Subgoal Discovery in Reinforcement Learning Using Local Graph Clustering

Negin Entezari[1] Mohammad Ebrahim Shiri[1] Parham Moradi[2]

[1] *Department of Computer Sicence, Amirkabir University of Technology, Tehran, Iran*
[2] *University of Kurdistan, Department of Electrical & Computer Engineering, Sanandaj, Iran*
negin.entezari@aut.ac.ir, shiri@aut.ac.ir, pmoradi@aut.ac.ir

Abstract

Reinforcement Learning is an area of machine learning that studies the problem of solving sequential decision making problems. The agent must learn behavior through trial-and-error interaction with a dynamic environment. Learning efficiently in large scale problems and complex tasks demands a decomposition of the original complex task into simple and smaller subtasks. In this paper, we present a subgoal-based method for automatically creating useful skills in reinforcement Learning. Our method identifies subgoals using a local graph clustering algorithm. The main advantage of the proposed algorithm is that only the local information of the graph is considered to cluster the agent state space. Clustering of the transition graphs corresponding to MDPs can be performed in linear time using the proposed method. Subgoals discovered by the algorithm are then used to generate skills using the option framework. Experimental results show that the proposed subgoal discovery algorithm has a dramatic effect on the learning performance.

Keywords: *Hierarchical Reinforcement Learning, Option, Skill Acquisition, Subgoal Discovery, Graph Clustering*

1. Introduction

Reinforcement Learning (RL) [1] studies the problem of solving sequential decision making problems in which an intelligent agent must learn behaviour through interactions with an unknown environment. In each step of interaction, the agent selects an action that causes a change in the state of the environment and then receives a scalar reinforcement signal called reward. The agent's objective is to learn a policy that maximizes the long-term reward.

Large state space and lack of immediate reward are crucial problems in the area of reinforcement learning. Two approaches which have been proposed to tackle these problems are function approximation [2] and task decomposition [3-6]. The main idea of Hierarchical Reinforcement Learning (HRL) methods is decomposition of the learning task into simple and smaller subtasks that increases the agent's learning performance. One common way to decompose the complex task to the set of simple subtasks is identifying important states known as subgoals and then learning sub-policies to reach these subgoals [7-14]. These sub-policies are also called temporally extended actions or skills or macro actions. A macro-action or a temporally extended action is a sequence of actions chosen from the primitive actions. A suitable set of skills can accelerate learning. It is desirable to devise methods by which an agent is automatically able to discover skills and construct high-level hierarchies of reusable skills [8-19].

There are a variety of approaches to discover subgoals automatically. Some approaches introduce highly visited states as subgoals [7-9]. Another group of approaches are graph based methods that introduce the border states between densely connected regions as subgoals [10-13]. In these approaches, the agent' transition history is mapped to a graph and each observed state is considered as a node in the graph and each transition between states is mapped to an edge in the graph. Graph nodes indicate the state space and the edges represent the state transitions. Menache et al. [10] discovered bottleneck states by performing Max-Flow/Min-Cut algorithm on the transition graph. In [12], the N-Cut algorithm is utilized to find landmark states of the local transition graph. In addition, recently researchers have shown interest in graph centrality measures to identify important states. To this aim Şimşek and Barto in [9] utilized betweenness centrality.

In this paper we present a local graph clustering algorithm to provide an appropriate partitioning of the input graph. The main characteristic of our clustering algorithm is that it only uses local information of the graph. Subgoals are discovered in linear time using the proposed method and therefore our approach is suitable for finding subgoals of large scale reinforcement learning problems. Other methods proposed earlier, use global information of graph and they are inapplicable for large graphs. Şimşek et al. in [12] addressed this problem by considering a local scope of the transition graph. They perform the Normolized-cut algorithm on the local transition graph which requires setting many parameters for different domains, while in our local graph clustering algorithm one parameter should be adjusted.

The rest of the paper is organized as follows. The reinforcement learning and its extension to use macro-actions is described in section 2. In section 3, the proposed method is described. In Section 4 the time complexity of the proposed algorithm is analysed. The benchmark tasks, simulation and results are described in section 5, and section 6 contains the final discussion and the concluding remarks.

