INFORMATICA, 2003, Vol. 14, No. 2, 237–250

Efficient Exploration in Reinforcement Learning Based on Utile Suffix Memory *

Arthur PCHELKIN

Faculty of Computer Science and Information Technology
Riga Technical University
1 Kalku Str., LV-1658 Riga, Latvia
e-mail: arturp@balticom.lv

Received: March 2003

Abstract. Reinforcement learning addresses the question of how an autonomous agent can learn to choose optimal actions to achieve its goals. Efficient exploration is of fundamental importance for autonomous agents that learn to act. Previous approaches to exploration in reinforcement learning usually address exploration in the case when the environment is fully observable. In contrast, we study the case when the environment is only partially observable. We consider different exploration techniques applied to the learning algorithm "Utile Suffix Memory", and, in addition, discuss an adaptive fringe depth. Experimental results in a partially observable maze show that exploration techniques have serious impact on performance of learning algorithm.

Key words: reinforcement learning, exploration, hidden state, short-term memory.

1. Introduction and Problem Statement

Reinforcement learning (Sutton *et al.*, 1998; Kaelbling *et al.*, 1996; Mitchel, 1999) examines how an autonomous agent (Maes, 1994) that senses and acts in its environment can learn to choose optimal actions to achieve its goals. It is related to the problem how such agents can learn successful control policies by experimenting in their environment. It is assumed that the goals of the agent can be defined by a reward function that assigns a numerical value to each distinct action the agent may take from each distinct state. The task of the agent is to perform sequences of actions, observe their consequences, and to learn a control policy. It is desired to find a control policy that maximizes the reward accumulated over time by the agent.

The problem of learning a control policy to maximize cumulative reward is very general and covers many problems beyond robot learning tasks. This includes, for example, manufacturing optimization problems or sequential scheduling problems.

The task of the agent is to learn a target function that maps each possible state to the optimal action. This learning task is defined as a Markov decision process. In the Markov decision process the agent can perceive a set of distinct states of its environment and has

*This research was supported in part by the Latvian Science Foundation under grant No.02-86d.

a set of actions that it can perform. At each discrete time step, the agent senses the current state, chooses a current action, and performs it. The environment responds by giving the agent a reward and by producing the succeeding state. The structure of the environment is not known to the agent and is presented by a black box.

Usually in practice multiple situations are indistinguishable from immediate perceptual input. These multiple situations may require different responses from the agent. Usual reinforcement learning techniques, such as Q-learning (Mitchel, 1999), can't be applied in partially observable domains. So, there is a need to solve this incomplete perception problem. One of the possible ways is to use short-term memory. This short-term memory is used to overcome the incomplete perception problem. Using the short-term memory it is possible to distinguish multiple situations that are indistinguishable from immediate perceptual input.

Efficient exploration plays a fundamental role (Thrun, 1992b) for autonomous agents that learn to act. In many reinforcement learning algorithms undirected exploration techniques are used. While undirected exploration techniques, e.g., random walk exploration, utilize no exploration-specific knowledge and ensure randomness into action selection, directed techniques rely on knowledge about the learning process itself, allowing for exploring in a more directed manner (Thrun, 1992a). In many finite deterministic domains, any learning technique based on undirected exploration is inefficient in terms of learning time, i.e., learning time is expected to scale exponentially with the size of the state space (Whitehead, 1991).

Efficient exploration in partially observable domains is a special difficulty (Chrisman, 1992). Previous approaches (Thrun, 1992a; Sutton, 1990) to exploration in reinforcement learning usually address exploration in the case when the environment is fully observable. In contrast, McCallum (McCallum, 1997) considers efficient exploration in the partially observable environment. In our study, we continue this research by experimental comparing of McCallum's techniques with some other different approaches. Like McCallum, we also use his learning algorithm "Utile Suffix Memory" (USM).

2. Efficient Exploration with USM

Utile Suffix Memory (McCallum, 1995) is a reinforcement learning algorithm that uses variable amounts of short-term memory in order to solve tasks in partially observable environments. USM organizes its short-term memory in a Suffix Tree (Ron *et al.*, 1994). The agent maintains, using dynamic programming, learned Q-values – estimates of expected future discounted reward for each state-action pair. These Q-values are used for action selection. The leaves of the tree are the internal states of the agent (see Fig. 1; percepts are indicated by integers, actions by letters; fringe nodes are drawn in dashed lines; nodes labeled with a Q are nodes that hold Q-values).

