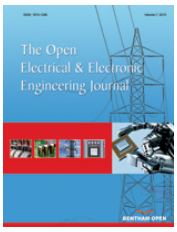
Send Orders for Reprints to reprints@benthamscience.ae

The Open Electrical & Electronic Engineering Journal, 2016, *10*, 69-79

The Open Electrical & Electronic Engineering Journal

Content list available at: www.benthamopen.com/TOEEJ/

DOI: 10.2174/1874129001610010069



RESEARCH ARTICLE

Hierarchical Reinforcement Learning Based Self-balancing Algorithm for Two-wheeled Robots

Juan Yan* and Huibin Yang

College of Mechanical Engineering, Shanghai University of Engineering Science, Shanghai, China

Received: February 15, 2016 Revised: May 10, 2016 Accepted: May 15, 2016

Abstract: Self-balancing control is the basis for applications of two-wheeled robots. In order to improve the self-balancing of two-wheeled robots, we propose a hierarchical reinforcement learning algorithm for controlling the balance of two-wheeled robots. After describing the subgoals of hierarchical reinforcement learning, we extract features for subgoals, define a feature value vector and its corresponding weight vector, and propose a reward function with additional subgoal reward function. Finally, we give a hierarchical reinforcement learning algorithm for finding the optimal strategy. Simulation experiments show that, the proposed algorithm is more effectiveness than traditional reinforcement learning algorithm in convergent speed. So in our system, the robots can get self-balanced very quickly.

Keywords: Balancing control, Hierarchical reinforcement learning, Q-learning, Robot.

1. INTRODUCTION

The two-wheeled self-balancing robot [1] is an important research topic in intelligent developmental robots. Comparing with traditional robots, the intelligence of this kind of robots can develop dynamically with external environments, and the intelligence comes from an inner system similar to human brain [2]. The self-balancing of two-wheeled self-balancing robot is controlled by its inner development mechanism, and is reinforced by the intelligence according to communications with external environment by sensors and executors [3].

Aiming at the self-balancing of two-wheeled robot, researches have proposed a lot of control approaches. Wu *et al*. [4] implement the control of an inverted Pendulum according to Q-learning [5] and back propagation neural networks [6]; Jeong and Takahashi [7] design a balancing controller based on the problem of two-level inverted pendulum, and they apply the linear quadratic regulator algorithm; Miasa *et al*. [8] apply a hierarchical fuzzy control model to acquire the self-balancing. The above algorithms for self-balancing in two-wheeled robots are all based on neural networks, and they have the advantage of high fault-tolerance. However, their disadvantages are weak learning ability and also sensitive to external noise, so the controller is hard to reach a stable status.

Reinforcement learning is a unsupervised learning approach, and is widely used in real time controlling [9]. In reinforcement learning, the controller communicates with external environment according to trial-and-error. Hierarchical reinforcement learning [10] is one of the most important researches in reinforcement learning, and it can solve the problem of dimensionality curse in status and action spaces. Classical hierarchical reinforcement learning approaches include Option [11], MAXQ [12], HAM [13].

In this paper, we study how the two-wheeled robots can get self-balanced very quickly. Based on traditional reinforcement learning algorithm, we propose a hierarchical reinforcement learning algorithm. In our hierarchical reinforcement learning algorithm, in order to speed up the convergence, we add a heuristic reward function in each

* Address correspondence to this author at the College of Mechanical Engineering, Shanghai University of Engineering Science, Shanghai, China; E-mail: yanjuanxz@sina.com

subgoals.

The rest of the paper is organized as follows. In section 2, we review related works about reinforcement learning especially on hierarchical reinforcement learning. In section 3, we describe our proposed hierarchical reinforcement learning approach. Experiments and conclusion are given in section 4 and 5 respectively.

2. RELATED WORKS

In this section, we describe the definition of reinforcement learning, and review related works on reinforcement learning, especially on hierarchical reinforcement learning.

Reinforcement learning is a unsupervised learning, and it is based on the idea that, if some action of an agent can get positive reward from environment, then the trend of this action will be reinforced in the future. The basic model of reinforcement learning can be described in Fig. (**1**).

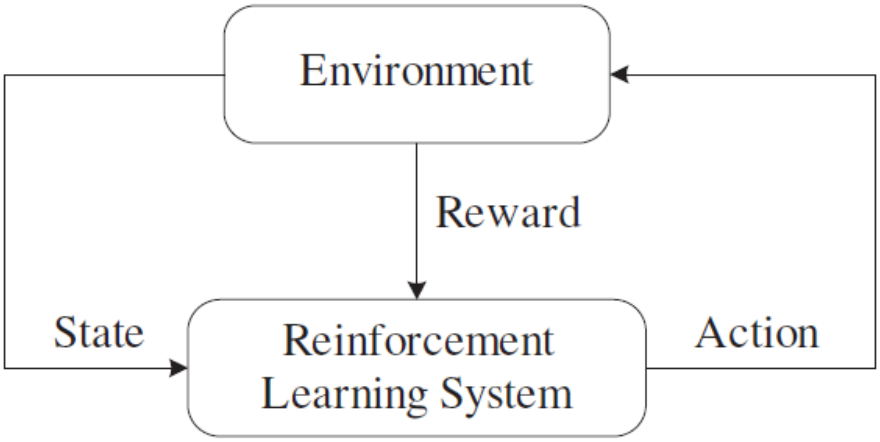


Fig. (1). Model of reinforcement learning.

