LOCAL GOALS DRIVEN HIERARCHICAL REINFORCEMENT LEARNING*

Arthur Pchelkin

Abstract: 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, the current paper, like the previous paper [Pch2003], studies the case when the environment is only partially observable. One additional difficulty is considered – complex temporal dependencies. In order to overcome this additional difficulty a new hierarchical reinforcement learning algorithm is proposed. The learning algorithm exploits a very simple learning principle, similar to Q-learning, except the lookup table has one more variable – the currently selected goal. Additionally, the algorithm uses the idea of internal reward for achieving hard-to-reach states [Pch2003]. The proposed learning algorithm is experimentally investigated in partially observable maze problems where it shows a robust ability to learn a good policy.

Keywords: reinforcement learning, hierarchical behaviour, efficient exploration, POMDPs, non-Markov, local goals, internal reward, subgoal learning.

Introduction and Problem Statement

One of the directions in artificial intelligence is adaptive autonomous agents' research. This research direction started growing actively in 1985 [Maes95,Wil85], however, it was proposed to make researches in similar directions also before it [Bong75]. Reinforcement learning [Sut98,KLM96,Mit99] examines a question how an autonomous agent [Maes95] that senses and acts in its environment can learn to choose optimal actions to achieve its goals. The agent can perceive a set of distinct perceptions from its environment and has a set of actions that it can perform. At each discrete time step, the agent senses the current percept, and chooses an action to perform it. The environment responds by producing the succeeding state and the agent can perceive a new observation. If the agent achieves the goal state, the environment gives the agent a reward.

The set of actions allowed to the agent is fixed and defined before learning. The structure of the environment is unknown to the agent and is represented by a black box. This means that it has to obtain all knowledge helping to achieve the agent's goals only by itself, only by experimenting with the environment. The task of the agent is to perform sequences of actions, observe their consequences, and to learn a control policy.

Irreversible transactions: Efficient exploration plays a fundamental role [Thr92b] 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 [Thr92a]. 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 [Whit91].

The reason for the difficult exploration by undirected techniques is the existence of irreversible transactions between states of the environment. Usually learning algorithms are being investigated in fully reversible domains [Sut98,KLM96,Mit99,McCallum95], *e.g.* the maze problem. In the maze problem each action has an opposite action, and the agent usually can easily undo its previous action by one step, *e.g.* the action "go left" can be undone by the action "go right". Because of this reason, each state of the environment becomes easily achievable. However, it could not be easily applied to the real case, *e.g.* if the goal of the agent is to build a house from blocks, one wrong random action may discharge all the previous work by destructing the built construct. That's why, the existence of irreversible transactions is not exclusion or the specifically invented difficulty but it is

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

a real difficulty that was unfortunately ignored in many investigations making simulations in the artificially designed environments.

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 [Wat89], can't be applied in partially observable domains. Due to this reason, McCallum has developed a learning algorithm "Utile Suffix Memory" (USM) [McCallum95] that is able to overcome the incomplete perception problem in order to learn a near-to-optimal policy in partially observable environments.

Efficient exploration in partially observable domains is a special difficulty [Chr92]. Previous approaches [Thr92a,Sut90] to exploration in reinforcement learning usually address exploration in the case when the environment is fully observable. In contrast, McCallum [McCallum97] considers efficient exploration applied to USM in the partially observable environment. In our study, we continued his research by applying to USM an exploration technique based on internal reward for hard-to-reach states [Pch2003], and our modification has outperformed the original algorithm in the case of difficult exploration.

Problem statement: Unfortunately, USM has one very important drawback – it is not able to make perceptual distinction by seeing too far future back in time. That's why, USM is not able to discover too complex temporal dependencies that include too many time steps. That was a motivation for the current paper to develop a reinforcement learning algorithm having an ability to overcome the combination of all the difficulties described above, *i.e.*:

- incomplete perception;
- irreversible transactions;
- complex temporal dependencies.

Automatic building of hierarchies: Complex temporal dependencies are usually solved by allowing the agent acting hierarchically [WS98,SS2000]. That's why, the present paper also considers hierarchical reinforcement learning as a key method for overcoming difficulties related with complex temporal dependencies.

