K-Means Clustering based Reinforcement Learning Algorithm for Automatic Control in Robots

Xiaobo Guo[1], Yan Zhai[2]

1 *School of Computer Science and Information Engineering*
Anyang Institute of Technology
Anyang, Henan, China
2 School of Mechanical Engineering
Anyang Institute of Technology
Anyang, Henan, China

***Abstract* — Reinforcement learning is key research in automatic control, and hierarchical reinforcement learning is a good solution to the problem of the curse of dimensionality. Hierarchical reinforcement learning can only deal with discrete space, but the state and action spaces in robotic automatic control are continuous. In order to deal with continuous spaces in hierarchical reinforcement learning, we partition the goal into sub-goals according to a modified K-means clustering algorithm. We designed a modified K-means algorithm to discrete continuous state and action spaces, and also proposed corresponding reinforcement learning algorithms on the discrete spaces. Simulation experiments show that, the proposed approach has lower control error and higher reward of Q-function than traditional hierarchical reinforcement learning algorithm.**

Keywords - robot; automatic control,; hierarchical reinforcement learning; K-means

I. INTRODUCTION

Automatic control is the basis in the research of robots. However, the problem of sequence decision in automatic control depends on continuous state space, and the state values can be infinite, so the state space is also infinite. Reinforcement learning is proposed to solve the problem of real-time control, but it can only deal with limited state space [1]. In order to handle the infinite state space in reinforcement learning, two methods are proposed, i.e. function approximation method based on parameterized representation [2] and discretization based method [3]. Parameter adaption in the first method is much complex, and the parameters may even be not convergent. In the second method, discretization of parameters may result in the following deficit: if the discretization is too coarse-grained, then the reinforcement learning may be chattered around the optimal strategy; and if the discretization is too fine-grained, then state space explodes exponentially, and this is called curse of dimensionality.

In order to solve the problem of curse of dimensionality, researchers proposed the method of hierarchical reinforcement learning [4-6]. The main idea of hierarchical reinforcement learning is reducing the dimensionality of state space by introducing the abstraction mechanism. In detail, the reinforcement learning task is divided into sub-tasks of different hierarchies, and the sub-tasks in each hierarchy run in low dimensional space. Classical algorithms of hierarchical reinforcement learning include Option [7], HAM [8] and MAXQ [9], and they implement the hierarchical mechanism from the perspectives of action, strategy and task respectively.

In this paper, we study the problem of partitioning sub-tasks, or called sub-goals, in hierarchical reinforcement learning. While partitioning the goal of reinforcement learning, we apply a modified K-means clustering algorithm to discrete continuous state and action spaces. We also propose adaptive reinforcement learning algorithms for discrete actions and continuous actions.

The rest of the paper is organized as follows. Section 2 reviews related works about reinforcement learning, especially hierarchical reinforcement learning. In section 3, we propose a K-means based sub-goal partitioning algorithm. In section 4, we propose adaptive reinforcement learning algorithms for discrete actions and continuous actions. Experiments and conclusion are given in sections 5 and 6, respectively.

II. RELATED WORKS

In the study of hierarchical reinforcement learning, the Option algorithm can not only accelerate the learning velocity of the current task greatly, but also the learned knowledge can be applied to other similar learning tasks [10]. The key part of constructing the option is generating sub-goals automatically. Different methods define the sub-goals differently, but they all believe that the sub-goals are useful states that the agent must go by. Currently, a lot of methods [11, 12] are proposed to generate sub-goals automatically, and they all find the sub-goals according to the past experience of the agent.

