Betweenness Centrality as a Basis for Forming Skills

Özgür Şimşek
Andrew G. Barto
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-9264
{ozgur|barto}@cs.umass.edu

April 12, 2007

University of Massachusetts
Department of Computer Science
Technical Report 07-26

Abstract

We show that *betweenness centrality*, a graph-theoretic measure widely used in social network analysis, provides a sound basis for autonomously forming useful high-level behaviors, or *skills*, from available *primitives*—the smallest behavioral units available to an autonomous agent.

1 Introduction

We consider the problem of how to form useful high-level behaviors, or *skills*, from available *primitives*—the smallest behavioral units that are available to an autonomous agent. A suitable set of skills can greatly improve an agent's efficiency in learning to solve difficult problems. The ability to form such skills autonomously is an essential component of artificial agents that can solve difficult real-world problems without relying on problem-specific human design effort.

In approaching this problem, we distinguish between two related questions: What constitutes a useful skill? And, how can an agent acquire such skills autonomously? In this paper, we address the former question with the objective of guiding the research for answering the latter. Our main contribution is a

characterization of a useful class of skills using the graph-theoretic measure *betweenness centrality*.

Many of the existing techniques for autonomous skill formation identifies states that are useful to reach and defines skills for efficiently reaching these states [7, 5]. This is also the approach we take here. The main distinction of our work from earlier methods is in how we define what constitutes a useful state to reach. We define the utility of a given state as a subgoal using a graphical representation of the problem and a measure of how prominent the node is on certain shortest paths on this graph. Although several recent approaches have utilized graphical approaches to skill formation [6, 8, 10], the approach we propose here is fundamentally different from these earlier methods, which are based on graph cut and clustering techniques.

Our approach may be used to form a set of skills suitable for a given problem if the graphical representation of the problem is readily available. But, perhaps more importantly, our work may form a useful guide for developing techniques that do not rely on explicitly or completely representing this graph.

2 Background

We use the MDP framework to represent the agent's interaction with its environment. A finite MDP is a tuple $\langle S, A, T, R, D, \gamma \rangle$, where S is a finite set of states, A is a finite state of actions, $T : S \times A \times S \to [0, 1]$ is a transition function, $R : S \times A \times S \to \Re$ is a reward function, D is the initial state distribution from which the start state is drawn, and γ is a discount factor, $0 \leq \gamma \leq 1$. At each decision stage, the agent observes a state $s \in S$ and executes an action $a \in A$ with probability $\pi(s, a)$, where $\pi : S \times A \to [0, 1]$ is a stationary stochastic policy. With probability $T(s, a, s')$, the agent observes state s' in the next decision stage and receives an immediate reward with expected value $R(s, a, s')$. The value function of policy π is a map $V^\pi : S \to \Re$ that specifies the expected return for executing π starting from state s, where return is the discounted sum of future rewards. An optimal policy is one that maximizes the value function over all states.

To represent skills, we use the options framework [12, 9]. A (Markov) *option* is a temporally-extended action, specified by a triple $\langle I, \pi, \beta \rangle$, where I denotes the option's initiation set, i.e., the set of states in which the option can be invoked, π denotes the policy followed when the option is executing, and $\beta : I \to [0, 1]$, denotes the option's termination condition, with $\beta(s)$ giving the probability that the option terminates in state $s \in I$.

3 Our Approach

Let $G = (V, E)$ be a directed graph in which V is a set of vertices representing the states of a given MDP and E is a set of edges representing possible state transitions through actions. For s, v, $t \in V$, let σ_{st} be the number of shortest

paths from vertex s to vertex t, and let $\sigma_{st}(v)$ be the number of shortest paths from vertex s to vertex t that pass through vertex v.