2. Reinforcement Learning With Option

In RL, the environment is usually formulated as a finite-state Markov Decision Process (MDP). Due to the Markov property of the environment, the probability distribution of new state and reward after executing an action depends only on the previous state and action, not on the entire history of states. An MDP consists of a set of states S, a set of actions A, a reward function $R(s,a)$, and a transition function $P(s'|s,a)$. In each step of interaction, in each state $s \in S$, the agent chooses an action $a \in A$. The value of this state transition is communicated to the agent through a scalar reinforcement signal called reward, $R(s,a)$ and with the probability of $P(s'|s,a)$, the agent observes a new state $s' \in S$. The agent's task is to optimize the action choices in such a way that it maximizes the expected discounted return:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \qquad (1)$$

where γ is the discount rate, $0 \le \gamma \le 1$. Q-Learning is the most commonly used algorithm in RL. Macro Q-Learning is the hierarchical form of Q-Learning including options. To represent skills, we use option framework [4]. Options are a generalization of primitive actions including temporally extended courses of actions. Even if adding options to primitive actions offers more choices to the agent and makes decision making more complicated, the decomposition of the original task provided by options can simplify the task and facilitate learning. An option is a tuple $< I, \pi, \beta >$, where $I \subseteq S$ is an initiation set consisting of all the states in which the option can be initiated. $\pi : S \times O \to [0,1]$, is an option policy where O is the set of all possible options and $\beta : S \to [0,1]$ is a termination condition that is the probability of terminating an option in each state. Macro Q-Learning updates the value of one-step options (primitive actions) and multi-step options. The update rule is:

$$Q(s,o) \to Q(s,o) + \alpha \left[r + \gamma^k \max_{o' \in O_{s'}} Q(s',o') - Q(s,o) \right], \qquad (2)$$

where k is the number of time steps that the option o is executing, and r is the cumulative discounted reward over this time, $r = \sum_{i=0}^{k} \gamma^i r_{t+i+1}$.

3. Proposed Method

In this paper we utilize a local graph clustering algorithm to indicate dense regions of the agent's transition graph and consider the border states of highly intra-connected regions as subgoals. A local graph clustering algorithm only uses local information to generate a clustering of the input graph. Local clustering algorithms address the complexity drawbacks of global clustering algorithms and are suitable for clustering large graphs. Given a graph $G = (V,E)$ where V is the set of vertices and E is the set of edges, the main goal of the clustering algorithm is finding a set of clusters $\mathrm{X_i}$ $\left(\mathrm{i} \in 1,...,\mathrm{k} \right)$ which maximizes the intra-cluster connectivity and minimizes the inter-cluster connectivity. To this aim the ratio $R(X)$ is defined according to definition 1. The outline of the proposed algorithm is shown in Figure 1.

Definition 1 : If $X \subseteq V$, the ratio $R(X)$ is defined as follows:

$$R(X) = \frac{\sum_{i \in X, j \in b(X)} a_{ij}}{\sum_{i \in n(X), j \in b(X)} a_{ij}} \qquad (3)$$

where $b(X)$ is the border nodes of X, $n(X)$ indicates the neighbors of border nodes that are not a member of X and a_{ij} denotes the proper element of the adjacency matrix of G, that is:

$$a_{ij} = \begin{cases} 1, & if\ i\ and\ j\ are\ connected\ through\ an\ edge, \\ 0, & if\ not. \end{cases} \qquad (4)$$

Neighbors that maximize the ratio $R(X)$ will be added to X in k iterations or while there are no more such neighbors. k is an input parameter of the clustering algorithm and it is usually a small integer.

Definition 2: The set of candidate nodes $C(X)$ is defined as follows:

$$C(X) := \arg\max_{v \in n(X)} R(X+v) \qquad \textbf{(5)}$$

Definition 3: the set of final candidate nodes that can be added to X is defined as follows:

$$C_f(X) := \arg\max_{u \in C(X)} \sum_{v \in n(X)} a_{uv} \qquad \textbf{(6)}$$

Nodes to be added to X are selected in two steps:

Define candidate nodes $C(X)$ that can be added to X according to definition 2.

To ensure that the intra-cluster connectivity is maximized in the future, according to definition 3, nodes of $C(X)$ that have maximum connectivity with neighbor nodes form final candidates.

All members of $C_f(X)$ are then added to X if the following condition is satisfied:

$$R\left(X + C_f(X)\right) \geq R(X) \qquad \textbf{(7)}$$

```
The Local graph clustering algorithm
foreach $v \in V$ do
    $X := \{v\}$;
    while $|n(X)| > 0$ and $repeat \leq k$ do
        $C(X) := \arg\max_{u \in n(X)} R(X+u)$;
        $C_f(X) := \arg\max_{u \in C(X)} \sum_{y \in n(X)} a_{uy}$;
        if $R(X \cup C_f(X)) \geq R(X)$ then $X := X \cup C_f(X)$;
        else break;
        $repeat = repeat + 1$;
    end
end
merge overlapping clusters;
```

FIGURE 1. The outline of the proposed local graph clustering algorithm.