The leaves may have different depths in different parts of the tree. Deeper branches of the tree correspond to the distinctions based on observations and actions further back in time. The structure can be understood as a n-order Markov model (i.e., a length n history window) with varying n in different parts of state space.

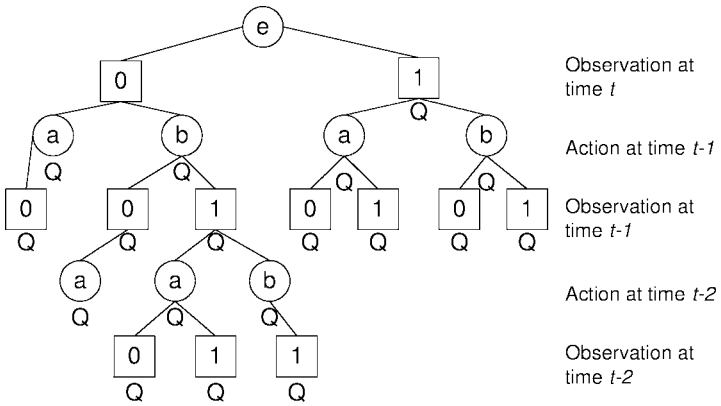

Fig. 1. A USM-style suffix tree.

Before learning, the tree begins with a single layer of nodes below the root (an order-0 Markov model, no memory), then during training, the tree grows as the agent discovers what memories are relevant to the task at hand. It finds these relevant memories by adding branches to the tree below which are normally considered the leaves, and then performing a Utile Distinctions Test on the hypothesis distinctions. The test asks the question "Is there a statistically significant difference between the utilities (utility is calculated using Q-values) of the newly distinguished states?" When answer is "Yes" it means that we have found a violation of the Markov property, and the hypothesis distinctions are promoted to official leaves of the tree, thus becoming real distinctions of agent's internal state space. The layers of hypothesis distinctions are called "fringes". The depth of the fringe is a configurable parameter of USM.

2.1. *Keeping Exploration Statistics in States or Fringes*

USM has shown efficient learning in the partially observable domain (McCallum, 1995). However, originally USM uses only undirected exploration technique: it selects a random action with probability e. McCallum (1997) has improved that property, he has proposed three more effective techniques. We will consider two techniques that have shown the best results, these are:

- *Counter-based Exploration* uses counts of the number of times an action was taken from a particular state (or fringe), and tries to choose actions that have been chosen less frequently;
- *Recency-based Exploration* (Sutton, 1990) uses counts of the number of time steps that have passed since an action was taken from a particular state (or fringe).

He has considered two cases: exploration statistics is associated with states (this is usual for reinforcement learning) or it is associated with fringes in the same manner.

2.2. *Giving the Internal Reward for Exploration*

We have found that McCallum techniques may fail in the case when the environment isn't reversible, for example, if there are one direction ways. In the last case, it may be difficult to find a goal first time. USM uses Q-values to discover distinctions in the environment, but these Q-values are accessible only when the agent has reached the goal and has received the reward from the environment at least one time. Until this, agent is unable to discover history distinctions and, thus, is unable to overcome incomplete perception problem. This problem can be solved by giving the agent additional internal reward for state space exploration. Receiving additional internal reward for exploration USM optimizes its control policy not only for exploiting the environment but also for exploration in the same manner.

We rely on hypothesis that *the perceptual distinctions discovered during exploration will help the agent to reach the goal state*. In general, there is no principal difference between exploitation and exploration because in both cases the goal is to reach some special states of the world. In many cases distinctions needed for reaching the goal state are also needed for reaching some particular state.

For each step in the world, the USM does one step of value iteration (Bellman, 1957), with the leaves of the tree as states. Value iteration consists of performing one step of dynamic programming on the Q-values:

$$Q^{new}(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} Pr\,(s'|s,a)\,U(s'), \tag{1}$$

where $Q(s,a)$ are estimates of the expected future discounted reward for executing action a from state s, $R(s,a)$ is the estimated immediate reward for executing action a from state s, $Pr(s'|s,a)$ is the estimated probability that the agent arrives at state s' given that it executed action a from state s, and $U(s')$ is the utility of state s', calculated as $U(s') = \max_a Q(s',a)$.

