Incremental Skills Discovery based on the Bottleneck Concept

Pierre-Luc Bacon PLBACON@CIM.MCGILL.CA
School of Computer Science, McGill University, Montreal, QC, H3A 2A7

Abstract

The options framework of (Sutton et al., 1999) provides temporal abstraction for reinforcement learning (RL) agents by allowing them to use macro-actions instead of primitive ones. If options intuitively capture the notion of *skills*, the problem of effectively designing them remains challenging. A useful class of approaches based on graphical representation of the state space has been shown to produce useful options when defining *bottleneck* states as subgoals. We evaluate some these heuristics and propose a near-linear incremental skill discovery algorithm based on a label propagation algorithm for community detection (Raghavan & Albert, 2007).

1. Introduction

The complex nature of learning agents interacting with the real world must also come with strategies which allow for reasoning to take place at different levels of abstraction. The options framework (Sutton et al., 1999) describes the idea of temporally-extended actions which are composed of sequences of primitive actions used towards achieving a subgoal. By developing options, or *skills*, an agent can learn more efficiently by reusing prior knowledge instead of continuously reasoning at the atomic scale of primitive actions. An example of an option for a robot equipped with an arm might be defined by a subpolicy for reaching and opening a door knob using the sensory information from tactile sensors and an actions set defined over the possible motor commands at each joint. By learning such policy and later using it as prior knowledge for some related task, learning time can be greatly reduced within large state spaces.

If the notion of a skill can be understood intuitively, it is much less obvious to unequivocally define the conditions under which to recognize options. Furthermore, relying on tailored-made options might quickly become overly complicated due to the abstract task representation that a human operator might have work with. In this paper, we are thus interested in addressing this question of "what makes for a useful skills", but more importantly "how can a reinforcement learning agent identify them throughout its lifetime ?"

We approach this problem by considering the interaction graph discovered while the learning agent is actively exploring its environment. More specifically, we define a subgoal in terms of the bottleneck concept according to which subgoals correspond to those states which are frequently visited along successful trajectories (Stolle & Precup, 2002) or allow for the efficient navigation through the state space (Wolfe & Barto, 2005).

This ability to integrate and re-use knowledge in a lifelong learning setting have been omitted in some of the approaches proposed for this problem. For example, (Simsek & Barto, 2009) relies on the complete representation of the interaction graph to detect bottleneck states using the betweenness centrality measure. Assuming that most effective solutions for a task might lie in a sparse subgraph of the state space, a complete representation of the interaction graph might be impossible to obtain or unnecessary for discovering good skills.

We present a algorithm meant to operate under a scenario where we monitor the interaction of an RL agent with its environment to incrementally reconstruct part of the interaction graph and extract useful skills from. Our approach is based on a near-linear time label propagation algorithm for detecting community structures in networks (Raghavan & Albert, 2007). The meaning of *community* here corresponds to those densely connected regions of a graph linked together by sparse connections (Medus et al., 2005). This definition is thus closely related to idea of *access states* from (Barto, 2002).

2. Related Work

The idea of *bottleneck* regions of the interaction graph has been instrumental in the development of most of the automatic skills discovery techniques. The canonical example of bottlenecks can be found in a navigation task within a multiple-rooms domain and where subgoals naturally corresponds to the doorways connecting two regions of the state space. Bottlenecks have been referred to as those states which appear frequently on successful trajectories to a goal but not on unsuccessful ones (Mcgovern & Barto, 2001; Stolle & Precup, 2002) or as nodes which allow for densely connected regions of the interaction graph to reach other such regions (Menache et al., 2002; Wolfe & Barto, 2005; Kazemitabar & Beigy, 2009). To a lesser extent, (Barto, 2002) could be belong to the latter class of approaches by its definition of *access states* corresponding to those nodes which allow for reaching difficult regions of the interaction graph from other ones.

Bottleneck states are found using graph centrality measures in (Simsek & Barto, 2009; Rad & Hasler, 2010) and capture most of the properties outlined above by defining important nodes as those which lie frequently on shortest paths between states.