The methods that find the sub-goals according to the past experience can be classified into two categories. The methods in the first category find the sub-goals via the statistics of the visited state frequencies. Chang et al. [13]

search the sub-goals via the change rates of the numbers of visited states. Solle et al. [14] assume the nodes with the highest visited frequencies to be the sub-goals. Mannor et al. [15] propose that the sub-goals are the frequently visited nodes by which the target can be reached successfully, and not the nodes by which the target cannot be reached. Simsek et al. [16] propose the concept of "relative novelty", which records the ratios of visited frequencies of adjacent states, and thus find sub-goals with all ratios of visited frequencies of adjacent states. The methods in the second category find sub-goals via the state transition graph. Simsek et al. [17] construct the state transition graph by partition the latest experience, and then find sub-goals according to the constructed state transition graph. Menache et al. [18] construct the history graph of the agent, and use the Max-Flow/Min-Cut algorithm to search the sub-goals.

The methods that generate sub-goals can also be classified into the offline and online methods. The offline method [19-21] first generates option, which is used in different learning tasks. This kind of methods can accelerate some learning tasks, but they need more preprocess time and need to re-learn for new tasks, so the practicality of the methods is limited. The online method [22-25] can generate option while running, accelerate the learning process automatically, and thus is the mainstream method.

In addition, machine learning methods can also be used to find the sub-goals. Jaidee et al. [26] assume the finding of sub-goals to be the problem of multiple-instances learning. Some researchers [15, 27] construct the option automatically by the clustering method, and others [28, 29] find the sub-goals in hierarchical reinforcement learning via the partition algorithms in graph theory.

III. K-MEANS BASED SUB-GOAL PARTITIONING ALGORITHM

In this section, we proposed a sub-goal partitioning algorithm for hierarchical reinforcement learning. The proposed algorithm is a K-means based adaptive partition algorithm

A. Centroids based state space discretization

Given a number of centroids in a continuous state space $T = \{t_k \mid k = 1,2,...,m\}$, we can have a partition of clusters in this state space.

Let $x \in \mathbf{R}^n$ be the state of an object, where n is the number of dimensions of the state, then x is a continuous state.

Definition 1. For a cluster centroid t_j, we define its continuous state $R_j \in \mathbf{R}^n$ as

$$R_j = \{x \in \mathbf{R}^n \mid j = \min\{\arg\min_k \| x - t_k \|\}, \quad k = 1,2,...,m\}. \tag{1}$$

The cluster centroid t_j is the optimal match for $\forall x, x \in R_j$. In equation 1, we minimize the subscript j, for there may be multiple centroids whose distances to state x are equal.

Given m centroids $T = \{t_k \mid k = 1,...,m\}$, we can have a partition of the state space, which is denoted as $P_T = \{R_j \mid R_j \subseteq \mathbf{R}^n, j = 1,2,...,m\}$, where R_j is a continuous state space region with respect to centroid t_j, and satisfies

$$\bigcup_{j=1}^{m} R_j = \mathbf{R}^n, \tag{2}$$

where $R_i \cap R_j = \varnothing$, and $\forall i \neq j, i, j = 1,2,...,m$.

Definition 2. Given a partition P for a state space, we define the state partitioning function $s(x): \mathbf{R}^n \to S$ as follows:

$$s(x) = s_j, \text{ if } x \in R_j, \tag{3}$$

where s_j is a discrete state, and $S = \{s_j \mid j = 1,...,m\}$ is a set of discrete states.

B. K-means based adaptive centroid partitioning

Given a set of cluster centroids, we can get a partition for a continuous state space, so adaptive partitioning for a continuous state space is transformed into how to get a suitable distribution of cluster centroids. In this paper, we propose a K-means based adaptive centroid partitioning algorithm, and the main idea of the proposed algorithm is that: predefine the number of maximal cluster centroids, $N_{\max}$, and the distance threshold q_d. For a continuous state x, if the distances between x with all cluster centroids are greater than q_d, i.e. $d_{\min} > q_d$, and the number of current cluster centroids satisfies $N < N_{\max}$, then we add a new cluster centroid $t_{N+1} = x$, and the discrete state for the continuous state x is S_{N+1}. Otherwise, we use the K-means clustering algorithm to adjust the location of cluster centroids, and let x be discrete state of its nearest centroid.