There are many different models of hierarchical reinforcement learning [KLM96]. Considering different approaches of making hierarchical reinforcement learning, it is possible to distinguish two cases: (1) the use of structurally pre-determined domain-specific hierarchies and (2) automatic building of hierarchies. Most of hierarchical reinforcement learning algorithms are based on an assumption that fixed domain-specific knowledge about hierarchies is provided and can be exploited by the algorithm, *e.g.* [Die2000,PR97,Hum96] are only a small part of them. In contrast, in the current paper it is assumed that no prior domain-specific knowledge about subtask hierarchies is provided to the agent. Like [WS98,SS2000], the current problem statement is more difficult, but it is also more realistic.

Learning Algorithm

Training skills at local goals: In previous paper [Pch2003], it has been found that McCallum exploration techniques [McCallum95,McCallum97] may fail in the case when the environment isn't reversible, *e.g.* 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, the agent is unable to discover history distinctions and, thus, is unable to overcome incomplete perception problem. This problem has been solved [Pch2003] by giving the agent additional internal reward for state space exploration. Receiving additional internal reward for exploration USM was able to optimise its control policy not only for exploiting the environment but also for exploration in the same manner. It was relied 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. Simulation results in the maze domain had successfully confirmed the hypothesis.

The agent has the goal defined by its environment, let's call it the *global* goal (externally defined goal). However, any other state of the environment can also be considered as a goal, and it could be called a *local* goal (internally set goal). In the previous paper, while the agent was not able to reach the global goal, it was trying to reach some local goal to advance its skills of the environment control. Training skills at reaching local goals had helped the

agent to obtain skills sufficient for achieving the global goal. In the current paper, it is proposed to use the same idea of learning to reach local goals when the global goal is not achievable or it is hardly achievable.

Key ideas: The proposed learning algorithm is based on two ideas:

- hierarchical behaviour;
- training skills at local goals.

The agent needs hierarchical behaviour in order to overcome incomplete perception and complex temporal dependencies. Training skills at local goals are needed to make efficient exploration when the environment has irreversible transactions.

The learning algorithm has two parts. The first part selects the current main goal using three possible reasons: (1) a need to achieve the global goal, (2) a need to explore rarely observed perceptions and (3) a need to train skills at hard-to-reach goals. The task of the second part is choosing actions or selecting subgoals (like the calls of subroutines) in order to reach the current main goal. The selection of some subgoal can also be considered as abstract actions. In order to make the learning algorithm more simple, it is proposed to learn the primitive action selection policy and the abstract action selection policy in the same manner, applying the same principles.

Observations from the environment are used twice: (1) as the context information for action selecting and (2) as subgoals. In this sense, any main goal or any subgoal is a normal perception that has been observed but at the current moment the agent is trying to reach the state producing this observation. If the current observation is equal to the current goal, the goal is considered to be achieved. This means that the proposed learning algorithm is driven by local internal goals and their subgoals, and each goal or subgoal is a usual observation temporally considered in such role.

The agent also has the memory about successful cases and the agent adaptation rule could be described as follows: if in the context of observation p the selection of action a helps to achieve the current goal g (that can be a subgoal at the higher level), then next time the probability of the selection of the same action in the same context (observation p and the goal g) must be increased. This means that the learning algorithm doesn't exploit dynamic programming ideas about the estimation of distance to the goal, but it performs only pure reinforcement of successful actions.

To sum up, it should be noted that the agent does not only learn when to select what action, but also - when to select what subgoal. It means that the agent also must learn subgoal selecting policy.

Formal description of algorithm: The agent has the fixed set of actions A. The set of perceptions P is not directly given to the agent, instead, it is maintained all time and contains perceptions observed by the agent till the current time moment. Similarly, the counter $c(p)$ – the number of times the agent has observed percept p is maintained in order to provide directed exploration. Additionally, it is proposed to maintain the degree of difficulty $d[g]$ (initially equal to zero) for each local goal $g \in P$ that stores the total number of all failures minus the number of all successes at achieving goal g. Consequently, $D = \{ p \in P \mid d[p] > 0 \}$ can be considered as a set of difficult goals.