As a measure of the utility of state v as a subgoal from state s, we propose to use the following scoring function:

$$c_s(v) = \sum_{\forall t \neq v \neq s} \frac{\sigma_{st}(v)}{\sigma_{st}}. \tag{1}$$

Equation 1 gives the proportion of shortest paths from vertex s to the remaining vertices on the graph that pass through vertex v, which may be considered a measure of how influential vertex v is in efficiently navigating the graph starting from vertex s. The higher the score, the more desirable v is as a subgoal from s.

If we add $c_s(v)$ scores for all $s \in V$, we obtain *betweenness centrality* of vertex v, a measure widely used in social network analysis to determine the importance and influence of nodes in a network [3, 4].

Note that the measure we propose is not betweenness centrality, but the contribution to betweenness centrality from paths that start from a given vertex. Unlike network analysts who are concerned with the network as a whole, we necessarily need to view the graph from the perspective of the individual node for which we are forming the skill.

Using a threshold t on our scoring function gives us a set of subgoals S and an initiation set $I(g)$, $\forall g \in S$:

$$S = \{v \in V \mid \exists s \in V \ \ s.t. \ \ c_s(v) \geq t\} \tag{2}$$

$$I(g) = \{s \in V \mid c_s(g) \geq t\} \tag{3}$$

As in other subgoal-based approaches in the literature, we can then use a given subgoal and the corresponding initiation set to define a skill for efficiently reaching the subgoal state from states in the initiation set using a pseudo-reward function [2].

An important detail is that the evaluation of subgoal candidates are done in the absence of any existing skills. When more than one skill are introduced

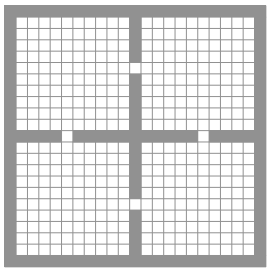

Figure 1: The Rooms domain.

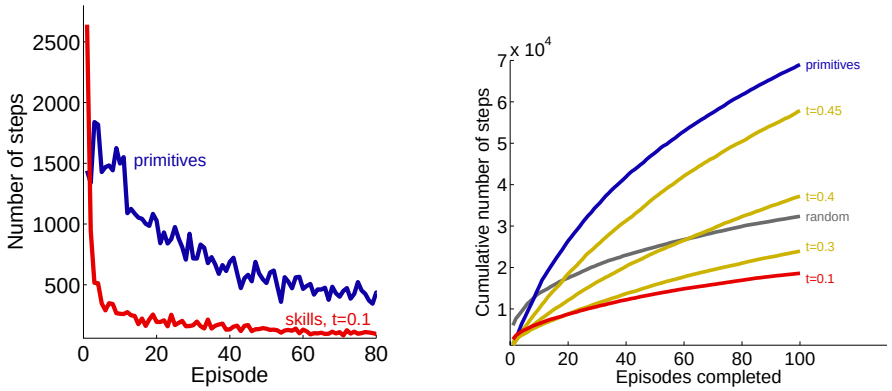

Figure 2: Performance measures in the Rooms domain.

based on this evaluation, some of the skills may prove to be redundant and detrimental. This is particularly a problem because the scoring function we propose assigns similar scores to neighboring states, resulting in subgoals that are very close to each other, causing an agent to incur the cost of additional skills while obtaining none of the benefits. To select a representative set of subgoals from S, we define $p(g)$, *support* for subgoal g:

$$p(g) = \sum_{s \in I(g)} c_s(g), \quad \forall g \in S. \tag{4}$$

To filter the redundant subgoals, we form a subgraph G' consisting of vertices in S and edges among them, identify strongly connected components of this subgraph, and choose as subgoals those states in S who have higher support than their neighbors in the strongly connected component they belong to.

In the next section, we explore what type of subgoals our scoring function identifies in a number of different domains and we experimentally evaluate the utility of corresponding skills.