In general, the K-means clustering algorithm uses the already generated data and trains offline. In order to satisfy the request of online train for reinforcement learning, we need to modify it, and the modified K-means algorithm is as follows:

1) get the continuous state $x(t)$;
2) compute the distances between $x(t)$ with all centroids: $d_k = \| x(t) - t_k(t) \|$, $k = 1,2,...,N$;
3) if the minimal distance $d_{\min} = \min_k\{d_k\} > q_d$ and $N < N_{\max}$, then go to step 4, and otherwise, go to step 5;
4) add a cluster centroid at the location of $x(t)$, i.e. $t_{N+1} = x(t)$, $x(t) \in R_{N+1}$, and $N \leftarrow N + 1$;
5) let $x(t) \in R_j$, then t_j is the cluster centroid of continuous state $x(t)$, and thus we can get

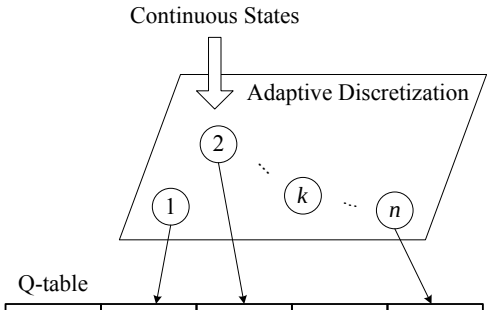
$\min_j\{d_j = d_{\min}\}$. Adjust the location of cluster centroid t_j : $t_j(t+1) = t_j(t) + h(x(t) - t_j(t))$, where h is the learning rate;

let $t = t+1$, and go to step 1.

IV. REINFORCEMENT LEARNING ALGORITHM

A. *Adaptive reinforcement learning algorithm for discrete actions*

Utilizing the algorithm described in section 3.2, we can partition the continuous state space into a discrete state representation, and then we can use the standard reinforcement learning algorithm to train and solve the problem. For the problem of limited discrete action space, we use the substitute trajectory Q-learning algorithm and the adaptive partitioning algorithm to solve the reinforcement learning problem, and the Q-function is in figure 1.



	s_1	s_2	...	s_n
a_1	$Q(s_1,a_1)$	$Q(s_2,a_1)$	...	$Q(s_n,a_1)$
a_2	$Q(s_1,a_2)$	$Q(s_2,a_2)$	...	$Q(s_n,a_2)$
...	...	...	...	...
a_m	$Q(s_1,a_m)$	$Q(s_2,a_m)$	...	$Q(s_n,a_m)$

Figure 1. Q-function after discretization.

The Q-learning algorithm based on adaptive partitioning is as follows:

1) initialize the system state as $x(0)$, the set of cluster centroids as $Æ$ and the number of cluster centroids as $N = 0$;
2) for a continuous state x , if $N < N_{\max}$ and $d_{\min} = \min_k\{d_k = \| x - t_k \|\} > q_d$, then we can get $t_{N+1} = x$, $N = N+1$, where q_d and $N_{\max}$ are defined in section 3; and otherwise, go to step 3;
3) with respect to the discrete state s_j , find the optimal match cluster centroids of x by equation $j(x) = \min_k\{d_k = d_{\min}\}$, adjust the location of t_j as $t_j(t+1) = t_j(t) + h(x(t) - t_j(t))$, where h is the learning rate;
4) applying the e - greedy strategy select action a_t :

$$a_t = \begin{cases} \arg\max Q(s_j, a_i), & \text{with prob. } 1-e, \\ \text{random action}, & \text{with prob. } e. \end{cases} \quad (4)$$

For the situation with minimal accumulated reward, modify arg max into arg min in the above equation;

5) update the suitable trajectory:

$$e(s,a) = \begin{cases} 1, & \text{if } s = s_j, a = a_t, \\ 0, & \text{if } s = s_j, a \neq a_t, \\ gl\, e(s,a) & \text{otherwise.} \end{cases} \quad (5)$$