It appears that all of these approaches can also be categorized on the basis of whether the total reward obtained on a trajectory is taken into account in the identification of useful subgoals. It could be said that solutions which consider only the structural properties of the interaction graph assume that the state-space constrains the solution space in such a way that one can act as a proxy for the other. The other class of approaches treats the two entities separately, accounting for the fact that bottlenecks of the state-space might not necessarily correspond to desirable subgoals to develop for a task.

The idea of sampling trajectories while the agent is interacting with the environment is central to most of them. A drawback of this class of methods is that primitive actions might not suffice to attain the task goal or yield to successful (defined in a domain-dependent way) trajectories. Since the complete state space does not have to be represented, they might however be more efficient. Furthermore, we can easily conceive that useful solutions might only lie within sparse subgraphs induced by the entire interaction graph: a similar intuition behind manifold embedding techniques of machine learning.

(Mcgovern & Barto, 2001) formulates the problem of finding subgoals as a multiple-instance learning problem over bag of feature vectors collected by interacting with the environment. Two sets of bags are obtained in this way: positive bags made of the observations collected along successful trajectories and the negatives from the unsuccessful ones. The notion of diverse density is then applied either by exhaustive search or gradient descent to find regions of the feature space with the most positive instances and the least negative ones. Because negative bags must only contain unsuccessful observations (posive bag only have to contain at least one successful feature vector), this method is sensible to noise. The problem of collecting successful trajectories would also arise here due if the agent is unable to reach satisfactory performances for the task.

(Barto, 2002) identify subgoals by looking for *access states* which lead to the regions of the state space which have not been visited recently and trying to capture in this way the notion of *relative novelty* or what we also qualify as *breakthrough* in the solution space. The authors posit that the distribution of their measure of relative novelty for subgoals differs significantly from the other kind of states. The time frame within which to search for such novelty events is a parameter of this algorithm and might can lead to different results.

The graph-cut algorithm proposed in (Menache et al., 2002) effectively tries to identify access states of those densely connected regions linked together via a few important edges. (Wolfe & Barto, 2005) essentially applies the same graph-cut partitioning but only over local transition graphs obtained from the sampled trajectories. (Kazemitabar & Beigy, 2009) takes a similar approach by finding strongly connected components (SCC) of the interaction graph. Since most MDP problems that we are interesting in exhibit ergodicity, we are doubtful in the applicability of the latter algorithm. For example, a non-ergodic Markov chain could be one where transition through a door is only allowed in one direction: once it has been traversed the agent cannot go back to the previous room. Such environments are certainly not the most commonly encountered ones. Ergodic chains would then only lead to large components being detected by SCC.

By representing the entire interaction graph, (Simsek & Barto, 2009) compute betweenness centrality to identify important subgoals. They also propose a potential incremental formulation of their algorithm that finds local maxima of betweenness within the subgraph obtained by collecting short trajectories. (Rad & Hasler, 2010) argue that the centrality measured called *connection graph stability* leads to better subgoals identification than betweenness or closeness centrality.

3. Background

Reinforcement Learning (RL) is an approach to machine learning which considers the process of acquiring knowledge through the interaction of an agent with its environment. In a tradeoff between the exploitation of prior knowledge and the exploration of possible new solutions, an RL agent seeks to maximize the total reward returned by the environment in response to the actions taken upon it.

3.1. Markov Decision Processes

Markov Decision Processes (MDP) form the theoretical foundations for this problem. We define an MDP as a tuple $\langle S, A, P, R, \gamma \rangle$ that comprises of a set of states S, primitive actions A, transition probabilities $P : S \times A \times, S \mapsto [0, 1]$, expected reward function $R : S \times A \mapsto \mathbb{R}$ and discount factor $\gamma \in [0, 1]$. At every discrete time step, an MDP agent decides which action a_t to take at the current time t, obtains a reward r_t from the environment and moves to next state $\prime s_t = s_{t+1}$. The learning task for an RL agent is to learn a policy $\pi : S \mapsto A$ that dictates the choice of action under any given state so as to maximize the total reward. We define the action-value function $Q^\pi(s, a)$ as the expected discounted reward for choosing action a_t in state s_t and later following policy π.