Figure 2 shows different steps of the algorithm on a sample graph. The original graph is represented in figure 2(a). Initially, each graph node $v \in V(v = 1,...,12)$ is considered as a cluster and in this sample graph there are 12 clusters in the beginning. Parameter k is set to 4. For each node v, all neighbors of v are examined and candidates nodes are then selected according to (6) and are then inserted to X. Repeatedly neighbors of new set X are searched for candidates until the maximal set X is achieved or until k iterations. Let $v = 1$, initially we have $X = \{1\}$. In this step, neighbors of X are nodes 2 and 3. According to equation (3), $R(X + 2) = 1$ and $R(X + 3) = 2/3$, so node 2 is the only candidate node and will be inserted in $X = \{1\}$. It can be seen that $X = \{1, 2\}$ is a maximal set for $v = 1$ and adding nodes 3 or 4 to X will decrease the ratio $R(X)$. Figure 2(b) shows tha maximal set X for node 1. These steps are repeated for all other nodes of V. As can be seen in figure 2(c), The result of this step is a soft clustering of the given graph. Overlapping clusters in figure 2(c) are then merged into one cluster. The final non-overlapping clusters as shown in figure 2(d), are our desirable clusters. Finally Border nodes of clusters are identified as subgoals. In our example, nodes 4 and 9 are the border nodes of clusters. Figure 3 also shows the result of clustering on the transition graph of a four-room gridworld.

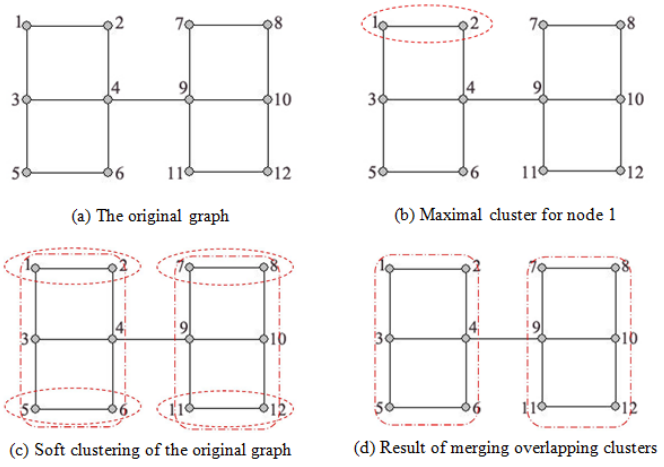

(a) The original graph
(b) Maximal cluster for node 1
(c) Soft clustering of the original graph
(d) Result of merging overlapping clusters

FIGURE 2. Different steps of the proposed graph clustering algorithm on a sample graph.

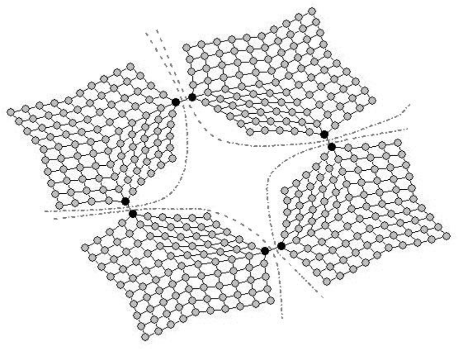


FIGURE 3. Result of clustering the transition graph of a six-room gridworld. Black nodes are border nodes of clusters.

Subgoals discovered by the proposed method are used to form skills. Skills are generated using option framework and each subgoal is considered as the terminal state of an option. Initial states of options are states in the same cluster with the subgoal. The generated options terminate with probability one at the goal state and outside the initiation set; at all other states they terminated with probability zero. Figure4 shows the outline of skill acquisition.

```
Skill acquisition algorithm

Interact with environment and learn using Macro Q-Learning.
Construct transition graph.
Run local graph clustering algorithm.
Introduce border nodes of clusters as subgoals.
Learn the options for reaching subgoals.
Add the new options to the agent's action set.
```

FIGURE 4. The outline of the skill acquisition algorithm.