4 Evaluation

In our experimental analysis, the agent used Q-learning with ϵ-greedy exploration with $\epsilon = 0.1$. When using skills, it performed both intra-option updates and macro-Q updates. The learning rate (α) was kept constant at 0.1. Initial Q-values were 0. Discount rate γ was set to 1 in episodic tasks, to 0.99 in continuous tasks. The options that were generated terminated with probability one at the goal state and outside the initiation set; at all other states they terminated with probability zero. We normalized the scores obtained by using Equation 1

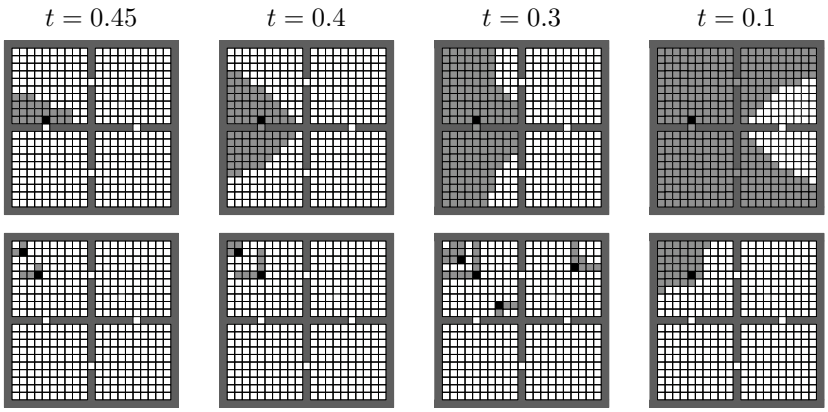

Figure 3: Skills formed in the Rooms domain using different values for threshold t.

by dividing them by $n - 2$, where n is the number of states in the domain, to obtain scores between 0 and 1. We experimented with a wide range of values for the threshold parameter t, ranging from 0.001 to 0.9.

4.1 Rooms

Our first example is a gridworld domain from [12] shown in Figure 1. The available actions at each state are `north`, `south`, `east`, and `west`, which move the agent in the intended direction with probability 0.9 and in a uniform random direction with probability 0.1. If the direction of movement is blocked, the agent remains in the same location.

Figure 3 shows the skills that were formed in this domain for various values of the threshold parameter t. The grid squares with a dark shading are the goal states of the identified skills, while the lighter shading surrounding them designates the corresponding initiation set. Because of the axes of symmetry present in the domain, many of the skills were images of each other. The figure shows only those skills with goal states in the lower triangular section of the northwest room—except when this would introduce ambiguity, in which case we present the corresponding skill in a different room. The remaining set of skills may be deduced from the ones we present.

The columns in the figure correspond to different settings of the threshold parameter t, while the rows correspond to the two types of skills that emerged in this domain. The type with the higher support consists of skills for reaching the eight states that surround the four doorways. The second type consists of skills that lead the agent towards the center of the rooms away from two opposing corners. Interestingly, varying the threshold parameter did not change the essential characteristics of the skills that were formed, influencing mainly the size of the initiation sets.

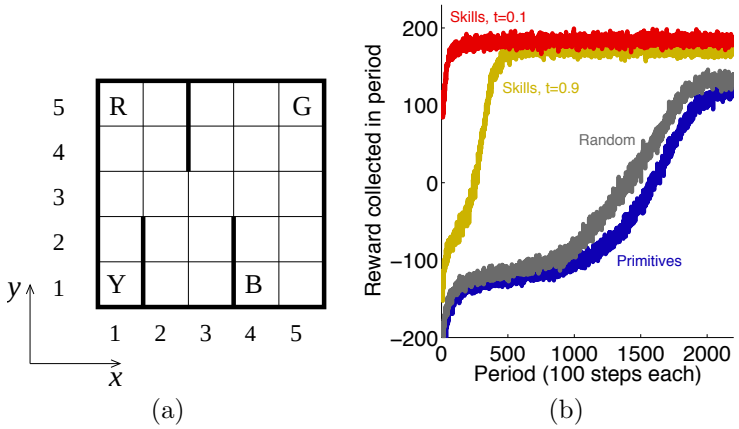

Figure 4: (a) The taxi domain. (b) Performance in the taxi domain.