$$Q^\pi(s, a) = \mathbb{E}(r_{t+1} + \gamma r_{t+2} + ...|s_t = s, a_t = a) \quad (1)$$

There exists an optimal action-value function for every MDP and is defined by the Bellman equation:

$$Q^*(s, a) = R(s, a) + \gamma \sum_{s\prime \in S} P(s\prime|s, a) \max_{b \in A} Q^*(s\prime, b) \quad (2)$$

A widely used algorithm for learning an action-value function is Q-Learning (Watkins, 1989) which has been shown to converge to the optimal action-value function when actions are sampled in every discrete state (Dayan, 1992). Q-Learning is incremental dynamic programming algorithm that successively improves the estimate on the action-value function by computing:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{\alpha\prime \in A} Q(s_{t+1}, a\prime) - Q(s_t, a_t)) \quad (3)$$

Where α is the learning rate and γ is the discount factor which determines to which extent the agent should strive for long term reward or only act opportunistically (values close to 0).

3.2. Options Framework

Options are temporally-extended actions that capture the idea of a skill, or macro-action, which behave somehow like subroutines that the RL agent can invoke under some specified conditions. It has been shown that options defined over MDPs lead to an an abstraction which coincides exactly with the class of decision problems known as Semi-Markov Decision Processes (SMDP). SMDP extend the mechanistic principles of MDP by allowing actions to last for some varying amount of time depending on the current state.

An option o consists of a tuple $\langle \mathcal{I}, \pi, \beta \rangle$ where the *initiation set* $\mathcal{I} \subseteq \mathcal{S}$ specifies under which states the option can be invoked, $\pi : \mathcal{S} \mapsto \mathcal{O}$ is a subpolicy defined for o and $\beta : \mathcal{S} \mapsto [0, 1]$ returns the probability of terminating the execution of π under any given state. The action set can now be made up of both primitive actions or options. In fact, a primitive action a can be redefined in terms of an option by setting β in such a way that termination is guaranted after only one step $\beta(s) = 1 \ \forall s \in \mathcal{S}$, making the corresponding option available everywhere a can be invoked $\mathcal{I} = \{s : a \in \mathcal{A}_s\}$ and finally having an option policy which selects a for every state $\pi(s) = a \ \forall s \in \mathcal{S}$

An SMDP can thus be defined in terms of the options framework by considering the tuple $\langle \mathcal{S}, \mathcal{O}, \gamma, R, P \rangle$, where $\mathcal{S}$ is the state-space, $\mathcal{O}$ is a set of options, $\gamma \in [0, 1]$ a discount factor, $R : \mathcal{S} \times \mathcal{O} \mapsto \mathbb{R}$ and transition probabilities $P(s\prime|s, o)$

As a generalization to the action-value function for MDP, an option-value function can be defined:

$$Q^\mu(s, o) = \mathbb{E}\{r_{t+1} + \gamma r_{t+2} \gamma^2 r_{t+3} + ...|\mathbb{E}(o\mu, s, t)\} \quad (4)$$

In equation 4, $o\mu$ corresponds to the SMDP which chooses option o, executes its subpolicy π until termination, and reinstalls its main behavior policy μ. Q-Learning can also be extended to options and the options-value function is learnt incrementally with the following update rule:

$$Q(s, o) = Q(s, o) + \alpha(r + \gamma^k \max_{a \in \mathcal{O}} Q(s\prime, a) - Q(s, o)) \quad (5)$$

Q-Learning is only evaluated after the option has terminated. In contrast, Intra-option learning (Sutton et al., 1998) can take place throughout the execution of an option. It leads to more efficient use of experience which can also be shared simultaneously among other options that are consistent with it.

4. Skills Discovery

We base our definition of subgoals on the intuitive notions of access states from (Barto, 2002; Wolfe & Barto, 2005). Such states are found to mediate access to other regions of the graph. More precisely, we posit that they lie on paths connecting densely connected regions with sparse connections between them. We present an incremental skill discovery algorithm based on label propagation (LPA) for detecting community structures. An overview of the skills detection algorithms based on centrality measures is also presented for comparison purposes with LPA later in this paper.