4. Complexity Analysis

To compute the complexity of our algorithm, we consider the worst case running time of the local graph clustering algorithm. As described earlier, in each step of the algorithm, neighbors of the current cluster are examined to choose the candidate nodes. Let $d_{\max}$ be the maximum degree of vertices of the graph. Consider the case in which the cluster X contains only the vertex $v \in V$. In this case cluster X has at most $d_{\max}$ neighbors and in the worst case all of them will be added to X. Therefore in the *k*-th iteration of the while loop at most ${d_{\max}}^k$ neighbors should be examined. So the while loop will iterate $O(n{d_{\max}}^{k+1})$ times in the worst case. This process is repeated for all *n* vertices of the graph. In addition, merging the overlapping clusters can be performed in $O(n)$ time. Therefore the local graph clustering algorithm needs $O(n{d_{\max}}^{k+1})$ in the worst case.

Due to the fact that the number of actions in MDPs are restricted and mostly the transition graphs are sparse, the maximum degree of vertices $d_{\max}$ is a small value. As stated earlier, parameter k is mostly set to a small positive integer. Therefore the time complexity of the proposed algorithm for sparse transition graphs will be reduced to $O(n)$.

5. Experimental Results

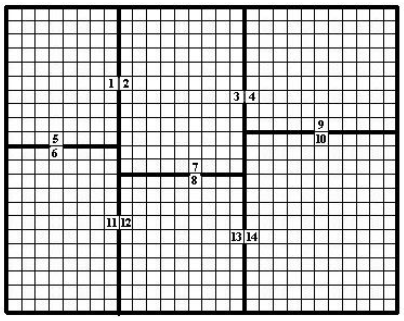
We experimented with the proposed algorithm in two domains: a six-room gridworld and a soccer simulation test bed. In our experimental analysis, the agent used an ϵ-greedy policy, where the ϵ was set to 0.1. The learning rate α and discount rate γ were set to 0.1 and 0.9 respectively. The generated options terminated with probability one at corresponding subgoal states indicated by the algorithm. The options also terminated with the same probability at the goal state and outside the initiation set; at all other states they terminated with probability zero.

5.1. Six-room Gridworld

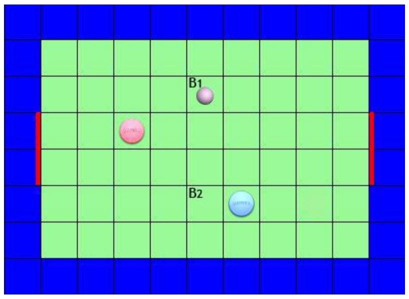
The six-room gridworld [4] is shown in figure 5(a). From any state the agent is able to perform four actions, up, down, left and right. There is a reward of -1 for each action and a reward of 1000 for actions directing the agent to the goal state. We performed 100 randomly selected episodic tasks (random start and goal states) and each task consists of 80 episodes.

5.2. Soccer Simulation Test Bed

Soccer domain shown in figure 5(b) is a 6×9 grid. At each episode the ball is randomly located at one of two positions B_1 and B_2. Two agents try to find and own the ball and score a goal. To score a goal, each player owning the ball must reach to the opponent goal (represented with red line in the figure 5(b)). Each agent has five primitive actions: North, East, South, West and Hold. The hold action does not change the position of agent. Agents receive a reward of -1 for each action and a reward of +100 for action causing the agent find and own the ball. Action that leads scoring a goal gives the reward of +1000 to the agent. If the agent owing the ball is going to enter the other agent's location, the ball owner will change and agents stay in their locations. If the agent is going to enter the same location as other agent owing the ball, with probability 0.8, owner of the ball does not change and agents stay in their location.



(a)

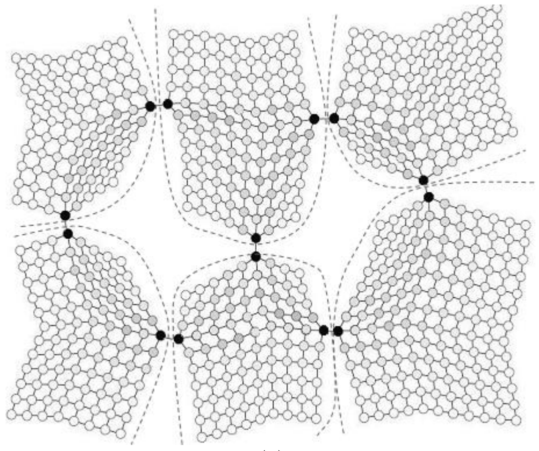


(b)

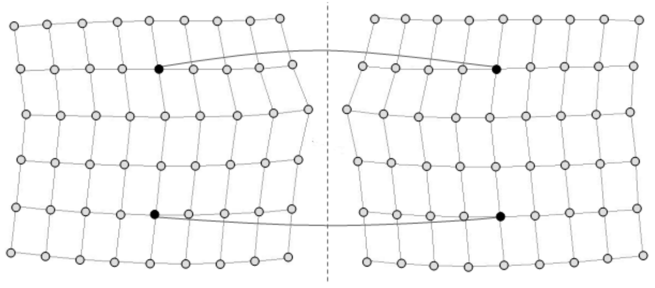
FIGURE 5. (a) Six-room gridworld (b) Soccer simulation test bed.