6) execute action a_t , observe the reward r and the next state x' ;
7) for x' , if $d_{\min} = \min_k\{d_k = \| x' - t_k \|\} > q_d$ and $N < N_{\max}$, then add a new cluster centroid: $t_N = x'$, $N = N+1$; and otherwise, go to step 8;
8) with respect to the discrete state $s_{j'}$, find the cluster centroid $j'(x')$ of the optimal match of x' , and adjust the cluster centroid as
$$t_{j'}(t+1) = t_{j'}(t) + h(x'(t) - t_{j'}(t)) \quad (6)$$
9) select action a' with the e - greedy strategy;
10) update the Q-function for all discrete states s_k and action a_i by
$$Q(s_k,a_i) \leftarrow Q(s_k,a_i) + be(s_k,a_i) + (r + gQ(s_{j'},a') - Q(s_j,a)) \quad (7)$$

let $x \leftarrow x'$, $a \leftarrow a'$, and if x is not a terminal state, then go to step 5.

B. *Adaptive reinforcement learning algorithm for continuous actions*

If the action space is continuous, we still need to partition it. Similar to the processing of continuous state space, we can still use the K-means algorithm to partition the continuous action space. However, different from the continuous state space, the control action in reinforcement learning is controlled by the controller, but the state transition of object is decided by the object itself.

Because of the difference of state controlling strategies, it can be distributed in the whole state space, so it is difficult to set the number of partitions and their accuracies, but however the modified K-means algorithm can be used to handle this problem. So, the K-means based partitioning algorithm for continuous action space is described as follows:

(1)let the initial action cluster centroid be $u_k(0)$, and each action $a_k = u_k$, for each $k = 1,2,...,M$;

(2)for discrete state s_j , select the action a_l according to $a_l = \arg\min_{a_k} Q(s_j, a_k)$, and let the cluster centroid be u_l ;

(3)let the actual action control be $u = u_l + e \times \Omega u$, where e is the random number uniformly distributed in $[-1,1]$, and

$D\mu$ is the predefined action exploring expansion, which decreases gradually;

(4)if the actual control strategy u is more suitable than current action u_l, $gQ(s',a')+r > Q(s_j,a_l)$, and adjust the cluster centroid $u_l = u_l + a(u - u_l)$. s' is the next state, a' is the next action, and a is the learning rate. For the problem of minimal sum of reward, adjust the cluster centroid when $gQ(s',a')+r < Q(s_j,a_l)$.

V. CONCLUSION

In this section, we do some simulation experiments to validate the performance of the proposed approach, and the experiments include three parts.

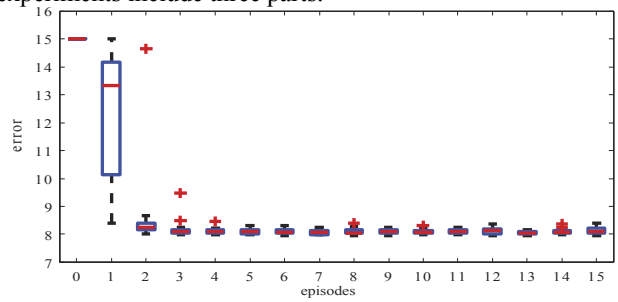


Figure 2. Error of the proposed approach.