The process of reinforcement learning can be described as a Markov decision process, and the Markov decision process can be defined by a quadri-tuple $< S,A,R,P >$, where S is the set of statuses, A is the set of actions,$R : S \times A \rightarrow R$ is the reward function, and $R : S \times A \rightarrow S$ is the status transition function. In Markov decision process, the reward and next status only depend on the current status and the selected action, and the objective of an agent is to find the optimal action strategy and maximize the expected discount reward.

In order to find optimal action strategy, researchers introduce the concept of quality function. Currently, the commonly used quality function is the Q function, *i.e.* Q-learning. In Q-learning [5], the reward is based on status-action pair, and the update rule is

$$\begin{aligned} V(s) &= \max_a Q(s,a), \\ Q_{t+1}(s,a) &= (1-c)Q_t(s,a) + c(r + \gamma V(s')) \end{aligned} \tag{1}$$

where C is the learning rate, γ is the discount factor, r is the current reward, and s' is the next status.

Traditional reinforcement learning approaches only improve their strategies via experienced rewards, and need very long time to train, so the convergence speed is very slow. In order to speed up the convergence, Singer and Veloso [14] propose to solve the new problem via inducing the local features of original problem; Hailu and Sommer [15] discuss the effects of different bias information on learning speed by introducing environment information. Moreno *et al*. [16] propose to to introduce prior in supervised reinforcement learning; Lin and Li [17] build a reinforcement learning model based on latent bias; and Fernández and Veloso [18] reuse past learnt bias to supervise the solving of similar tasks. The above approaches use bias to supervise the selection of strategies from actions, can utilize the bias from external environment or past tasks, and thus the learning speed is accelerated. However, whenever the bias is unsuitable, it can mislead the agent, and then the learning would be not convergent.

Hierarchical reinforcement learning, *e.g*. Option [11], MAXQ [12], HAM [13], can speed up the convergence of learning efficiently, and is widely used in many fields. Many extended approaches are based on Option, MAXQ and HAM. Mehta *et al*. [19] propose to denote the reward function as vector of status features, and use the learnt experience to reward functions (vectors) of the same hierarchies; Schultink *et al*. [20] propose an economic hierarchical q-Learning algorithm, which can converge to hierarchical optimums theoretically. Moreover, in finding subgoals in hierarchical reinforcement learning, Stolle and Precup [21] assume the frequently accessed points be subgoals; Mannor *et al*. [22] argue that subgoals should be the points accessed frequently by valid pathes; Şimşek *et al*. [23] find subgoals using recent status transition graph according to graph partition; and Wixson [24] applies modularity to find subgoals such that the algorithm can be scaled to large scale reinforcement learning.

3. HIERARCHICAL REINFORCEMENT LEARNING

In this paper, we propose a hierarchical reinforcement learning algorithm based on heuristic reward function. With this approach, the two-wheeled robot can learn from environment, and thus keep self-balancing as far as possible.

3.1. Description of Subgoals

Description of subgoals is the hardest part in hierarchical reinforcement learning. In this paper, we describe each subgoal with an uncertain limited status controller, which is proposed by Dietterich [12].

Given an objective or goal M, we partition M into a subgoal set $\{M, M_1,...,M_n\}$, and each $M_i(0 \le i \le n)$ is denoted as a triple $< T_i, A_i, R_i >$, where T_i is the status set in subgoal M_i, A_i is the executable action set in M_i, and R_i is its corresponding reward function in M_i. In this paper, we aim to extend the reward function R_i of subgoal M_i, and then accelerate the convergence speed. After extending reward function R_i of M_i to R_i', we get a new triple $< T_i, A_i, R_i' >$. While Learning all subgoals, the strategy π is partitioned into $\{\pi, \pi_1,...,\pi_n\}$, where π_i is the strategy of subgoal M_i, and all π_is are independent with each other. Then, all optimal strategy of subgoals will make up the final optimal strategy of the goal M, *i.e*. $\pi^* = \{\pi^*, \pi^*_1,...,\pi^*_n\}$.

3.2. Subgoal Feature Extraction and Reward Function

In order to utilize heuristic reward function, we extract features from subgoals. We use the extracted features to evaluate the current status, and then design corresponding reward function F. For each subgoal M_i in M, we give the following definitions.

Definition 1. M_i is the i-th subgoal of M, then the feature vector T_i of subgoal M_i is

$$T_i = \{t_{i1}, t_{i2}, \cdots, t_{ik}\} \tag{2}$$

where k is the number of features, and t_{ij} $(1 \le j \le k)$ is the j-th status feature.

T_i is a vector description of status features. For example, extracting features from a polygon can get a feature vector

T_i is an abstraction of status features, and it can't be used in the learning system directly. In order to apply status features, we need to quantify each element in the feature vector.

Definition 2. *Feature vector T_i can be quantified to be a feature value vector N_i, which is*

$$N_i = \{n_{i1}, n_{i2}, \cdots, n_{ik}\} \tag{3}$$

where n_{ij} is the feature value corresponding to feature t_{ij} in subgoal M_i.

In a reinforcement learning system, each feature belongs to a certain value scope, and this value scope is determined by concreting applications. In addition, each feature has different effectiveness on learning process, so each feature must have its own weight.