We propose to give the agent the reward for exploration by modifying the above value iteration procedure in this way:

$$Q^{new}(s,a) \leftarrow R(s,a) + \gamma \sum_{s'} Pr(s'|s,a)U(s') + \frac{1}{\alpha} ER(s,a), \tag{2}$$

where $ER(s,a)$ is the internal reward for exploration – the reward for executing action a from state s, and α is a constant gain weighting exploration versus exploitation. This exploration technique means that there is no need to use any additional undirected or directed exploration for action selection. The agent can deterministically choose the action with highest Q-value. The idea of using dynamic programming for efficient exploration has been considered by (Sutton, 1990), but it was applied only in fully observable domains.

The reward for exploration can be calculated in different ways, but we consider only two:

• *Counter-based Exploration*:

$$ER(s,a) = \begin{cases} \dfrac{\delta\dfrac{t}{n} - c(s,a)}{c(s,a)+1} & \text{if } c(s,a) < \delta\dfrac{t}{n}, \\ 0 & \text{otherwise;} \end{cases} \tag{3}$$

• *Recency-based Exploration*:

$$ER(s,a) = \begin{cases} \left(\rho(s,a) - \dfrac{n}{\delta}\right)\dfrac{\delta}{n} & \text{if } \rho(s,a) > \dfrac{n}{\delta}, \\ 0 & \text{otherwise;} \end{cases} \tag{4}$$

where δ is a constant $(0 < \delta < 1)$ that defines how frequently each state-action pair must be explored, $c(s,a)$ (see formula 3) is the number of times action a was executed from state s, $\rho(s,a)$ is the number of steps since action a was last executed from state s, n is a total amount of state-action pairs, calculated as $n = (|A| \times |S|)$, $|A|$ is the count of possible actions, $|S|$ is the count of internal agent states (i.e., official leaves), and t is the current time moment.

When selecting values of parameter δ, it is necessary to take into account its meaning – the agent tries to explore the states of the environment until the condition $c(s,a) \geqslant \delta\dfrac{t}{n}$ (or $\rho(s,a) \leqslant \dfrac{n}{\delta}$) holds for each state s and each action a. Too strong condition selection may destroy the learning process.

2.3. *Exploration Based on Expected Counter Value*

Additionally, for comparison purposes we consider one more counter-based exploration technique (Thrun, 1992a). McCallum has used (McCallum, 1997) only action-frequency statistics kept in the current state for action selection but Thrun has described (Thrun, 1992a) the way to use the expected counter value. The action count is considered in conjunction with Q-value to arrive at an exploration-adjusted utility value according to the following formula:

$$Eval(s,a) = \alpha Q(s,a) + \frac{c(s)}{E[c|s,a]}, \tag{5}$$

where $E[c|s,a]$ denotes the expected counter value of the state obtained by applying action a at state s, $c(s)$ counts how often state s occurred. The agent deterministically chooses the action with highest $Eval$.

3. Experimental Results

USM with different exploration techniques has been tested using ANIMAT problem (Wilson, 1985). It is a local perception grid world (see Fig. 2). The essence of this problem is searching for immovable goals in a maze.

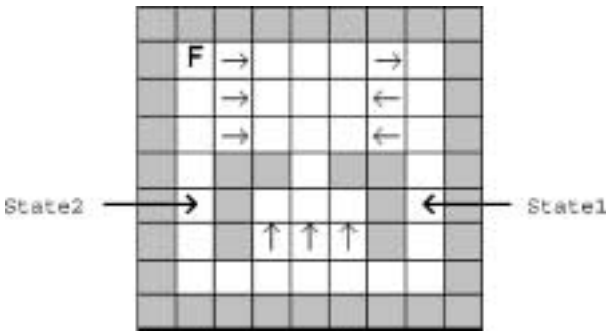

Fig. 2. `World1` (`state1` and `state2` are perceptually identical).

The agent's life consists of several cycles: it is placed in a random empty cell, after which the agent has to find the goal (marked "F") searched with the least possible number of steps. Initially the agent has no any knowledge on the environment. Each cycle can be treated as a task solved by the agent. In the course of cycle execution the agent has to learn to quickly find this object.

The agent can move only to nearest empty cells (eight possible directions mean eight possible actions that the agent can execute). If the agent tries to move onto barrier, it stays at the same position. This creates many cycles in the environment, and makes the learning task more difficult. The agent can perceive only the containment of nearest eight cells. So, there are different, but perceptually identical, world states.

Additionally, the cells can contain special symbols – arrows. These are normal empty cells, except the agent can move only in the direction defined by a corresponding arrow (in other case it stays in the same position).