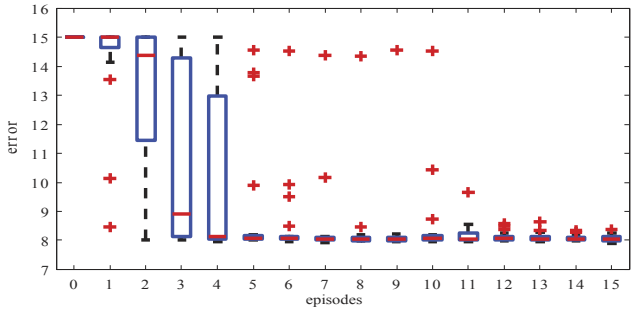
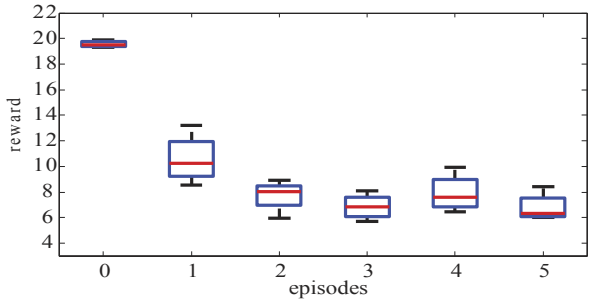
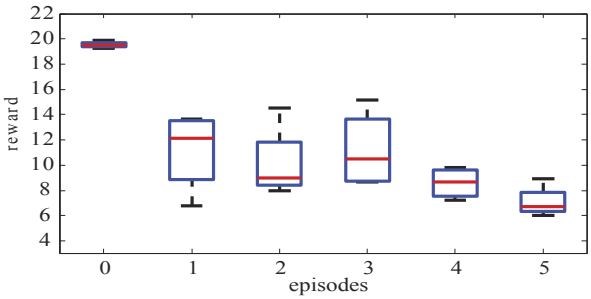


Figure 3. Error of the traditional Q-learning algorithm.



(a) Proposed approach



(b) Traditional Q-learning algorithm

Figure 4. Reward of Q-functions.

A. *Performance of learning error*

We simulated 20 independent runs of a robot, and observed the behaviors of the robot. In each run of experiment, we simulated the behaviors with and without the proposed reinforcement learning approach. Without the proposed approach, we run the robot with traditional Q-learning reinforcement learning algorithm. The average results of the two situations are in figures 2 and 3. Figure 2 is the average behavior error of the proposed approach, and figure 3 is the average behavior error of traditional Q-learning algorithm. Here, the boxes indicate the respective mean and percentiles, while red crosses represent runs interpreted as outliers. As can be seen from the two figures, the proposed approach converges faster and more robustly than the traditional Q-learning algorithm.

B. *Performance of reward of Q-function*

With the same experimental setup as the previous experiments, we also observed the rewards of Q-functions under the above algorithms. The learning results with our proposed method and the traditional Q-learning algorithm are shown in figure 4. The same as section 5.1, the boxes indicate the respective mean and percentiles, while red crosses represent runs interpreted as outliers. It can be seen that learning with the proposed method converges after only 3 episodes, corresponding to 90 seconds system interaction or 180 dynamics samples, and outperforms the traditional Q-learning algorithm.

C. *Comparison of performance*

Finally, we simulated 20 independent runs of a robot, and compared the average normalized rewards of MaxEnt, MaxMargin and our proposed approach. There are many factors that affect the performance of Q-learning, and we randomly select four feature sets. While comparing the rewards of Q-functions, we normalized them such that they are between 0 and 1. While computing the rewards of Q-functions, MaxEnt compute them with the maximal entropy method, and MaxMargin compute them with the max margin method.

Figure 5 is the comparison results of the MaxEnt, MaxMargin and our proposed approach. From the figure we

can see that, our proposed approach has the most normalized reword among the three methods under the four feature sets.

We did the experiments again, and compared the normalized rewards between MaxEnt, MaxMargin and traditional Q-learning algorithm, and the result is in figure 6. The feature sets are selected randomly, so the normalized rewards of MaxEnt and MaxMargin are different from figure 5. However, the normalized rewards of traditional Q-learning algorithm are almost the lowest among the three methods. So, we can conclude that our proposed approach has bigger reward than traditional Q-learning algorithm, and thus is more suitable for the problem of reinforcement learning.