During learning process, the agent maintains its lookup table with real values $q[p,g,a]$, initially equal to 1, for each $p,g \in P$ and $a \in A \cup P$. This table doesn't store estimates of expected future discounted external rewards, *e.g.* as in Q-learning, instead, the value $q[p,g,a]$ stores the sum of all the internal rewards for performing action a or selecting a as a subgoal (if a is a perception) in the context of observation p and local goal g. The internal reward is not obtained from the environment, but the agent internally generates it for obtaining local goals or subgoals. To prevent the recursive calling of the same subgoal, it is proposed to define a set of the currently selected goals in the stack, noted by G.

It is also assumed that there is only one goal state in the environment, and the agent maintains a variable f storing initially *null*, or the observation of the goal state if the goal state has been achieved. The goal state can be recognized obtaining a positive reward from the environment.

The lookup table $q[p,g,a]$ is used for action selecting. However, this table is too big and it needs some kind of generalization on its values. For example, it may have an empty cell for some action a in the context of goal g and perception p, but at the same time it may have learned values for the same action in another context, and the last information also can be exploited in selecting of action a. For this purpose, we can define generalized value $Q(p,g,a)$ as follows:

$$Q(p,g,a) = \alpha_1 \cdot q[p,g,a] + \alpha_2 \cdot \frac{1}{|P|}\sum_{p \in P} q[p,g,a] + \alpha_3 \cdot \frac{1}{|P|^2}\sum_{p,g \in P} q[p,g,a]$$

Other notations: *t* – the current time moment, *random* – random value in interval [0;1).

The proposed hierarchical reinforcement learning algorithm can be described as follows:

```
Main:
  G = ∅
  repeat
    ExecuteAction(RandomAction); TryToReachGoal(GetMainGoal,λ)

GetMainGoal:
  if f ≠ null and (D = ∅ or random ≥ δ)  then result = f
  else if D ≠ ∅ and random < ½ then
         result = select g∈D with probability Pr(g) = 1
       else result = select g∈P with probability Pr(g) = 1/c(g)^η

TryToReachGoal(g,s):
  G = G∪{g} ; t0 = t ;  i = 0 ; E = ∅
  while CurrentObservation ≠ g and i < τ
    i = i+1
    p[i] = CurrentObservation
    V = { x∈A∪P | s ≥ 1/Q(p[i],g,x) & x∉G∪E∪{p[i]} }
    if V = ∅ then V = A
    a[i] = select x∈V with probability Pr(x) = Q(p[i],g,x)^β
    if a[i] ∈ A then ExecuteAction(a[i]) ; r[i] = SUCCESS
    else r[i] = TryToReachGoal(a[i], s - 1/Q(p[i],g,a[i]))
    if r[i] ≠ SUCCESS then E = E∪{a[i]}
  if CurrentObservation = g then
    while r[i] = SUCCES and i > 0 do
      q[p[i],g,a[i]] = q[p[i],g,a[i]] + 1/(t-t0)^γ
      i = i-1
  G = G -{g}
  if CurrentObservation = g then d[g] = d[g]-1 else d[g] = d[g]+1
  if CurrentObservation = g then result=SUCCESS else result=FAILURE
```

Notation *result* means the resulting value of a function, and notations t_0, *i*, *E*, *p*[*i*], *a*[*i*], *r*[*i*], *V* are local variables of function "TryToReachGoal". The algorithm has a series of parameters: α_1, α_2, α_3, β, η, δ, γ, λ, τ, and it is proposed to use the following settings: α_1=1, α_2=0.01, α_3=0.0001, β=5, η=5, δ=0.7, γ=0.1, λ=3, τ=4.

Simulation Results

The presented above learning algorithm has been tested using three different maze problems: maze1, maze2 and maze3 (see Fig.1). Each maze is a local perception grid world. The essence of this problem is searching for immovable goals in a maze. The agent's life consists of many trials: it is placed in a random empty cell, after which the agent has to find the goal (marked "G") searched with the least possible number of steps. Initially the agent has not any knowledge on the environment. Each trial can be considered as one problem solved by the agent. In the course of trials, the agent has to learn to quickly find this object.