The agent receives reward 100 upon reaching the goal (marked "F"). The agent used a temporal discount factor, $\gamma = 0.9$, and a fringe of depth 5. Other parameter settings are presented in Table 1. We have found experimentally that these settings are most appropriate.

Table 1

Parameters used for the simulation (α – constant gain weighting exploration versus exploitation; δ – constant defining state-action pairs exploration frequence; e – exploration probability)

Exploration kind			e	α	δ
Undirected exploration			0.1		
McCallum's exploration techniques	Statistics in states	Counter-based		10	
		Recency-based		4	
	Statistics in fringes	Counter-based		10	
		Recency-based		50	
Giving the internal reward for exploration		Counter-based		10	0.4
		Recency-based		2	0.2
Exploration based on expected counter value				10	

To compare different exploration techniques, we have selected two grid worlds `world1` (see Fig. 2) and `world2` (see Fig. 6). In `world1` the agent needs at average $9.8 * 10^8$ random steps to reach the goal from a typical cell. However, `world2` is more difficult. Firstly, it is much more complicated and larger, so, it is more difficult to learn it. Then in `world2` the agent needs at average $1.7 * 10^{10}$ random steps to reach the goal from a typical cell. And finally, `world1` has only 2 perceptually aliased states, but `world2` has 38 perceptually aliased states. So, it is much more difficult to orient in it.

3.1. *World1*

USM with eight different exploration techniques has been tested in `world1` (see Fig. 2). Fig. 3 shows the average number of steps (see "steps to goal") performed by the agent to reach the goal. The agent's life consists of cycles: it is placed in a random empty cell, after which the agent has to find the goal, so, "steps to goal" means the number of steps in one cycle. After each cycle, the value "steps to goal" is calculated as an average value using only last 10% of all trials. Another axis labeled "steps" means time measured in steps performed by the agent from the begging of experiment till the end. In the begging, the agent needs much more time to reach the goal state than in the end, and the number of "steps to goal" is converging to the theoretically optimal value. Additionally, Fig. 4 shows standard deviation of "steps to goal" that is converging to the zero value.

The undirected exploration technique or the techniques proposed by McCallum have shown the negative results. These results aren't presented because the agent hasn't reached the goal (the agent was able to do so only if it had been randomly placed near to the goal cell). All other used techniques have shown approximately the same performance, but Recency-based exploration has been less successful because the environment is static and the actions performed at the beginning of learning aren't less informative.

Why this simple grid world is so hard if the undirected exploration technique or the techniques proposed by McCallum are applied? Because, it is very hard to reach the goal in this world selecting random steps, for example, in our experiments the agent was unable to reach the goal even performing 10^6 random steps in `world1`.

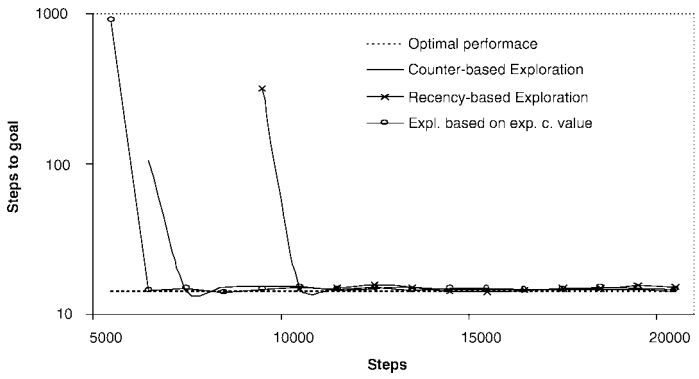

Fig. 3. The experimental results in `world1` (average of 10 runs).

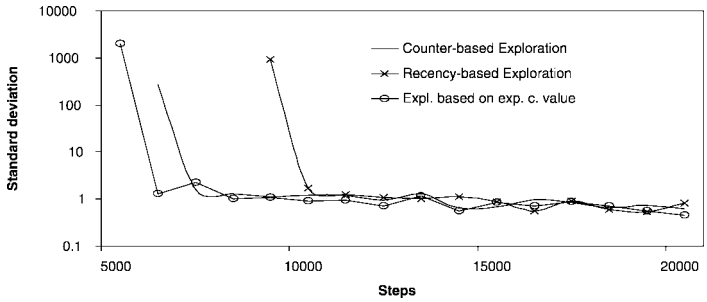

Fig. 4. The experimental results in `world1` (standard deviation of "steps to goal").