Definition 3. The weight vector W_i of the feature value vector N_i is

$$W_i = \{w_{i1}, w_{i2}, \cdots, w_{ik}\} \tag{4}$$

where w_{ij} is the weight of the j-th feature value n_{ij}.

In W_i, the value of its element w_{ij} can be positive, zero or even negative. When $w_{ij} > 0$, this feature is encouraged in learning, when $w_{ij} = 0$, this feature can be ignored, and when $w_{ij} < 0$, this feature should be penalized. Based on the above three definitions, we define an additional function for the heuristic reward function.

Definition 4. The additional function F_i for the heuristic reward function is

$$F_i = W_i^T N_i = w_{i1} n_{i1} + w_{i2} n_{i2} + \cdots + w_{ik} n_{ik} \tag{5}$$

where W_i^T is the transpose of W_i.

In equation 5, the value of F_i can also be positive, zero or even negative. When $F_i > 0$, the environment gives some reward to the heuristic function, and when $F_i < 0$, the environment punishes the heuristic function. Finally, the reward function for a generalized Markov Decision Process is defined as follows.

Definition 5. The reward function for a generalized Markov Decision Process is

$$R_i^{'} = R_i + F_i = R_i + w_{i1} n_{i1} + w_{i2} n_{i2} + \cdots + w_{ik} n_{ik} \tag{6}$$

where R_i is the reward function in a standard reinforcement learning.

In order to make the learning algorithm converge, R_i must be bounded. In addition, because the elements in W_i and N_i are constants and the number of status features is limited, *i.e.* $k < \infty$, F_i is also bounded. So, $F_i^{'}$ is also a bounded reward function, which is the basis for an learning algorithm to converge.

3.2.1. Optimal Strategy

The goal of reinforcement learning is to find an optimal strategy, and this can be done by continuous iterations. Let $V^{\pi i}(i, s)$ be the value of status s in strategy π_i in subgoal M_i, then the Bellman equation is

$$V^{\pi_i}(i,s) = V^{\pi_i}(\pi_i(s), s) + \sum_{s'} \gamma V^{\pi_i}(i,s') \cdot P_i^{\pi_i}(s \to s' \mid s, \pi_i(s)). \tag{7}$$

By continuous iterations, we can get the optimal function of the i-th subgoal under the optimal strategy π^*. That is

$$\begin{aligned} &V*(i,s) = \max_{a \in A(s)} E\{R_{t+1} + \gamma V*(i, s_{t+1}) \mid s_t = s, a_t = a\} = \\ &\max_{a \in A(s)} \sum_{s'} P_{ss'}^a ((R^{i'})_{ss'}^a + \gamma V*(i,s')), \end{aligned} \tag{8}$$

where $P^{a}{}_{ss'}$ and $(R^{i'})^{a}{}_{ss'}$ are the transition probability and heuristic reward value from status s to status s' while executing action a, and γ is the discount factor.

In equation 8, we present the optimal Bellman equation with status-value function In order to implement an online control algorithm, the Bellman equation is usually represented with an action-value function. In subgoal M_i, the current status is s, if we act with action a according to strategy $\boldsymbol{\pi_i}$, then the acquired Q value is

$$\begin{aligned} &Q^{\pi_i}(i,s,a) = E_{\pi_i}\{R_t' \mid s_t = s, a_t = a\} = E_{\pi_i}\{\sum_{k=0}^{\infty} \lambda^k (R'_{t+k+1}) \mid s_t = s, a_t = a\} = \\ &\sum_{s'} P_{ss'}^a + \gamma \max_{a'} Q(i, s', a)), \end{aligned} \tag{9}$$

where λ is the parameter of step number, γ is the discount factor. Then, the corresponding action-value function Bellman equation is

$$Q^*(i,s,a) = E\{R' + \gamma \max_{a'} Q^*(i,s',a') \mid s_t = s, a_t = a\} = \sum_{s'} P_{ss'}^{a}(R'_{ss'} + \gamma \max_{a' \in A(s)} Q^*(i,s',a')) = Q(i,s,a) + \alpha(R' + \gamma \max Q(i,s,a)). \quad (10)$$

In order to assure the algorithm be convergent, it only needs to select the actions greedily. So during the execution of algorithm, we use the follow equation to update the $\boldsymbol{Q}$ value.

$$Q(i,s,a) = Q(i,s,a) + \alpha(R' + \gamma \max_{a \in A(s)} Q(i,s',a') - Q(i,s,a)). \quad (11)$$

3.3. Hierarchical Reinforcement Learning Algorithm

Algorithm 1: HRLearning algorithm

Partition M into $\{M, M_1, \ldots, M_n\}$,such that for each $i(0 \le i \le n)$, $M_i = <T_i, A_i, R_i>$
Randomly initialize $\pi = \{\pi, \pi_1, \ldots, \pi_n\}$
For each M_i in M do
    repeat
        Extract features and get T_i and N_i ;
        Randomly initialize W_i ;
        Let $s \leftarrow s$;
        repeat
            Select an action a for s from strategy of Q
            Execute action a computer r, s' and $R' = r + F'$
            Let $Q(i, s, a) \leftarrow Q(i, s', a') + \alpha(R' + \gamma \max_{a \epsilon A(s)})$
    Until $s \in T_i$;
Until M_i is finished;