5.3 Results

The transition graph of each domain was clustered by the proposed clustering algorithm and the border states of clusters were identified as subgoals. In the six-room gridworld, as illustrated in figure 5(a), cells labeled with 1,2,…,14 are identified as subgoals by our clustering algorithm. These 14 states correspond to border nodes (black nodes) of transition graph of six-room gridworld shown in figure 6(a).



(a)



(b)

FIGURE 6. (a) Transition graph of a six-room gridworld. (b) Transition graph of soccer simulation test bed. Black nodes are border nodes of clusters introduced as subgoals.

Figure 7 shows the average steps to reach the goal. Compared to the Q-Learning with only primitive actions, the skills improved performance remarkably. In addition, as can be seen in figure 8, average reward obtained by the learning agent is significantly increased.

The same experiments were implemented in soccer simulation domain and similar results were achieved. Figure 9 compares the number of goals scored by the agent while learning with primitive actions with the case of learning with additional generated options. As expected, options speed up the learning process and as a result the agent is able to score a larger number of goals.

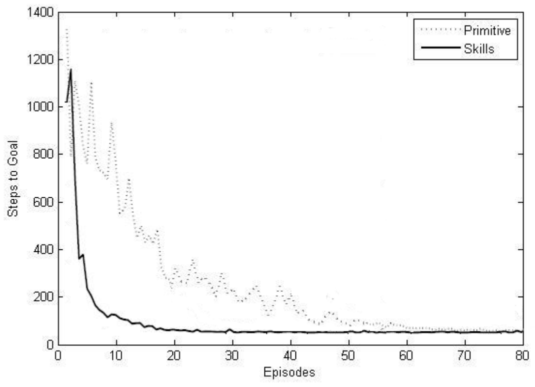


FIGURE 7. Comparison of Q-Learning and Q-Learning with options generated based on the subgoals extracted by the proposed algorithm in a six-room gridworld. Average number of steps to reach the goal is shown.

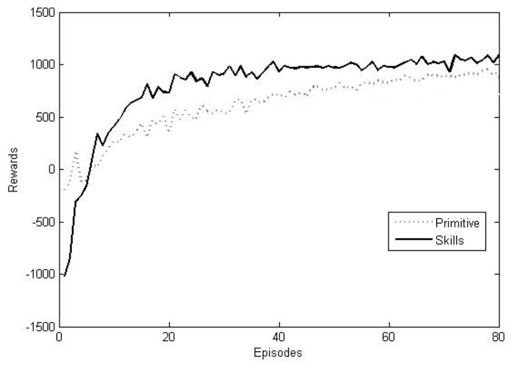


FIGURE 8. Average reward obtained by the agent, comparing Q-Learning with primitive actions and with skills.

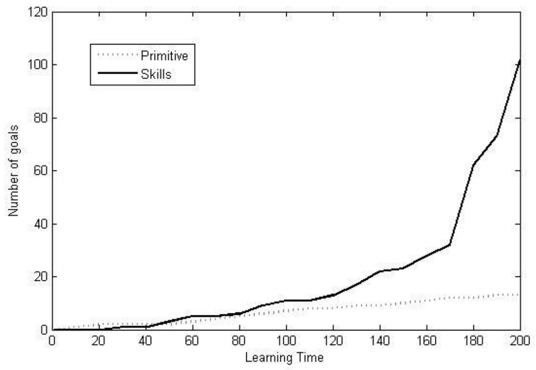


FIGURE 9. The number of goals scored by the agent. Comparing Q-Learning with Q-Learning with options generated based on the subgoals extracted by the proposed algorithm in soccer simulation test bed.