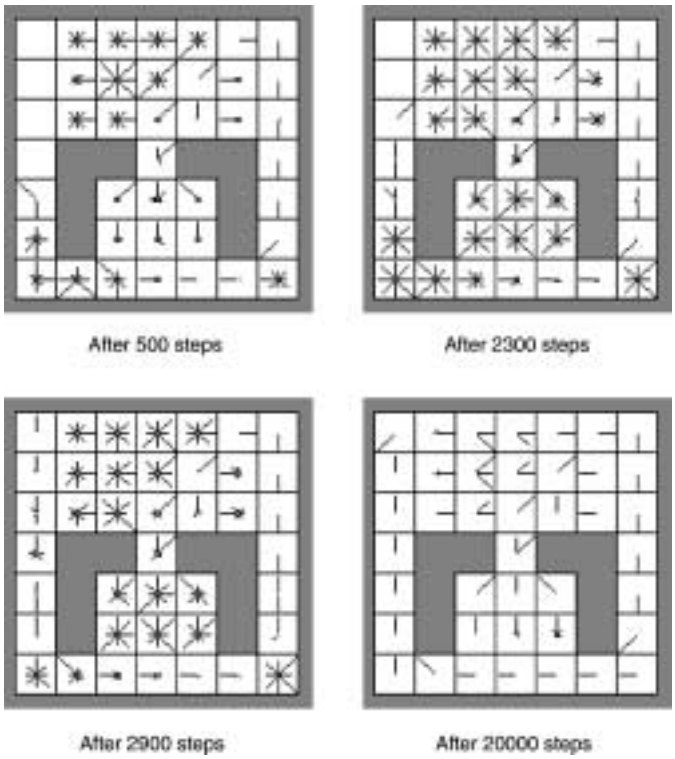

Fig. 5. The dynamics of the step trace in `world1`.

In contrast, Fig. 5 illustrates the dynamics of the step trace if the agent receives internal reward for exploration. The step trace is presented using a "vector-field" representation of the agent movements, which indicates the number of actions executed at a corresponding cell (the length of a vector is proportional to this number).

After 500 steps the agent hasn't discovered any perceptual distinction, thus, it is unable to distinguish between `state1` and `state2`. Then the agent makes multiple trials of reaching any particular state from any other particular state. `State1` and `state2` are

presented as one internal state in the agent memory because these states are perceptually identical. So, additional percepts (future back in time) can help to predict the success of achieving some particular state from this internal state. Thus, the agent discovers perceptual distinctions (it is not possible for original USM to discover distinctions before receiving external reward) (see after 2300 steps), and then after 600 (see after 2900 steps) steps the agent reaches the goal using discovered distinctions.

After 20000 steps the agent has developed optimal control policy that efficiently combines exploitation with exploration: the agent selects one action out of the most optimal actions.

3.2. *World2*

USM with most successful exploration techniques has been tested in a harder environment `world2` (see Fig. 6). The key difficulty is existence of many perceptual identical states that require different optimal actions. In our experiments the agent wasn't able to reach the goal even performing 10^9 random steps in `world2`.

Fig. 7 shows the average number of steps performed by the agent to reach the goal, but Fig. 8 shows standard deviation of it. Very successful in the previous case exploration based on the expected counter value has failed in `world2`, and the results are not presented because the agent hasn't reached the goal more than once during learning. Like original USM this exploration technique is not able to discover distinctions before receiving external reward. `World2` has so many perceptually identical states that it makes impossible to perform efficient exploration without discovering distinctions. But it can be improved giving internal reward for exploration.

Like in the previous case, the experimental results (see Fig. 7) show that the counter-based exploration is more successful. In contract to the previous case, only near to optimal

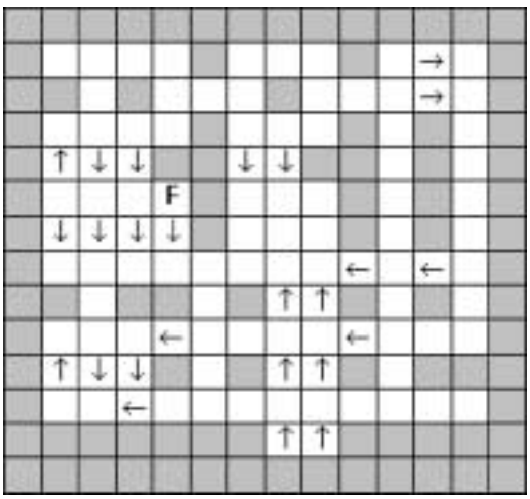

Fig. 6. `World2`.