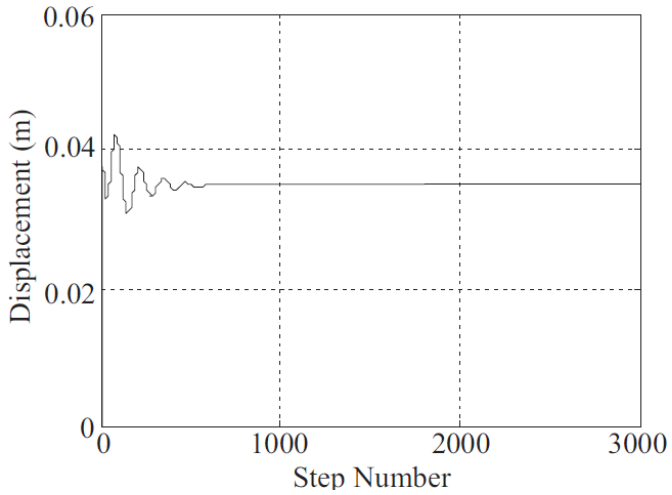
Partition **[Trial mode]** into **[Trial mode]**, such that for each **[Trial mode]** (**[Trial mode]**), **[Trial mode][Trial mode][Trial mode][Trial mode]**M**[Trial mode]** in **[Trial mode] [Trial mode]** is finished extract features and get **[Trial mode]** and **[Trial mode][Trial mode]**W**[Trial mode][Trial mode]**s**[Trial mode][Trial mode]** s **[Trial mode]** select an action **[Trial mode]** for **[Trial mode]** from strategy of **[Trial mode][Trial mode]**a**[Trial mode]**r**[Trial mode]**s'**[Trial mode]**R' = r+F'**[Trial mode]**Q(i,s,a)**[Trial mode][Trial mode]**s**[Trial mode]** and **[Trial mode][Trial mode]**

Based on the above subsections, we propose a hierarchical reinforcement learning in algorithm 1. The algorithm first partitions the goal into subgoals, and then learns strategies for each subgoals. Each subgoal iterates the Q value according to the Q-learning algorithm. When all subgoals converge to the optimal values, the goal is convergent to the optimal value too. The proposed reward function can learn from environment heuristically, and accelerates the exploration of unknown environment, so the convergence is speeded up.

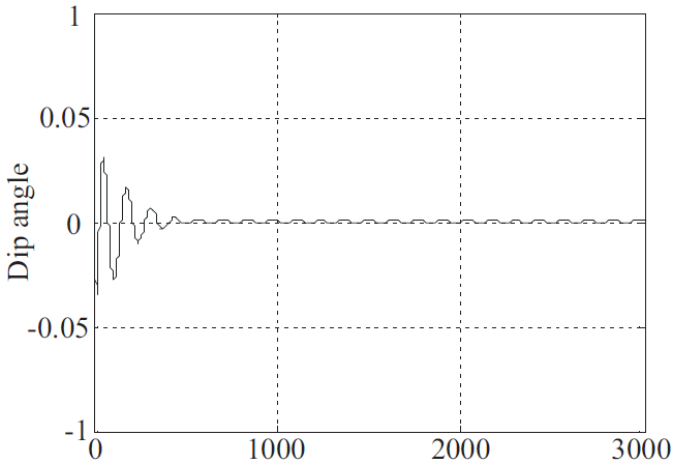
4. EXPERIMENTS

The control system in two wheeled self-balancing robots is unstable. If the two wheeled self-balancing robots want to do other behaviours, they must keep themselves self-balanced, so self-balancing is the basis of the action controller. In order to validate the effectiveness of the proposed algorithm, we do some simulation analysis for controlling the self-balancing of two-wheeled robots.

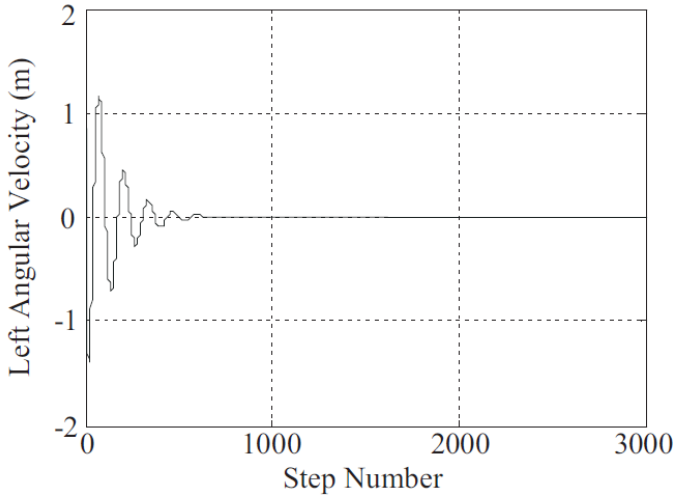
During the reinforcement learning process, the input statuses are the displacement, dip angle and the angular velocities of both wheels, and the output is the control variable. During the evaluation process, the inputs are the displacement, dip angle, the angular velocities of both wheels, and the control variable. In every experiment, when the trail number is larger than 1000 or the step number needed by self-balancing is larger than 20000, we stop the learning process and restart the experiment. After each fail trail, we randomly initialize the statuses and weights in the system, and restart the learning process.