The agent can move to nearest empty cells only (white or silver cells, but not black ones; some cells are specially highlighted with silver – it means that these cells have duplicated observations). 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). These arrows are needed to simulate discussed above irreversible transactions between the environment states.

To simulate complex temporal dependencies, there are presented two special cells: a door (marked "D") and a key (marked "K"). To be able to come into the cell with a door, the agent needs to visit the cell with a key before. After visiting the cell with a key, the agent is able to come into the cell with a door only once. If the agent is not able to come into the door, it stays at the same cell. The idea about the door and the key has been taken from the paper [WS98].

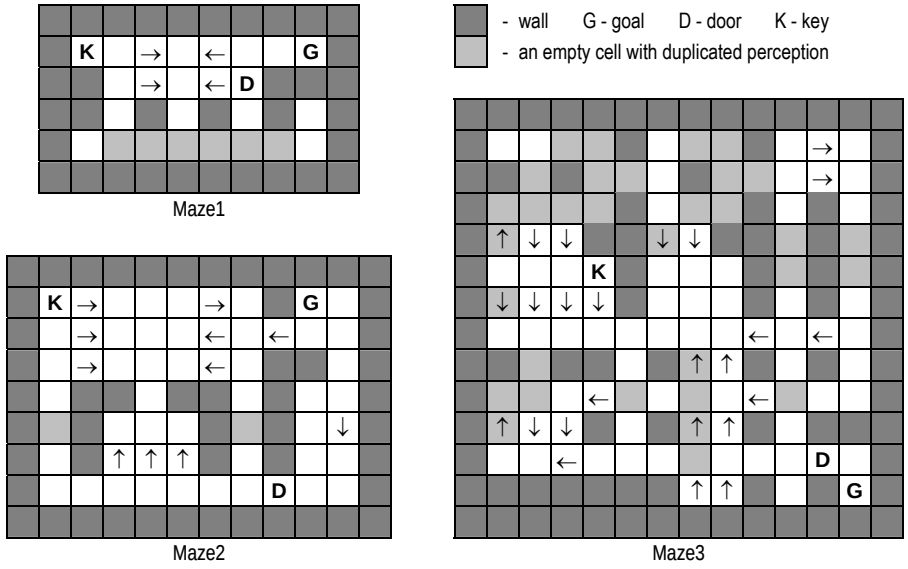

Figure 1. Three different maze problems: maze1, maze2 and maze3 for experiments

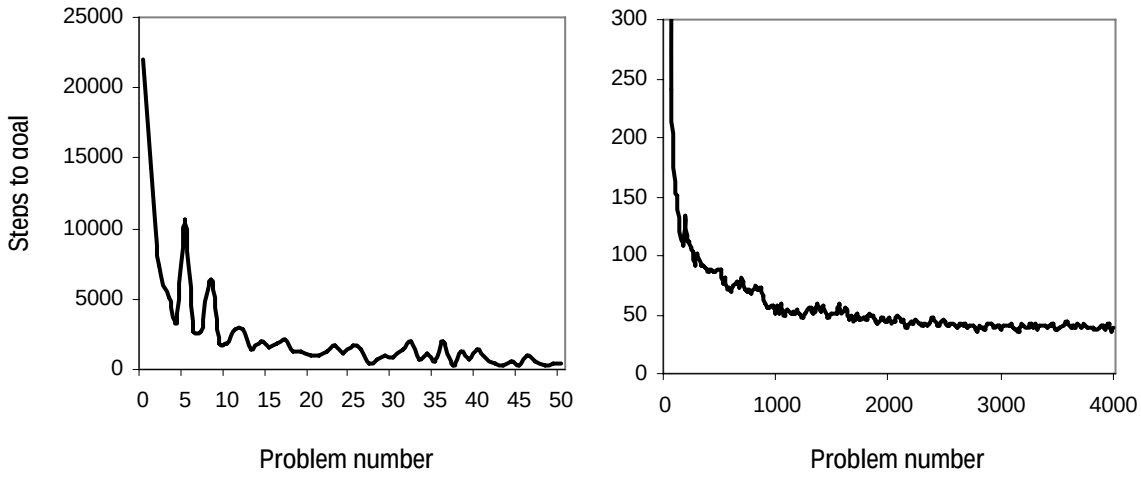

Figure 2. Experimental results in maze1

Results: Figures 2,3 and 4 show the dynamics of the number (see “steps to goals”) of steps performed by the agent to reach the goal (a cell marked “G”) in each trial during the learning process. Abscise “problem number” means the index of a goal searching trial. All experiments were repeated 10 times, and there are presented only averaged results.