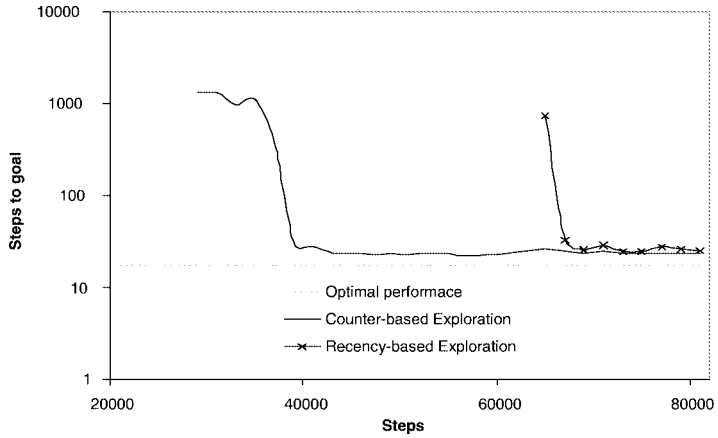

Fig. 7. The experimental results in `world2` (average of 10 runs).

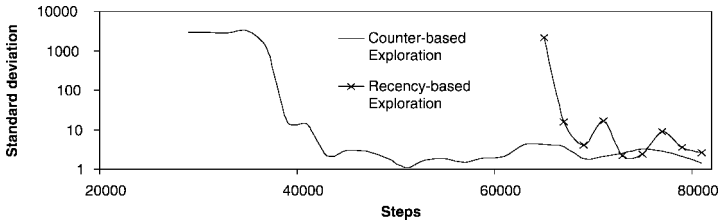

Fig. 8. The experimental results in `world2` (standard deviation of "steps to goal").

performance is achieved. At the same time the resulting step trace (see Fig. 9) (counter-based exploration is applied) is optimal enough. That is because the efficient combination of exploitation with exploration is harder achievable. In `world1` most of states that need exploration are situated on the main path to the goal but in `world2` there are many hard-to-reach states that require many additional steps to reach them.

3.3. *Distribution of the Internal Reward*

The agent assigns the internal reward for exploration to each disctinct state-action pair. To analyze the key features of the exploration techniques based on the internal reward for exploration, we can use the distribution of this reward over the state-action pairs (see Fig. 10). The presented distribution was computed at one particular time moment.

The form of the distribution consists of two parts: a small set of goals (on the right) and other state-acton pairs (on the left). This form makes possible for agent to concentrate on trying to achieve only hard-to-reach states – a small subset of all the internal states, and to avoid the trials to achieve conflicting goals at one time moment.

It is possible to setup the form of such distribution by selecting values of parameter δ. This value determine the exploration condition discussed before, so, too strong condition means too big set of goals, but too weak condition means too small set of goals.

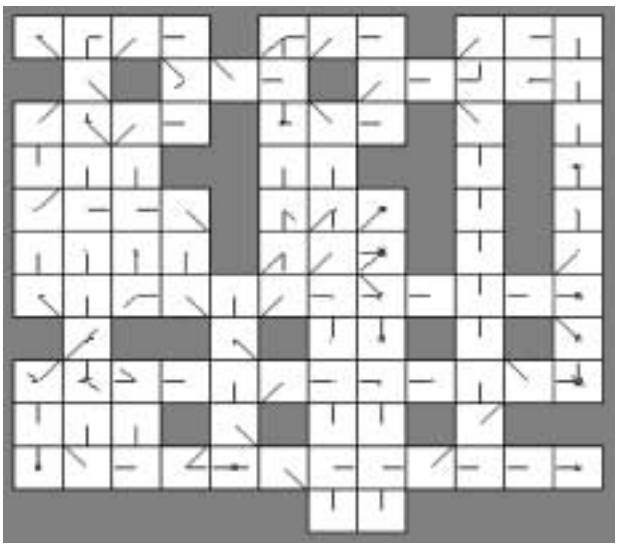

Fig. 9. The step trace in `world2` after 80000 steps.