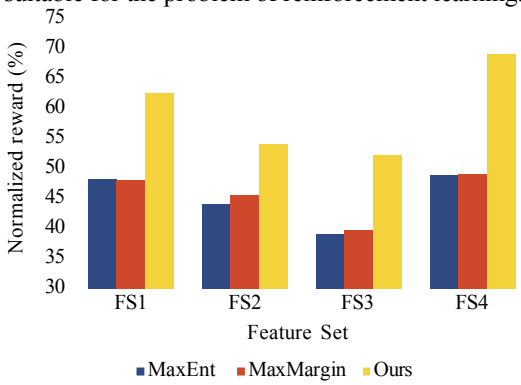


Figure 5. Normalized rewards of MaxEnt, MaxMargin and our proposed approach.

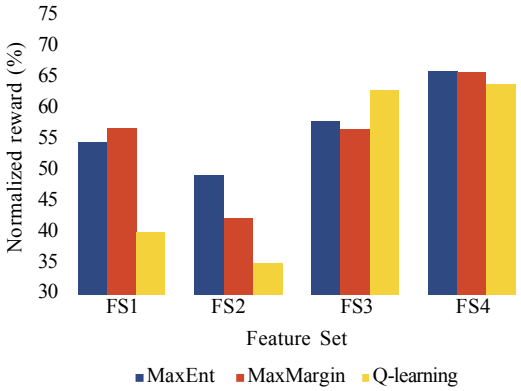


Figure 6. Normalized rewards of MaxEnt, MaxMargin and traditional Q-learning algorithm.

VI. CONCLUSION

In this paper, we study the problem of sub-goal partitioning in hierarchical reinforcement learning. In order to deal with continuous spaces in hierarchical reinforcement learning, we partition the goal into sub-goals according to a modified K-means clustering algorithm. We designed modified K-means algorithms to discrete continuous state and action spaces, and also proposed corresponding reinforcement learning algorithms on the discrete spaces. Simulation experiments show that, the proposed approach has lower control error and higher reward of Q-function than traditional hierarchical reinforcement learning algorithm.

ACKNOWLEDGMENT

This work was financially supported by the Science and Technology Research Program for the Education Department of Hubei province of China (Q20156002).

REFERENCES

[1] Kober J, Bagnell J A, Peters J," Reinforcement learning in robotics: A survey", The International Journal of Robotics Research, 2013.

[2] Prashanth L A, Bhatnagar S," Reinforcement learning with function approximation for traffic signal control", Intelligent Transportation Systems, IEEE Transactions on, vol. 12, No. 02, pp. 412-421, 2011.

[3] Modayil J, White A, Sutton R S," Multi-timescale nexting in a reinforcement learning robot", Adaptive Behavior, vol. 22, No. 02, pp. 146-160, 2014.

[4] Botvinick M M," Hierarchical reinforcement learning and decision making", Current option in neurobiology, vol. 22, No. 06, pp. 956-962, 2012.

[5] Cao F, Ray S," Bayesian hierarchical reinforcement learning",Advances in Neural Information Processing Systems,pp. 73-81, 2012.

[6] Ribas-Fernandes J J F, Solway A, Diuk C, et al," A neural signature of hierarchical reinforcement learning", Neuron, vol. 71, No. 02, pp. 370-379, 2011.

[7] Sutton R S, Precup D, Singh S," Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning", Artificial intelligence, vol. 112, No. 01, pp. 181-211, 1999.

[8] Parr R E," Hierarchical control and learning for Markov decision processes",University of california at Berkeley, 1998.

[9] Dietterich T G," An overview of MAXQ hierarchical reinforcement learning",Abstraction, Reformulation, and Approximation,Springer Berlin Heidelberg, pp.26-44, 2000.

[10] Diuk C, Schapiro A, Córdova N, et al," Divide and conquer: hierarchical reinforcement learning and task decomposition in humans",Computational and Robotic Models of the Hierarchical Organization of Behavior,Springer Berlin Heidelberg, pp.271-291, 2013.

[11] Doll B B, Simon D A, Daw N D," The ubiquity of model-based reinforcement learning", Current option in neurobiology, vol. 22, No. 06, pp. 1075-1081, 2012.