Each simulation is represented by two graphs. The graph on the left shows the convergence of the number of steps to the goal in the beginning. Here, the learning algorithm shows its ability to roughly sketch out non-optimal but successful behavior. However, after this sketching the optimization of a policy continues very slowly. The second graph on the right shows this slow optimization. The same tendency could be noted on other two graphs.

Table 1 shows summary of all the experiments. It also contains additional information describing the selected maze problems. It should be noted that the number of random steps to the goal from a typical state is extremely large in maze2 and maze3. It is because of irreversible transactions between states in these environments.

The proposed learning algorithm shows very stable ability to form a good policy in each case. However, the experiments have also discovered the drawbacks of the algorithm: slow adaptation and non-optimality of resulting policy. The agent was able to form only a good policy, but not a theoretically optimal one.

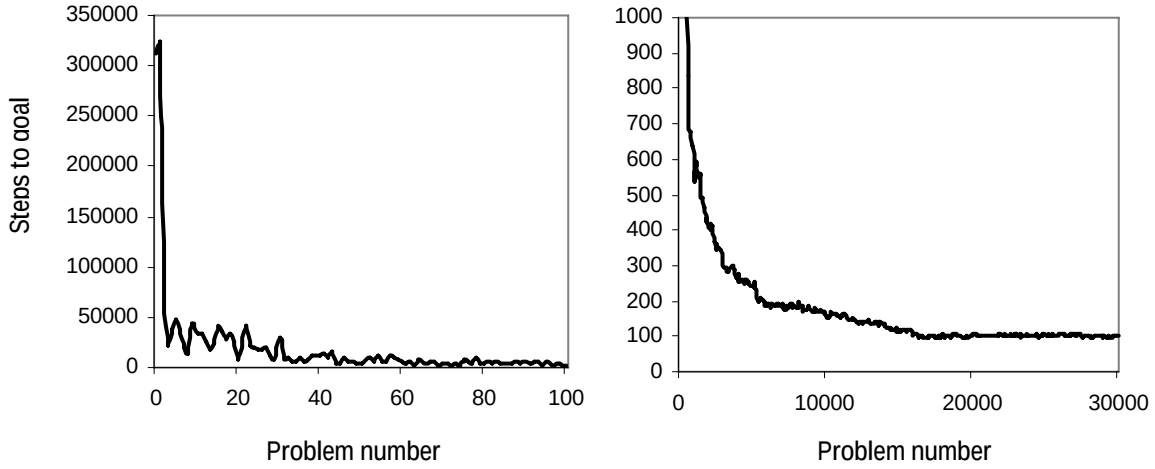

Figure 3. Experimental results in maze2

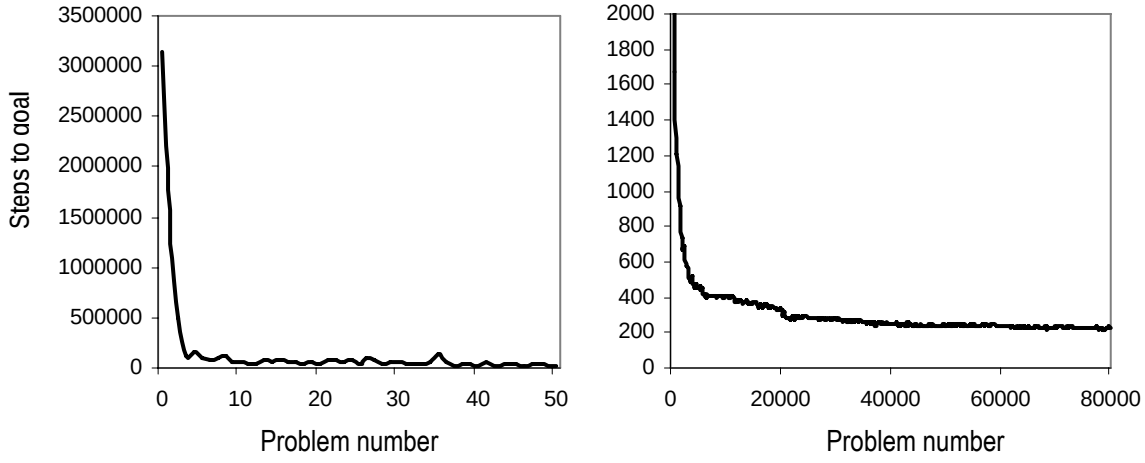

Figure 4. Experimental results in maze3

Table 1. Results summary

		Maze1	Maze2	Maze3
Environment description	The number of perceptions	21	55	89
	The number of positions in the maze	25	56	
	The number of positions in the maze with not unique observations	6	2	32
	The number of states	45	97	208
	The number of belief states	51	98	225
	The number of random steps to the goal from a typical state	$4.1 \cdot 10^4$	$2.3 \cdot 10^9$	$4.3 \cdot 10^{10}$
	The theoretically optimal number of steps to the goal	12	25	29
Experimental results (on average)	The resulting average steps to the goal	39	97	234
	The time of the first goal achievement	$2.2 \cdot 10^4$	$3 \cdot 10^5$	$3.1 \cdot 10^6$
	The total number of solved problems	2^*10^3	$1.7 \cdot 10^4$	$2 \cdot 10^4$
	The total learning time needed to converge to a good policy	$2.6 \cdot 10^5$	$6.4 \cdot 10^6$	$2 \cdot 10^7$
	During learning the number of steps to the goal was reduced X times	$5.6 \cdot 10^2$	$3 \cdot 10^3$	$1.3 \cdot 10^4$
	The resulting number of steps to the goal is X times bigger than the optimal one	3.25	3.88	8.07

Comparison to Other Algorithms

Developing a new algorithm, it is important to analyze its place in the context of other algorithms serving for similar purposes. For this aim, it was decided to compare the proposed learning algorithms with other reinforcement learning algorithms. In order to make such comparison, several criteria were proposed in Table 1. These criteria were used for the comparison presented in Table 2.