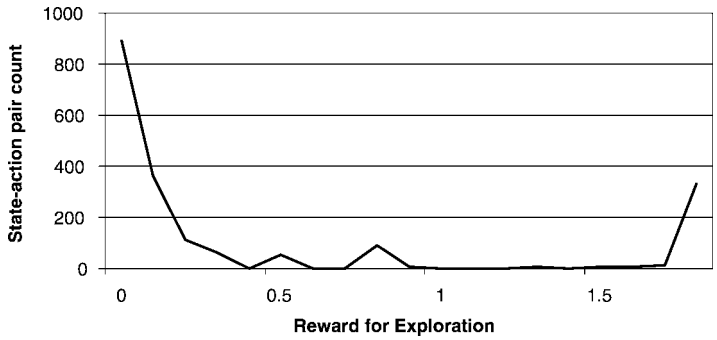

Fig. 10. The distribution of the internal reward for exploration over the state-action pairs (counter-based exploration, experimental results in `world2`).

Stronger exploration condition selection can speed up the learning process but may make it unstable.

4. Adaptive Fringe Depth

Potential leaf nodes are represented in USM in a fixed-depth fringe, an extension of the actual tree below the "official" leaf nodes. However, the use of a fringe of fixed depth incurs a large cost in terms of tree size and provides a limited degree of lookahead capability. The addition of a fringe of fixed depth adds exponentially to the size of the tree that must be stored or retained in memory. The same depth of fringe is maintained uniformly below the leaf nodes of a tree, including those leafs that provide an adequate basis for

prediction. So, to handle all these problems, the Greedy Util Suffix Memory (GUSM) algorithm was introduced (Breslow, 1995).

GUSM has the adaptive fringe depth. It uses a positive criterion for splitting a leaf node, similar to that used in USM, based on the immediate advantage of node splitting and resultant tree expansion. In addition, GUSM includes a negative split criterion, based on the *inadequacy* of a leaf node and the state it represents for determining the next correct action to take.

On the basis of this negative criterion, the agent can expand a tree below a leaf node of inadequate predictivity until it reaches a leaf node that is predictive. Rather than requiring a large fixed-depth fringe to handle hard problem, GUSM can handle all such problems with a fringe of depth 1.

The negative criterion used in GUSM is satisfied if a leaf node fails to offer a clear action recommendation. Specifically, it is satisfied if the recommended action in the node is not significantly superior to the action having the second-highest Q-value.

Experimental results in (Breslow, 1995) show that GUSM is able to solve problems on which USM will sometimes fail (i.e., depending on the setting of the fringe depth parametr).

In the previous section, the most efficient exploration technique – exploration based on internal reward was applied to USM that uses only fixed-depth fringe. That's why, we have applied GUSM ideas to USM with the exploration technique based on internal reward.

Fig. 11 shows the comparison of GUSM with USM in `world2`. In both cases, the agent uses exploration technique based on the internal reward.

In our experiments, we have not used the full GUSM implementation, instead, we have applied only the idea of the negative criterion for the fringe depth increasing. At each time, the agent has performed a simple test: if the Q-value of the best action differs

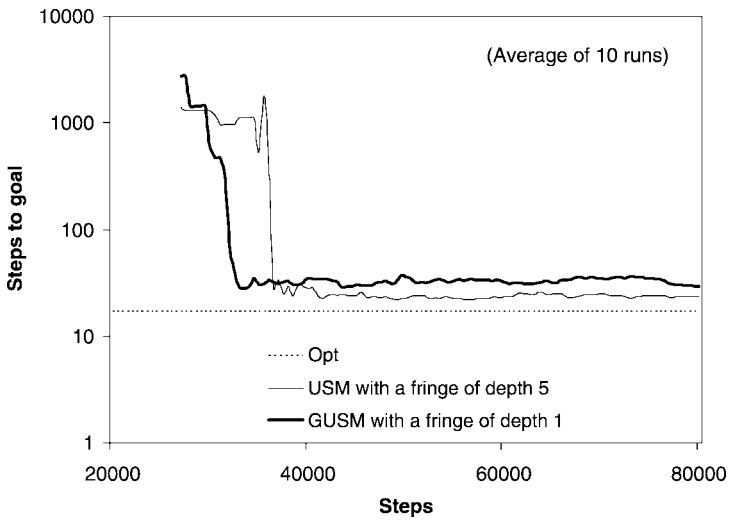

Fig. 11. The comparing of GUSM with USM in `world2`.

from the Q-value of the action having the second-highest value by value smaller than 10^{-7} then the agent increase the depth of the current fringe node by 1.