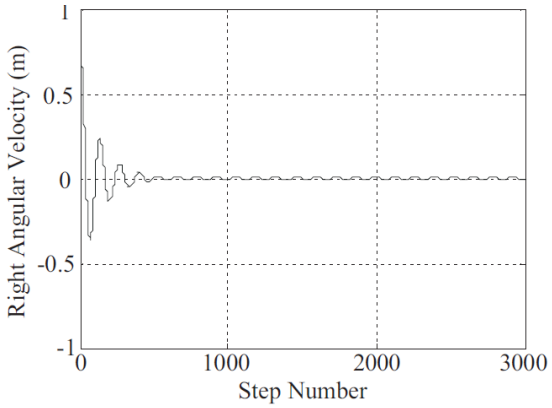
(a) Displacement



(b) Dip angle



(c) Left angular velocity



(d) Right angular velocity

Fig. (2). Observed status convergence.

In the simulation experiments, we observe the displacement, dip angle and the angular velocities of both wheels, and the convergence of these status features are in Fig. (**2**). In addition, we also observe the evaluation function and the error along with the step number, and these results are in Figs. (**3** and **4**) respectively. From these simulation results we can see that, under an unknown environment without any disturbance, the controller needs about 350 steps to be stable, and the convergence speed is very fast.

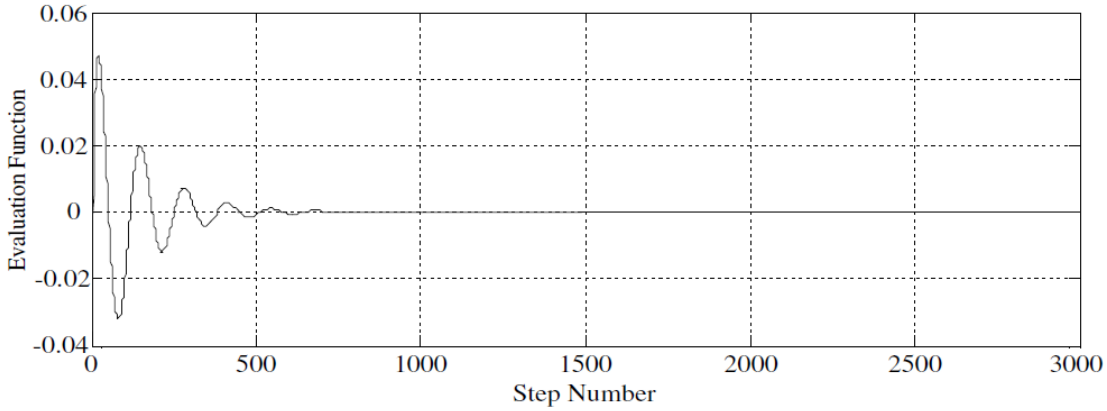


Fig. (3). Simulation result of evaluation function.

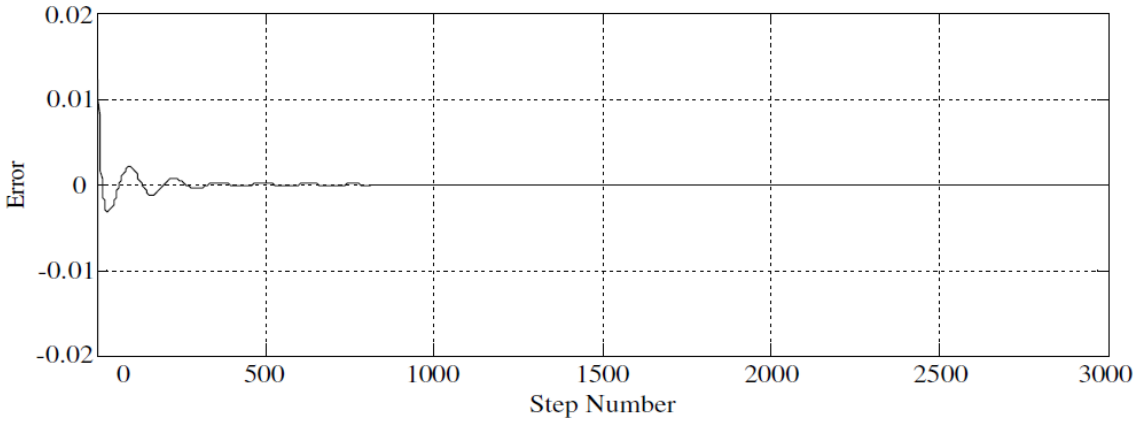


Fig. (4). Simulation result of error.

We compare the evaluation function between our proposed hierarchical reinforcement learning (HRLearning) with

the traditional reinforcement learning (RLearning) algorithm, and the result is in Fig. (**5**). As can be seen from the figure that, our proposed HRLearning converges much faster than RLearning, and is also much more robust than RLearning.

