural Information Processing Systems (NIPS) 3, D. Touretzky, R. Lippmann (eds.), Morgan Kaufmann, San Mateo, CA. (1991).
13. Whitehead, S. D. and Ballard, D. H. Learning to perceive and act by trial and error. Machine Learning, V. 7, pp: 45-83. (1991).
14. Winston, P. H. Artificial Intelligence. Reading, UK: Addison Wesley. (1992).
15. Wyatt, J., Hoar, J. and Hayes, G. Design, analysis and comparison of robot learners, accepted for publication in Robotics and Autonomous Systems: Special Issue on quantitative methods in mobile robotics, Ulrich Nehmzow, Michael Recce and David Bisset (eds). (1998).