It should be noted that the presented evaluation of algorithms is very rough and could not be considered as a fully proved comparison of different algorithms. However, it can help to describe features of the developed algorithm. The advantages of the proposed algorithm are the simple implementation and its stable ability to form a good policy in the extremely complex case when the environment has irreversible transactions and complex temporal dependencies. However, it has also drawbacks: non-optimality and slow adaptation.

Table 2. The description of criteria

Criterion	Description
SI	The implementation is simple.
CT	The computational time taken at each time step is small.
LT	The ability to learn the policy fast.
OPT	The resulting policy is very close to the optimal one.
PO	The ability to overcome the incomplete perception in partly observable environments.
EF	The ability to perform efficient exploration when the environment has many irreversible transactions.
CD	The ability to learn complex temporal dependencies.
PD	The ability to discover perceptual distinctions.
SA	The architecture is not comprehensive or composite.
UL	The number of hierarchy levels is not limited.

Table 3. Comparison to other algorithms

Learning algorithm	SI	CT	LT	OPT	PO	EF	CD	PD	SA	UL
Q-learning [Wat89]	+	+	+/-	+	-	-	-	-	+	-
SSS algorithm [SS2000]	+/-	+	-	+/-	+	-	+/-	-	-	+
HQ-learning [WS98]	+/-	+	-	+/-	+	-	+	-	-	-
USM algorithm [McCallum95]	-	-	+	+	+	-	-	+	+	-
USM + "internal reward for hard-to-reach states" [Pch2003]	-	-	+	+/-	+	+	-	+	+	-
The proposed algorithm	+	+	-	-	+	+	+	-	+	+

Conclusion

In this paper a new hierarchical reinforcement learning algorithm was presented that doesn't exploit any domain-specific knowledge about subtask hierarchy, but automatically builds useful hierarchies. The algorithm was developed with purpose to overcome the combination of three, previously known in reinforcement learning, difficulties: (1) incomplete perception, (2) irreversible transactions and (3) complex temporal dependencies. The key idea of the algorithm is to exploit the observation from the environment not only as context information for action selecting, but also as local, internally selected, goals and subgoals. This makes the agent to be hierarchical reinforcement learner, driven by local goals, that has a native ability of efficient exploration.