6. Conclusion

This paper presents a graph theoretic method for discovering subgoals by clustering the transition graph. The proposed algorithm is a local clustering algorithm that solely uses local information to generate an appropriate clustering of the input graph. Global clustering algorithms have time complexity $O(N^3)$, where N is the total number of visited states. The L-Cut algorithm [12] which is a local graph partitioning method is of complexity $O(h^3)$, with h as the number of states in local scope of the transition graph. One drawback of the L-cut algorithm is that the local cut may not be a global cut of the entire transition graph. Another disadvantage of the L-Cut algorithm is that it demands setting a lot of parameters. In L-Cut a global clustering algorithms is used to partition a local transition graph, whereas we use a local graph clustering approach on transition graphs. The proposed algorithm uses the local information to generate a global clustering of the transition graph and comparing to global graph clustering algorithms has less time complexity and is able to discover subgoals in linear time. In addition, just one parameter setting is needed in the algorithm that can be easily adjusted with no special knowledge . Our Experiments in two benchmark environments show that discovering subgoals and including policies to achieve these subgoals in the action set can significantly accelerate learning in other, related tasks.

References

[1] L. P. Kaelbling, M. L. Littman: Reinforcement Learning: A Survey, J. Artificial Intelligence Research 4, 1996.

[2] D. B. Bertsekas, J. N. Tsitsiklis: Neuro-dynamic programming, Athena Scientific, 1995.

[3] R. Parr, S. Russell: Reinforcement learning with hierarchies of machines, in Proc. the 1997 conference on Advances in neural information processing systems, Cambridge, MA, USA, pp. 1043-1049, 1997.

[4] R. Sutton, D. Precup, S. Singh: Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning, J. Artificial Intelligence, vol. 112, pp. 181-211, 1999.

[5] T. G. Dietterich: Hierarchical reinforcement learning with the MAXQ value function decomposition, J. Artificial Intelligence, vol. 13, pp. 227-303, 2000.

[6] A. G. Barto, S. Mahadevan: Recent Advances in Hierarchical Reinforcement Learning, Discrete Event Dynamic Systems, vol. 13, pp. 341-379, 2003.

[7] Ö. Şimşek, A. G. Barto: Learning Skills in Reinforcement Learning Using Relative Novelty, pp. 367-374, 2005.

[8] B. L. Digney: Learning hierarchical control structures for multiple tasks and changing environments, Proc. the fifth international conference on simulation of adaptive behavior on From animals to animats 5, Univ. of Zurich, Zurich, Switzerland, 1998.

[9] A. McGovern, A. G. Barto: Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density, Proc. the eighteenth international conference on machine learning, pp. 361-368, 2001.

[10] I. Menache, S. Manno, N. Shimkin: Q-Cut - Dynamic Discovery of Sub-goals in Reinforcement Learning, Proc. the 13th European Conference on Machine Learning, 2002.

[11] S. Mannor, I. Menache, A. Hoze, U. Klein: Dynamic abstraction in reinforcement learning via clustering, Proc. the twenty-first international conference on Machine learning, Banff, Alberta, Canada, 2004.

[12] Ö. Şimşek, A. P. Wolfe, A. G. Barto: Identifying useful subgoals in reinforcement learning by local graph partitioning, Proc. the 22nd international conference on Machine learning, Bonn, Germany, 2005.

[13] S Jing, G. Guochang, L. Haibo: Automatic option generation in hierarchical reinforcement learning via immune clustering, in Systems and Control in Aerospace and Astronautics. ISSCAA 2006. 1st International Symposium on, 2006, pp. 4 pp.-500, 2006.

[14] Ö. Şimşek and A. G. Barto: Skill Characterization Based on Betweenness, in Advances in Neural Information Processing Systems 21, pp. 1497-1504, 2009.

[15] A. Jonsson, A. G. Barto: Automated state abstraction for options using the u-tree algorithm, In Advances in Neural Information Processing Systems: Proceedings of the 2000 Conference, pp. 1054-1060, 2001.

[16] S. Elfwing, E. Uchibe, K. Doya: An Evolutionary Approach to Automatic Construction of the Structure in Hierarchical Reinforcement Learning, Genetic and Evolutionary Computation, pp. 198-198, 2003.

[17] A. Jonsson and A. Barto: A causal approach to hierarchical decomposition of factored MDPs, In Proc. the 22nd international conference on Machine learning, Bonn, Germany, 2005.

[18] A. Jonsson, A. Barto: Causal Graph Based Decomposition of Factored MDPs, J. Machine Learning, Res., vol. 7, pp. 2259-2301, 2006.

[19] N. Mehta, S. Ray, P. Tadepalli, T. G. Dietterich: Automatic discovery and transfer of MAXQ hierarchies, In Proc. of the 25th international conference on Machine learning, Helsinki, Finland, 2008.