[12] Andre D, Russell S J," State abstraction for programmable reinforcement learning agents",AAAI/IAAI,pp. 119-125, 2002.

[13] Chang S, Yang G, Shi-fu C," The study of recognizing options based on SMDP", Pattern Recognition and Artificial Intelligence, vol. 18, No. 06, pp. 679-684, 2005.

[14] Stolle M, Precup D," Learning options in reinforcement learning",5th International Symposium, SARA 2002, Kananaskis, Alberta, Canada, pp. 212-223, 2002.

[15] Mannor S, Menache I, Hoze A, et al," Dynamic abstraction in reinforcement learning via clustering",Proceedings of the twenty-first international conference on Machine learning, ACM, pp. 71, 2004.

[16] Şimşek Ö, Barto A G," Using relative novelty to identify useful temporal abstractions in reinforcement learning",Proceedings of the twenty-first international conference on Machine learning, ACM, pp. 95, 2004.

[17] Şimşek Ö, Wolfe A P, Barto A G," Identifying useful subgoals in reinforcement learning by local graph partitioning",Proceedings of the 22nd international conference on Machine learning,ACM,pp. 816-823, 2005.

[18] Menache I, Mannor S, Shimkin N," Q-cut—dynamic discovery of sub-goals in reinforcement learning",Machine Learning: ECML 2002,Springer Berlin Heidelberg,pp. 295-306, 2002.

[19] Lewis F L, Vrabie D, Vamvoudakis K G," Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers", Control Systems, IEEE, vol. 32, No. 06, pp. 76-105, 2012.

[20] Gregory S, Blair R J, Simmons A, et al,“ Punishment and psychopathy: a case-control functional MRI investigation of reinforcement learning in violent antisocial personality disordered men”, The Lancet Psychiatry, vol. 2, No. 02, pp. 153-160, 2015.

[21] Özcan E, Mısır M, Ochoa G, et al,“ A Reinforcement Learning: Great-Deluge Hyper-Heuristic”, Modeling, Analysis, and Applications in Metaheuristic Computing: Advancements and Trends: Advancements and Trends, pp. 34, 2012.

[22] Bishop J, Miikkulainen R,“ Evolutionary feature evaluation for online reinforcement learning”,Computational Intelligence in Games (CIG), 2013 IEEE Conference on. IEEE,pp. 1-8, 2013.

[23] Yang Q, Jagannathan S,“ Reinforcement learning controller design for affine nonlinear discrete-time systems using online approximators”, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 42, No. 02, pp. 377-390, 2012.

[24] Garlapati A, Raghunathan A, Nagarajan V, et al,“ A Reinforcement Learning Approach to Online Learning of Decision Trees”, arXiv preprint arXiv:1507, 2015.

[25] Ortner R, Ryabko D,“ Online regret bounds for undiscounted continuous reinforcement learning”,Advances in Neural Information Processing Systems, pp.1763-1771, 2012.

[26] Jaidee U, Muñoz-Avila H, Aha D W,“ Case-based goal-driven coordination of multiple learning agents”,Case-Based Reasoning Research and Development, Springer Berlin Heidelberg,pp. 164-178, 2013.

[27] Yang Y, Cui Z, Wu J, et al,“ Fuzzy c-means clustering and opposition-based reinforcement learning for traffic congestion identification”, Journal of Information and Computer Science, vol. 9, pp. 2441-2450, 2012,

[28] Li X, Mabu S, Hirasawa K,“ A novel graph-based estimation of the distribution algorithm and its extension using reinforcement learning”, Evolutionary Computation, IEEE Transactions on, vol. 18, No. 01, pp. 98-113, 2014.

[29] Moradi P, Shiri M E, Rad A A, et al,“ Automatic skill acquisition in reinforcement learning using graph centrality measures”, Intelligent Data Analysis, vol. 16, No. 01, pp. 113-135, 2012.