The proposed learning algorithm was experimentally investigated in different and very complex maze problems, showing very stable ability to form a good policy in each case. However, the experiments have also discovered the drawbacks of the algorithm: slow adaptation and non-optimality of resulting policy (the agent was able to form only a good policy, but not a theoretically optimal one).

Bibliography

[Bong75] Бонгард, М.М., Лосев, И.С., Смирнов М.С. Проект модели организации поведения – Животное // Моделирование обучения и поведения. – М.: Наука, 1975.

[Chr92] Chrisman, L. Reinforcement Learning with Perceptual Aliasing: The Perceptual Distinctions Approach, In Tenth National Conference on Artificial Intelligence, 1992, pp.183-188.

[Die2000] Dietterich, T.G. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, Journal of Artificial Intelligence Research, 13, pp.227-303, 2000.

[KLM96] Kaelbling, L.P., Littman, L.M., Moore, A.W. Reinforcement learning: a survey, Journal of Artificial Intelligence Research, Vol. 4, 1996, pp.237-285.

[Hum96] Humphrys, M. W-learning: a simple RL-based society of mind. Technical report 362, University of Cambridge, Computer Laboratory, 1996.

[Maes95] Maes P., Modeling Adaptive Autonomous Agents, Artificial Life: An overview, MIT Press, 1995.

[McCallum95] McCallum, R.A., Reinforcement learning with selective perception and hidden state (Ph.D. dissertation). Department of Computer Science, University of Rochester, Rochester, NY, 1995.

[McCallum97] McCallum, R.A., "Efficient Exploration in Reinforcement Learning with Hidden State", AAAI Fall Symposium on "Model-directed Autonomous Systems", 1997.

[Mit99] Mitchel, T.H., Machine learning, The McGraw-Hill Companies. Inc., 1999.

[Pch2003] Pchelkin A. Efficient exploration in reinforcement learning based on Utile Suffix Memory, journal "Informatica", Lithuanian Academy of Sciences, Vol.14., 2003.

[PR97] Parr, R., Russul, S. Reinforcement learning with hierarchies of machines. Advances in Neural Information Processing Systems 9. MIT Press, Cambridge, MA.

[Sut98] Sutton, R.S., Barto, A.G. Reinforcement learning: An introduction. Cambridge, MA: MIT Press, 1998.

[SS2000] Sun, R., Sessions, C. Self-segmentation of sequences: automatic formation of hierarchies of sequential behaviors. IEEE Transactions on Systems, Man, and Cybernetics: Part B Cybernetics, 30(3), 2000.

[Thr92a] Thrun, S.B., Efficient exploration in reinforcement learning, Technical Report CMU-CS-92-102, Carnegie Mellon University, Pittsburgh, PA 15213, 1992.

[Thr92b] Thrun, S.B., The role of exploration in learning control. In DA White & DA Sofge, editors, Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches. New York, NY: Van Nostrand Reinhold, 1992.

[Wat89] Watkins, C. Learning with Delayed Rewards. Ph.D Thesis, Cambrige University, Cambrige, UK.

[Whit91] Whitehead, S.D. Complexity and cooperation in Q-learning, In Proceedings of the Eighth International Workshop on Machine Learning, 1991, pp.363-367.

[Wil85] Wilson S.W. Knowledge growth in an artificial animal, Proceeding of the First International Conference on Genetic Algoritms and Their Applications. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1985, pp.16-23.

[WS98] Wiering, M., Schmidhuber, J. HQ-learning, Adaptive Behavior, vol. 6.2, pp. 219-246, 1998.

Author Information

Arthur Pchelkin – PhD. Student; Institute of Information Technology, Technical University of Riga; 1, Kalku St., Riga LV-1658, Latvia; e-mail: arturp@inbox.lv