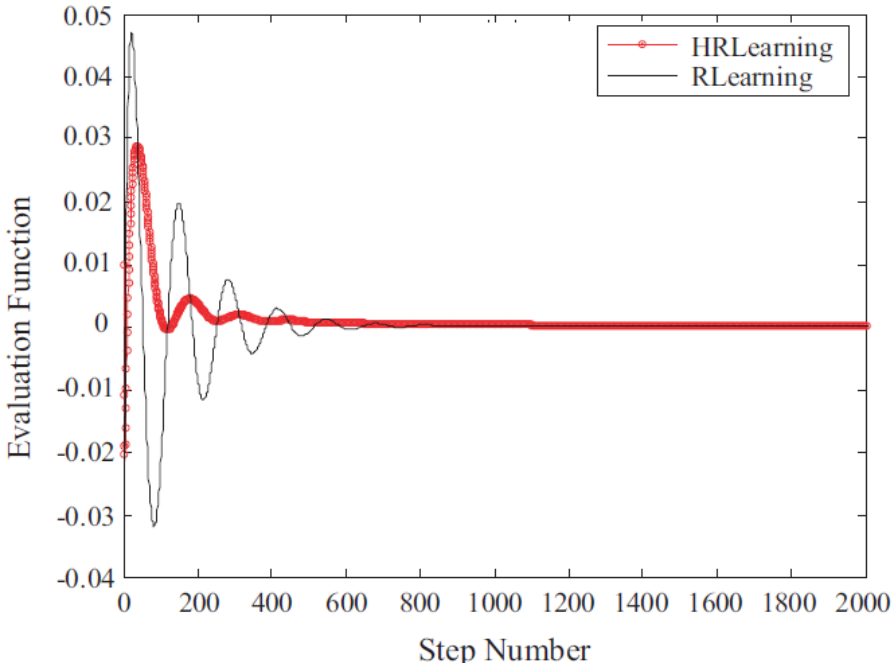
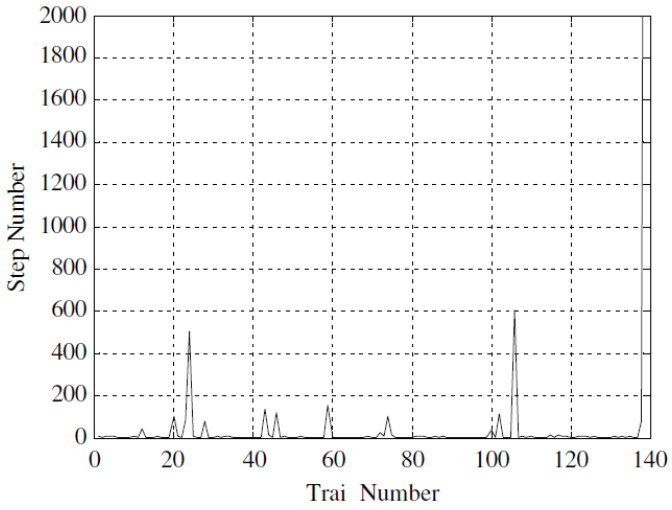
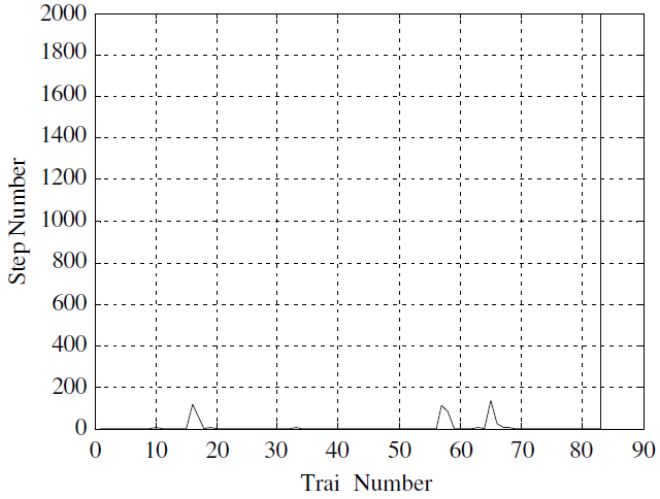


Fig. (5). Comparison of evaluation functions.

We compare the learning curves of the two algorithms, and the results are in Fig. (**6**). Comparing Fig. (**6a**) and Fig. (**6b**) we can see that, RLearning implements self-balanced after 108 failure trails, and our proposed HRLearning algorithm only needs 84 failure trails. So, our algorithm has quicker learning speed and better dynamics than traditional reinforcement learning, and thus has better abilities of self-learning and self-balancing.