We explored the utility of these skills in 100 randomly selected episodic tasks in the domain. In each task, the reward was -0.001 for each transition and an additional $+1$ when transitioning into a randomly selected terminal state. Figure 2a shows mean number of steps as a function of episode, comparing an agent using only primitive actions to an agent using both the primitive actions and the skills formed with $t = 0.1$, revealing a big improvement in performance with the skills. Figure 2b displays performance for additional settings of t, showing instead the cumulative number of steps as a function of episodes completed, for easier comparison of the additional cases presented. The figure reveals performance improvements as t values get smaller. As noted above, the main impact of reducing t was to expand the initiation sets, so the figure reveals a sensitivity to the size of the initiation sets. Lowering t beyond 0.1 did not introduce new skills or impact performance.

Figure 2b also shows the performance of an agent that used skills with randomly selected subgoals. This setting can be directly compared to only the $t = 0.1$ setting because the number of subgoals used and the size of their initiation sets were matched to the values observed in this particular case. The figure shows that while random skills did not perform as well as the skills formed by our method, they improved performance remarkably compared to the primitives-only setting.

4.2 Taxi

Our next example is the taxi domain [2] which consists of a taxi and a passenger on a 5×5 grid depicted in Figure 4a. Initially, the taxi is at a random grid location while a passenger is waiting to be taken to her destination. At each grid location, the taxi has six primitive actions: `north`, `east`, `south`, `west`,

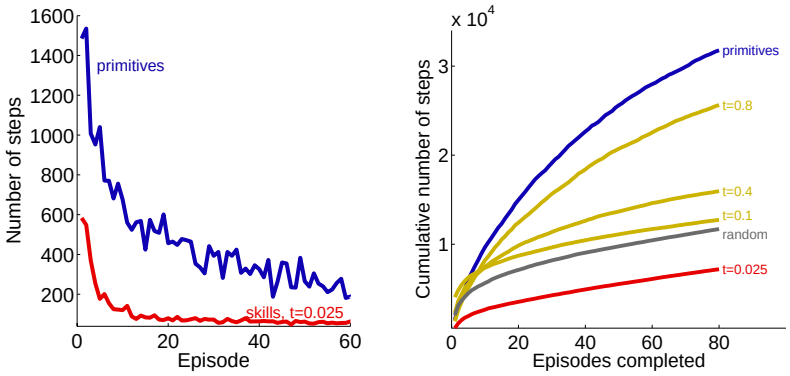

Figure 5: Performance in the small playroom domain.

`pick-up`, and `put-down`. The navigation actions succeed in moving the taxi in the intended direction with probability 0.80; with probability 0.20, the action takes the taxi to the right or left of the intended direction. If the direction of movement is blocked, the taxi remains in the same location. The action `pick-up` places the passenger in the taxi if the taxi is at the same grid location as the passenger; otherwise it has no effect. Similarly, `put-down` delivers the passenger if the passenger is inside the taxi and the taxi is at the destination; otherwise it has no effect. We consider a continuing version of the problem, in which, simultaneously with successful delivery of a passenger, a new passenger appears on the grid. The source and destination of all passengers are chosen uniformly at random from among the four grid squares marked with R, G, B, Y. Reward is -1 for each action, an additional $+50$ for passenger delivery, and an additional -10 for an unsuccessful `pick-up` or `put-down` action.

With $t = 0.1$ setting, our method formed thirty skills in this domain. These fall into one of the following categories: (1) Skills for taking the passenger to her destination, (2) Skills for dropping off the passenger, (3) Skills for picking up the passenger from her location, (4) Skills for efficiently reaching grid squares (2, 3) and (3, 3), which act as bottlenecks that restrict navigation, although the grid is quite small. Figure 7 shows the goal states of these skills as filled vertices on the state transition graph of the domain. As in the Rooms domain, increasing values of t did not lead to different skills, but mainly reduced the size of their initiation set.