4.1. Community Structures

Communities are groups of nodes (also known as *modules*) with dense connections within themselves but sparse links to the other groups. The algorithms presented in (Newman & Girvan, 2004) iteratively remove edges based on their measure of *betweenness* defined as the relative number of all the shortest paths going to through a given edge. Edges with the highest values are removed first and betweenness is recomputed after every deletion.

The notion of *modularity* is introduced (Newman & Girvan, 2004) to measure the quality of the current partitioning of a network into communities and serve as termination condition. Modularity is defined as the difference between the fractions of the edges belong to communities and their expected number in a network which would have the same community partitioning but with random edges assignment. Hence, a modularity value of $Q = 0$ corresponds to a community partitioning which does not differ from random networks whereas at the other end of the spectrum, $Q = 1$ indicates the strongest presence of community structures. While splitting the network into smaller communities, sudden increase in the modularity will be considered as an indicator that an optima might have been attained. Simulated Annealing (SA) was applied in (Medus et al., 2005) as a global optimization method to find such optima. Even with Brandes' algorithm(Brandes, 2001) for computing betweenness centrality in $\mathcal{O}(nm)$, this community detection technique remains prohibitively expensive.

4.2. Label Propagation Algorithm (LPA)

(Raghavan & Albert, 2007) introduced a near-linear time algorithm for finding community structures that does not rely on any prior knowledge about the networks nor any definition of an objective function for global optimization purposes. This algorithm assigns unique initial community labels to every nodes and at each iteration neighboring nodes try to reach a consensus on their labels. LPA can be summarized in the following steps:

1. Assign an initial unique label to every vertices in the graph G
2. Arrange the vertices in a random sequence X
3. For every vertex $v \in X$, take the label which appears the most frequently in the neighborhood of v and break ties uniformly at random.
4. Repeat steps 2 and 3 as long as there remain vertices with non-maximal labels

Under certain graph structures such as bipartite graphs, LPA might not not converge to a stable solution and a termination condition would have to be defined arbitrarily. We will not handle this issue in this paper but rather concentrate on the problem of updating the labelling efficiently. By collecting trajectories from the RL agent, the resulting interaction graph will be continuously altered: edges will be added and other will be removed if deemed unnecessary. We would thus want to update the current labelling online in an incremental fashion. (Pang et al., 2009) extends LPA by labelling nodes with a pair that consists of a time and a *label* components. It then proceeds to local updates in a similar way to LPA:

1. Given a new edge, label its endpoints u and v as (t, u) and (t, v) where t is the current time
2. Add u and v to a local set X_l together with their neighbors
3. Shuffle X_l and apply the same label assignment rule as LPA except that ties are resolved based on the label with the **largest** t.
4. Add every nodes that have changed their label into a new set X_l.
5. Repeat steps 2 to 4 until a consensus is reached

Updating the labels after an edge has been added is $\mathcal{O}(m_l + n_l)$ where m_l is the number of vertices in the set X_l and n_l, the edges.

4.3. Using Weight Information

While collecting trajectories, it can be useful to set the weight on an edge as the number of times a transition has been experienced over it. This allows us not only

to prune the interaction graph from possibly irrelevant edges (and thus lower the computation burden) but also to bias the label propagation algorithm towards those nodes with the highest importance in terms of weight. We have modified the label selection criterion of LPA such that the label occurring with the most cumulative weight among neighbors is the one that gets assigned.

4.4. Finding Subgoals within Communities

The set of states within a community naturally fits the idea of the initiation set component in the options framework. It is however less obvious to come up with a precise definition of which node within a community should be developed as a subgoal. We used a similar approach as in (Kazemitabar & Beigy, 2009) and try to find those *edge points* that have at least one outgoing edge to another community of a significant size. In the aforementioned paper, the authors make the assumption that the distribution of community sizes is normally distributed and only consider those which are least $Pr(X < \mu + 2\sigma) \approx 0.95$. Since a community can also have multiple such edge points, we decided to only use the one with the highest weight to another community.