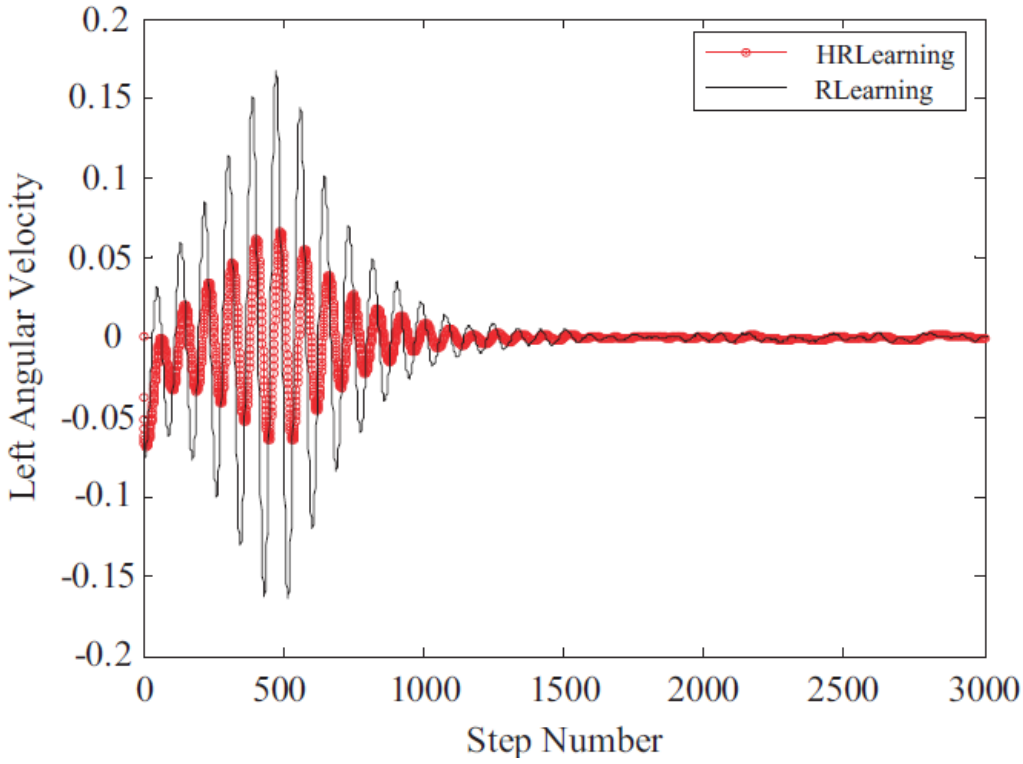
(a) Traditional RLearning algorithm



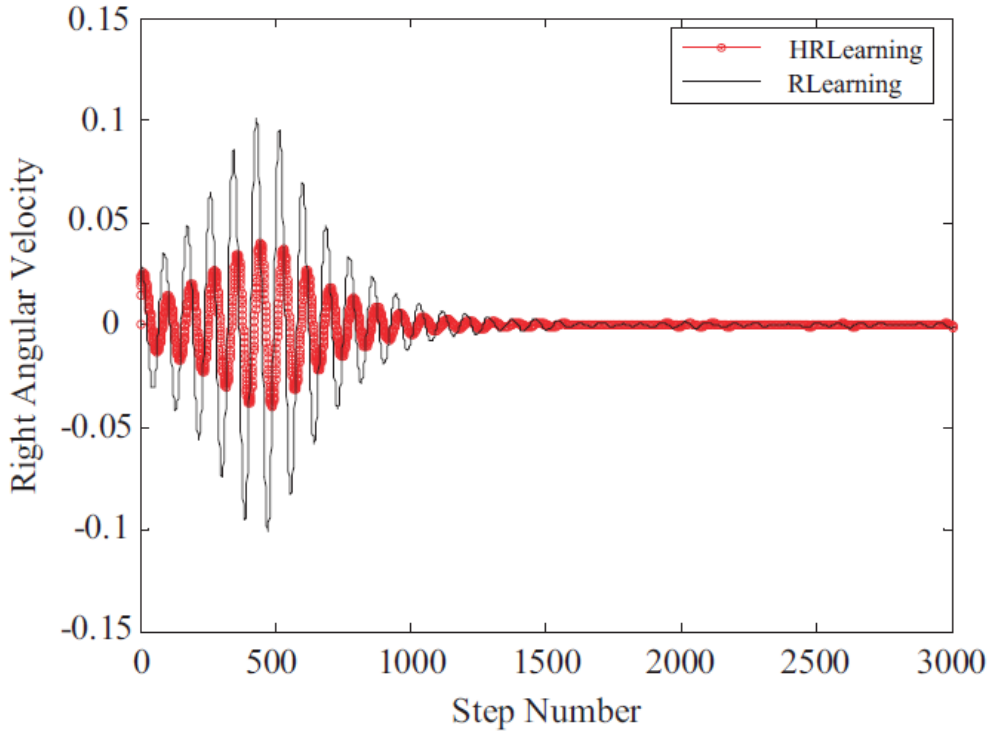
(b) Our HRLearning algorithm

Fig. (6). Comparison of learning curves.

Finally, we compare the angular velocities of both wheels, and the results are in Fig. (**7**). From the figure we can see that, our proposed algorithm has less angular velocities than the traditional reinforcement learning algorithm. This illustrates that, in order to keep self-balanced, the wheels in our system moves gently, and rough moves in RLearning will lead the wheels much more imbalanced.



(a) Angular velocity of left wheel



(b) Angular velocity of right wheel

Fig. (7). Comparison of angular velocity.

CONCLUSION

In this paper, we study the problem of self-balancing in two-wheeled robots, and propose a hierarchical reinforcement learning algorithm. In order to speed up the convergence of learning algorithm, we add a heuristic reward function in each subgoals. After describing the subgoals of hierarchical reinforcement learning, we extract features for subgoals, define a feature value vector and its corresponding weight vector, and propose a reward function with additional subgoal reward function. Simulation experiments show that, the proposed algorithm is more effectiveness than traditional reinforcement learning algorithm in convergent speed. So in our system, the robots can get self-balanced very quickly.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflicts of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] C-C. Tsai, H-C. Huang, and S-C. Lin, "Adaptive neural network control of a self-balancing two-wheeled scooter", *Ind. Electron. IEEE Trans.,* vol. 57, no. 4, pp. 1420-1428, 2010.