Figure 4b shows the utility of the formed skills on the problem. Compared to the primitives-only baseline, the skills improved performance remarkably. The *random* condition, which was again matched to the $t = 0.1$ setting, did not perform as well in this domain as in the Rooms domain.

4.3 Playroom

We next consider a variant of the playroom domain [1, 11], in which an agent interacts with a number of objects in a room: a light switch, a ball, a bell, a button for turning music on and off, and a toy monkey. The agent has an eye and a hand and can perform the following actions: (1) direct its eye to the object at its hand, (2) hold the object it is looking at, (3) move the object in its hand to the location it is looking at, (4) flip the light switch, (5) press the play button, and (6) ring the bell. The first two are reliable actions that the agent can perform with certainty. The remaining actions are stochastic and succeed with probability 0.75. In order to operate on an object, both the eye and hand must be on the object. In addition, to be able to press the music button, the light should be on. The toy monkey starts making frightened sounds if the bell is rung while the music is playing and stops only when the music is turned off.

The state transition graph of this domain is shown in Figure 8. The state representation we used included the object at hand, the object at eye, and the state of the music (on/off), the lighting (on/off), the monkey (frightened/not), and the bell(ringing/not). The graph was drawn using a force-directed algorithm that models the edges as springs and minimizes the force on the system. The different clusters reflect different settings of the values of music, light, and noise variables (not all combinations are possible). The shaded nodes on the graph are the ones that were identified as subgoals with $t = 0.025$. They are states that allow transitions between different clusters or those that facilitate navigation within clusters. More specifically, the subgoals seen in the cluster centers are those that have the eye and hand on the same object, which is necessary to manipulate the object. Again, higher or lower values of the threshold parameter t did not essentially change the type of the skills that were formed, but influenced the size of the initiation sets. Figure 5 shows the performance of an agent using these skills on 100 randomly-selected episodic tasks. The skills in the *random* condition was matched with the skills formed at $t = 0.025$ condition.

We also experimented with a larger version of the playroom domain, in which the agent can place a marker on objects, introducing an additional state variable, and, in which the action effects are slightly more involved, with the marker position also playing a role in some of the actions. Figure 9 shows the subgoals identified on this domain as shaded nodes on the state transition graph. Figure 6 shows performance on 100 randomly-selected episodic tasks. The skills in the *random* condition was matched with the skills formed at $t = 0.005$ condition. The results in both versions of the playroom were consistent with earlier findings.

5 Discussion

Our analysis shows that the scoring metric we propose is effective in identifying states that are useful to reach. Our technique relies on the availability of the state transition graph of the domain and therefore is not always directly usable for forming skills, but it may prove to be a useful guide for developing techniques

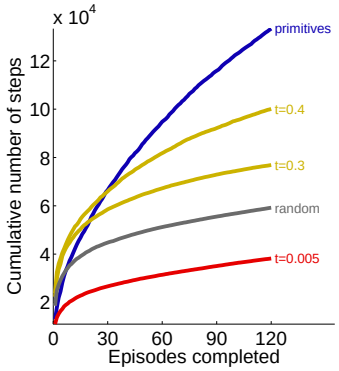

Figure 6: Performance in the large playroom.

for forming the same type of skills without directly or explicitly representing the transition graph. Two existing approaches may provide a useful starting point. The first is the approach by McGovern & Barto [7] which relies on analysis of state trajectories to identify states that are prominent in successful trajectories, but not on unsuccessful ones. Although their method takes into account all paths rather than a small number of desirable paths, it may form a basis for developing a similar approach that focuses on short paths. The second approach is the method by Şimşek, Wolfe & Barto [10] which partitions state transition graphs constructed from short state trajectories—and therefore generally only include short paths.