4.5. Centrality Measures

For comparison with our LPA algorithm, we also used some graph centrality techniques for options detection (Simsek & Barto, 2009; Rad & Hasler, 2010)

$$C_B(v) = \sum_{s \neq t \neq} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{6}$$

$$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v,t)} \tag{7}$$

Betweenness centrality (eq. 6) is defined as the fraction of the shortest paths σ_{st} between any possible pair passing through a vertex v. Naive computation of betweenness centrality would be $\Theta(n^3)$ but (Brandes, 2001) showed that the complexity be can reduced to $\mathcal{O}(nm)$ or $\mathcal{O}(nm + n^2 \log n)$ in the unweigthed and weighted case respectively. As for closeness centrality (eq. 7), it corresponds to the inverse of the distances from a given vertex to every other vertices in the graph.

We use the scoring measure of (Rad & Hasler, 2010) in order discover maxima of either betweenness or closeness centrality.

$$S(v) = C(v) \left(\frac{C(v)}{\max_{u \in N(v)}(C(v)} \right)^2 \tag{8}$$

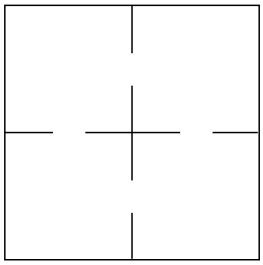

Figure 1. The four-rooms domain that we use has four 5×5 grid rooms connected together using four doors. The corresponding state space for this problem has 103 states.

In equation 8, $C(v)$ is either the betweenness or closeness centrality measures and $N(v)$ is the set of neighbors for node v. We set a threshold on the minimum score that a node must achieve comparatively to the scores computed for the other nodes. For example, we might require that the score should be 97 percent larger than for any other node.

5. Experiment

We implemented the incremental label propagation algorithm described in section 4.2 to find the community structures present in the interaction graph obtained from the trajectories experienced by an RL agent. We tested our algorithm in a grid world consisting of four-rooms in which the agent can navigate through via doorways but by avoiding collisions with unpenetrable walls (figure 5).

The graph discovery procedure that we then applied over this domain was the following:

1. Collect trajectories for 100 episodes for a primitive Q-Learning agent
2. Build the transition graph
3. Prune edges with weight less than th
4. Run weighted label propagation
5. Identify subgoals
6. Create options

We used our automatic options detection algorithm to find potential subgoals in the interaction graph obtained from a hundred trajectories (figure 5). Q-Learning for options (eq. 5) was then executed for another hundred episodes in order to learn the option's subpolicy. The initiation set $\mathcal{I}$ for the options was defined as those states within a community and the termination probabilities specified by β was set to 1 at

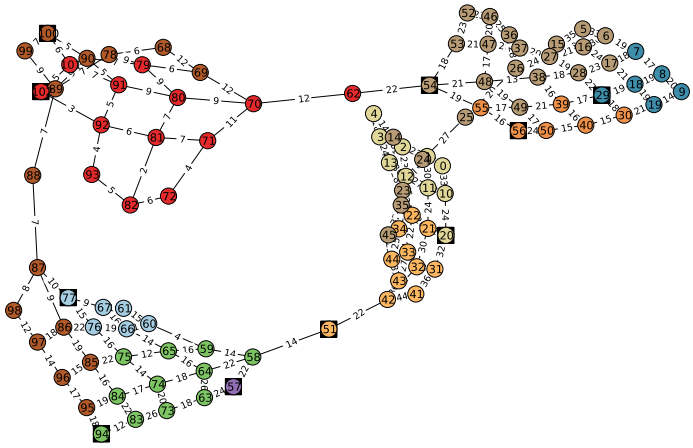

Figure 2. Nine communities were detected by LPA when the subgoals (highlighted in black) detection threshold was set to 0.68 (one point of standard deviation) and edges with only one transition were removed. A spring layout was used to draw the graph.