Additionally, the agent used a different constant gain weighting exploration versus exploitation, $\alpha = \frac{1}{3}$, instead of $\alpha = 10$, and a fringe of depth 1, instead of 5.

So, GUSM with a fringe depth 1 has shown approximately the same performance as USM with a fringe depth 5. That means that resulting learning algorithm was able not only to discover perceptual distinctions before receiving external reward from the environment but also to adapt its lookahead capability.

5. Conclusions

Eight different exploration techniques applied to the learning algorithm "Utile Suffix Memory" have been considered and experimentally tested in the paper. The exploration techniques that use internal reward for exploration have shown the best learning results.

The hypothesis that *the perceptual distinctions discovered during exploration will help the agent to reach the goal state* has been successfully confirmed in experimental way. In case of using internal reward for exploration the agent has been able to discover perceptual distinctions before receiving external reward from the environment while it is not possible using other exploration techniques.

In addition, the exploration technique has been successfully applied to Greedy Utile Suffix Memory, that provides adaptive fringe depth.

Experimental results in a partially observable maze show that exploration techniques have serious impact on performance of learning algorithm.

Acknowledgement. I thank Proffessor Arkady Borisov (Riga Technical University, Latvia) for helpful comments on my work.

References

Bellman, R.E. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ.

Breslow, L. (1995). *Greedy Utile Suffix Memory for Reinforcement Learning with Perceptually-Aliased States*. Technical Report AIC-96-004. DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence, Washington.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: the perceptual distinctions approach. In *Tenth National Conference on Artificial Intelligence*. pp. 183–188.

Kaelbling, L.P., L.M. Littman and A.W. Moore (1996). Reinforcement learning: a survey. *Journal of Artificial Intelligence Research*, **4**, 237–285.

Maes, P. (1994). Modeling adaptive autonomous agents. *Artificial Life*, **1**(1&2), 135–162.

McCallum, A. (1995). *Reinforcement Learning with Selective Perception and Hidden State* (Ph.D. dissertation). Department of Computer Science, University of Rochester, Rochester, NY.

McCallum, R.A. (1997). Efficient exploration in reinforcement learning with hidden state. In *AAAI Fall Symposium on "Model-directed Autonomous Systems"*.

Mitchel, T.H. (1999). *Machine Learning*. The McGraw-Hill Companies. Inc.

Ron, D., Y. Singer and N. Tishby (1994). Learning probabilistic automata with variable memory length. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*. pp. 35–46.

Thrun, S.B. (1992a). *Efficient Exploration in Reinforcement Learning*. Technical Report CMU-CS-92-102. Carnegie Mellon University, Pittsburgh, PA 15213.

Thrun, S.B. (1992b). The role of exploration in learning control. In DA White & DA Sofge (Eds.), *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. NY: Van Nostrand Reinhold, New York.
Sutton, R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*. pp. 216–224.
Sutton, R.S., and A.G. Barto (1998). *Reinforcement Learning: An Introduction*. MA: MIT Press, Cambridge.
Whitehead, S.D. (1991). Complexity and cooperation in Q-learning. In *Proceedings of the Eighth International Workshop on Machine Learning*. pp. 363–367.
Wilson, S.W. (1985). Knowledge growth in an artificial animal. In *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*. pp. 16–23.

A. Pchelkin has received his MSc degree in computer science at the University of Latvia in 2001. At present, he is PhD student at Riga Technical University. His research interests include artificial intelligence, reinforcement learning, neural networks, clustering, finite automata learning algorithms.

Sustiprinto mokymosi efektyvus tyrimas remiantis “Utile Suffix” atmintimi

Arthur PCHELKIN

Sustiprintas mokymasis nagrinėja kaip autonomiški agentai mokosi išsirinkti optimalius tikslo atžvilgiu veiksmus, t.y. agentai mokosi efektyviai tirti aplinką. Egzistuojantys darbai sustiprinto mokymosi srityje paprastai nagrinėja uždavinius su pilnai stebima aplinka. Čia nagrinėjamas uždavinys, kuriame aplinka stebima tik iš dalies. Pasiremdami USM (angl. “Utile Suffix Memory”) algoritmu, naudojančiu medžio tipo atminties struktūrą, nagrinėjame įvairius aplinkos tyrimo būdus, tame tarpe panaudojant adaptyvaus gylio medžius. Eksperimentiškai parodoma, kad aplinkos tyrimo būdo parinkimas iš dalies stebimame labirinte daro didelę įtaką mokymosi algoritmo atlikimui.