[2] C-N. Huang, "The development of self-balancing controller for one-wheeled vehicles", *Engineering,* vol. 2, no. 04, p. 212, 2010. [http://dx.doi.org/10.4236/eng.2010.24031]

[3] Q. Yong, L. Yanlong, and Z. Xizhe, "Balance control of two-wheeled self-balancing mobile robot based on ts fuzzy model", In: *6th Int. Forum Strategic Technol. (IFOST)-IEEE,* vol. 1. 2011, pp. 406-409.

[4] J. Wu, Y. Liang, and Z. Wang, "A robust control method of two-wheeled self-balancing robot", In: *6th Int. Forum Strategic Technol. (IFOST)-*

IEEE, vol. 2. 2011, pp. 1031-1035.

[5] C.J. Watkins, and P. Dayan, "Q-learning", *Mach. Learn.,* vol. 8, no. 3-4, pp. 279-292, 1992. [http://dx.doi.org/10.1007/BF00992698]

[6] A. Goh, "Back-propagation neural networks for modeling complex systems", *Artif. Intell. Eng.,* vol. 9, no. 3, pp. 143-151, 1995. [http://dx.doi.org/10.1016/0954-1810(94)00011-S]

[7] S. Jeong, and T. Takahashi, "Wheeled inverted pendulum type assistant robot: inverted mobile, standing, and sitting motions", In: *International Conference on IEEE/RSJ,* IEEE, 2007, pp. 1932-1937.

[8] S. Miasa, M. Al-Mjali, A.A Ibrahim, and T. Tutunji, "Fuzzy control of a two-wheel balancing robot using dspic", In: *7th International Multi-Conference on Systems Signals and Devices (SSD),* IEEE: Amman, 2010, pp. 1-6. [http://dx.doi.org/10.1109/SSD.2010.5585525]

[9] S. Adam, L. Bu¸soniu, and R. Babu˘ska, "Experience replay for real-time reinforcement learning control", *Syst. Man Cybern., Part C: Appl. Rev., IEEE Trans.,* vol. 42, no. 2, pp. 201-212, 2012.

[10] M.M. Botvinick, "Hierarchical reinforcement learning and decision making", *Curr. Opin. Neurobiol.,* vol. 22, no. 6, pp. 956-962, 2012. [http://dx.doi.org/10.1016/j.conb.2012.05.008] [PMID: 22695048]

[11] R.S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning", *Artif. Intell.,* vol. 112, no. 1, pp. 181-211, 1999. [http://dx.doi.org/10.1016/S0004-3702(99)00052-1]

[12] T.G. Dietterich, "Hierarchical reinforcement learning with the maxq value function decomposition", *J. Artif. Intell. Res.,* vol. 13, pp. 227-303, 2000. [JAIR].

[13] R.E. Parr, *Hierarchical Control and Learning for Markov Decision Processes.,* University of California: Berkeley, 1998.

[14] B. Singer, and M. Veloso, *Learning State Features from Policies to Bias Exploration in Reinforcement Learning. CMU-CS-99-122.,* School of Computer Science Carnegie Mellon University: Pittsburgh, 1999.

[15] G. Hailu, and G. Sommer, "On amount and quality of bias in reinforcement learning", In: *Syst. Man Cybern., IEEE SMC '99 Conf. Proc. 1999 IEEE Int. Conf.,* vol. 2. 1999, pp. 728-733. [http://dx.doi.org/10.1109/ICSMC.1999.825352]

[16] D.L. Moreno, C.V. Regueiro, R. Iglesias, and S. Barro, *Using Prior Knowledge to Improve Reinforcement Learning in Mobile Robotics. Proceedings Towards Autonomous Robotics Systems,* University of Essex: UK, 2004, pp. 1744-8050.

[17] Y-P. Lin, and X-Y. Li, "Reinforcement learning based on local state feature learning and policy adjustment", *Inf. Sci.,* vol. 154, no. 1, pp. 59-70, 2003. [http://dx.doi.org/10.1016/S0020-0255(03)00006-9]

[18] F. Fern'andez, and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent", In: *Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems,* ACM, 2006, pp. 720-727. [http://dx.doi.org/10.1145/1160633.1160762]

[19] N. Mehta, S. Natarajan, P. Tadepalli, and A. Fern, "Transfer in variable-reward hierarchical reinforcement learning", *Mach. Learn.,* vol. 73, no. 3, pp. 289-312, 2008. [http://dx.doi.org/10.1007/s10994-008-5061-y]

[20] E.G. Schultink, R. Cavallo, and D.C. Parkes, *Economic Hierarchical q-Learning.,* AAAI, 2008, pp. 689-695.

[21] M. Stolle, and D. Precup, "Learning options in reinforcement learning", In: *SARA,* Springer, 2002, pp. 212-223. [http://dx.doi.org/10.1007/3-540-45622-8_16]

[22] S. Mannor, I. Menache, A. Hoze, and U. Klein, "Dynamic abstraction in reinforcement learning via clustering", In: *Proceedings of the Twenty-first International Conference on Machine Learning,* ACM, 2004, pp. 71-80. [http://dx.doi.org/10.1145/1015330.1015355]

[23] O. Simsek, A.P. Wolfe, and A.G. Barto, "Identifying useful subgoals in reinforcement learning by local graph partitioning", In: *Proceedings of the 22nd International Conference on Machine Learning,* ACM, 2005, pp. 816-823.

[24] L. Wixson, "Scaling reinforcement learning techniques *via* modularity", In: *Proceedings of the 8th International Workshop on Machine Learning,* 2014, pp. 368-372.