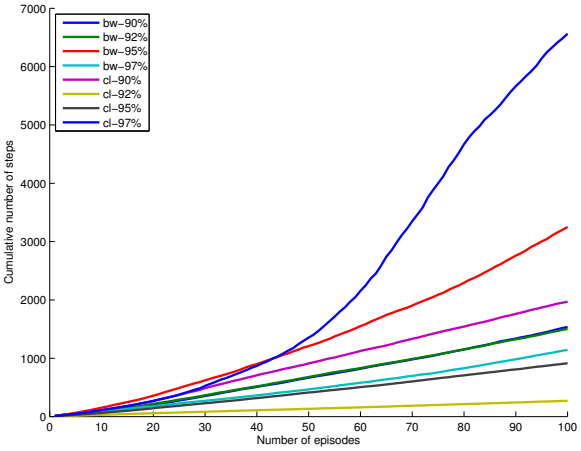

Figure 3. Cummulative number of steps as a function of the number of episodes for RL agents with behavior policies based on betweenness and closeness centrality. The set of options obtained by selecting only the subgoals corresponding to maxima of closeness centrality under a threshold of 0.92 seems to produce the most efficient behavior; the worst is obtained for betweenness at 0.90.

the subgoal and 0 everywhere else. We compared the performance of our LPA algorithm against other skill discovery methods based on betweenness and closeness centrality. Skills based on centrality methods were selected using different thresholds: 0.90, 0.92, 0.95, and 0.97. The performance of 1000 independent agents for each set of options was averaged over 100 episodes.

As shown in figure 4, it seems that the options discovered by our LPA algorithm are useful enough to reduce the number of steps needed to attain the goal compared to the performance of an agent using only a flat policy over primitive actions. However, our algorithm is outperformed by the centrality-based approaches. We think that this could be explained by the simple nature of the problem domain and for which the 9 communities detected by LPA might lead to an overly complicated solution.

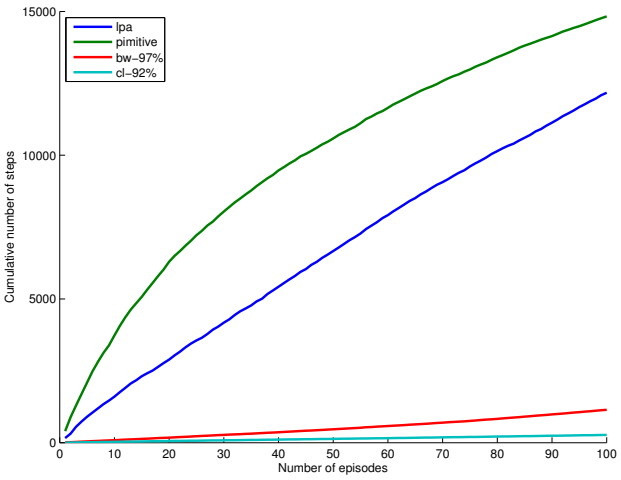

Figure 4. Our label propagation algorithm for skills discovery leads to shorter trajectories to the goal than a primitive agent but is outperformed by other approaches based on betweenness or closeness centrality.

6. Future Work

The community detection approach to skills detection presented in this paper has the advantage of providing a clear definition for what the initiation set for an option should be: any member of a community should also belong to the initiation set of the corresponding option. On the other hand, the question of identifying subgoals within those regions has a less definite answer. We think an important reason which explain the observed results has to do with the heuristic that we designed for selecting a subgoal within the multiple edge points of a community. This definition might

have to be revised in future work. Local computation of betweenness centrality might also be a possible avenue to explore.

As expected because of its stochastic nature, we have observed instability in the solutions returned by LPA. The extension to label propagation proposed in (Cordasco & Gargano, 2010) shows that by using graph coloring for determining the order of label updates, stability and convergence can be guaranteed. If it could be formulated incrementally, semi-synchronous label propagation could be a useful improvement for our application.

The four-rooms domain was used to establish the performance of our algorithm against other skills detection techniques. We think it would be crucial to investigate more complex problem domains and that another performance profile might be revealed in this way. It is reasonable to believe as well that an autonomous skill detection system could make use of a combination of different subgoal identification techniques, scaling up with the state space complexity.

Finally, an important critique of our approach, but also to the broader class to which it belongs to, is that we make use of only the structural properties of the state space and avoiding those of the solution space for finding subgoals. A stronger approach should also take the reward obtained along the collected trajectories in its choice of subgoals.

References

Barto, Andrew G. Using Relative Novelty to Identify Useful Temporal Abstractions in Reinforcement Learning. *Learning*, 2002.

Brandes, Ulrik. A faster algorithm for betweenness centrality*. *Journal of Mathematical Sociology*, 25(1994):163–177, 2001. URL http://www.tandfonline.com/doi/abs/10.1080/0022250X.2001.9990249.

Cordasco, G. and Gargano, L. Community detection via semi-synchronous label propagation algorithms. In *Business Applications of Social Network Analysis (BASNA), 2010 IEEE International Workshop on*, pp. 1 –8, 2010. ISBN 9781424490004. doi: 10.1109/BASNA.2010.5730298. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5730298.

Dayan, Peter. Technical Note Q , -Learning. *Machine Learning*, 292(3):279–292, 1992. ISSN 08856125. doi: 10.1080/07900629208722557. URL http://www.springerlink.com/index/t774414027552160.pdf.

Kazemitabar, Seyed Jalal and Beigy, Hamid. Using Strongly Connected Components as a Basis for Autonomous Skill Acquisition in. In *Proceedings of the 6th International Symposium on Neural Networks on Advances in Neural Networks*, pp. 794–803, Wuhan, China, 2009.

Mcgovern, Amy and Barto, Andrew G. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. *Machine Learning*, 2001.

Medus, a., Acuña, G., and Dorso, C.O. Detection of community structures in networks via global optimization. *Physica A: Statistical Mechanics and its Applications*, 358(2-4):593–604, December 2005. ISSN 03784371. doi: 10.1016/j.physa.2005.04.022. URL http://linkinghub.elsevier.com/retrieve/pii/S0378437105003973.

Menache, Ishai, Mannor, Shie, and Shimkin, Nahum. Q-Cut - Dynamic Discovery of Sub-Goals in Reinforcement Learning. In *Proceedings of the 13th European Conference on Machine Learning*, pp. 295—-306, London, UK, 2002. Springer-Verlag.

Newman, M. and Girvan, M. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):1–15, February 2004. ISSN 1539-3755. doi: 10.1103/PhysRevE.69.026113. URL http://link.aps.org/doi/10.1103/PhysRevE.69.026113.

Pang, Sheng, Chen, Changjia, and Wei, Ting. A Realtime Clique Detection Algorithm: Time-Based Incremental Label Propagation. *2009 Third International Symposium on Intelligent Information Technology Application*, pp. 459–462, 2009. doi: 10.1109/IITA.2009.394. URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5370363.

Rad, Ali Ajdari and Hasler, Martin. Automatic Skill Acquisition in Reinforcement Learning using Connection Graph Stability Centrality. *Sciences-New York*, pp. 697–700, 2010.

Raghavan, UN and Albert, R. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, pp. 1–12, 2007. URL http://pre.aps.org/abstract/PRE/v76/i3/e036106.

Simsek, Ozgur and Barto, Andrew G. Skill characterization based on betweenness. In *In Advances in Neural Information Processing Systems 22*, pp. 1–8, 2009. URL http://www-all.cs.umass.edu/pubs/2008/simsek_b_NIPS08.pdf.

Stolle, Martin and Precup, Doina. Learning options in reinforcement learning. *Abstraction, Reformulation, and Approximation*, pp. 212–223, 2002. URL http://www.springerlink.com/index/8293N8MGE4N0DPGV.pdf.

Sutton, Richard S, Precup, Doina, and Singh, Satinder. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999. ISSN 00043702. doi: 10.1016/S0004-3702(99)00052-1. URL http://linkinghub.elsevier.com/retrieve/pii/S0004370299000521.

Sutton, RS S, Precup, D, and Singh, S. Intra-option learning about temporally abstract actions. In *Fifteenth International Conference on Machine Learning*, pp. 556–564. Morgan Kaufman, 1998. ISBN 1558605568. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.7134&rep=rep1&type=pdfhttp://www.cs.umich.edu/~baveja/Papers/ICML98_SPS.pdf.

Watkins, C. *Learning from Delayed Rewards*. Phd thesis, Cambridge University, England, 1989.

Wolfe, Alicia P. and Barto, Andrew G. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *In Proceedings of the Twenty-Second International Conference on Machine Learning*, pp. 816–823, 2005.