The main component of our approach is path analysis, which differentiates it from other graphical approaches to skill discovery. The main advantage of the path-based approach is that it makes it possible to disregard a large portion of the graph, in other words, to extract from the graph the parts that are most directly relevant for the task at hand. This is not as easily accomplished when using graph cut and clustering techniques, except when working with partial graphs as in the method proposed by Şimşek, Wolfe & Barto [10].

The approach we presented may be refined in a number of ways. First, while the analysis of shortest paths at the exclusion of all others has proved useful, a binary classification of paths is simplistic and methods that consider a broader set of paths may improve the quality of the skills that are formed. In addition, action effects may be more accurately represented on the state transition graph. In the analysis we presented here, we used identical edge weights, which may be a large departure from the actual state transition dynamics. And third, a weight function on vertices may be used in Equation 1, allowing an agent to take into consideration its estimate of the reward distribution in the domain when forming skills.

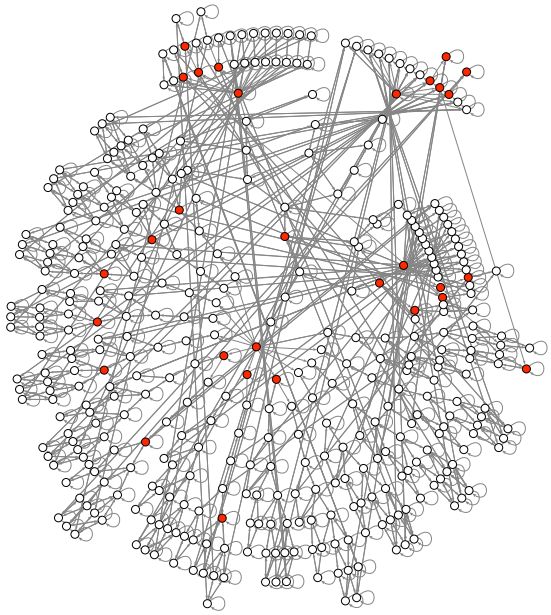

Figure 7: The state transition graph of the taxi domain, showing identified subgoals as shaded nodes.

Acknowledgments

This research was supported by the National Science Foundation under Grant No.CCF-0432143. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Andrew G. Barto, Satinder Singh, and N. Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *International Conference on Developmental Learning (ICDL)*, 2004.

[2] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[3] L. C. Freeman. A set of measures of centrality based upon betweenness. *Sociometry*, 40:35–41, 1977.

[4] L. C. Freeman. Centrality in social networks: Conceptual clarification. *Social Networks*, 1:215–239, 1979.

[5] Bernhard Hengst. Discovering hierarchy in reinforcement learning with HEXQ. In Claude Sammut and Achim G. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 243–250. Morgan Kaufmann, 2002.

[6] Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the Twenty-First International Conference on Machine Learning*, pages 560–567. ACM Press, 2004.

[7] Amy McGovern and Andrew G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 361–368. Morgan Kaufmann, 2001.

[8] Ishai Menache, Shie Mannor, and Nahum Shimkin. Q-Cut - Dynamic discovery of sub-goals in reinforcement learning. In Tapio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Proceedings of the Thirteenth European Conference on Machine Learning*, volume 2430 of *Lecture Notes in Computer Science*, pages 295–306. Springer, 2002.

[9] Doina Precup. *Temporal abstraction in reinforcement learning*. PhD thesis, University of Massachusetts Amherst, 2000.

[10] Özgür Şimşek, Alicia P. Wolfe, and Andrew G. Barto. Local graph partitioning as a basis for generating temporally-extended actions in reinforcement learning. In *AAAI Workshop Proceedings*, 2004.

[11] Satinder Singh, Andrew G. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *18th Annual Conference on Neural Information Processing Systems (NIPS)*, 2005.

[12] Richard S. Sutton, Doina Precup, and Satinder P. Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

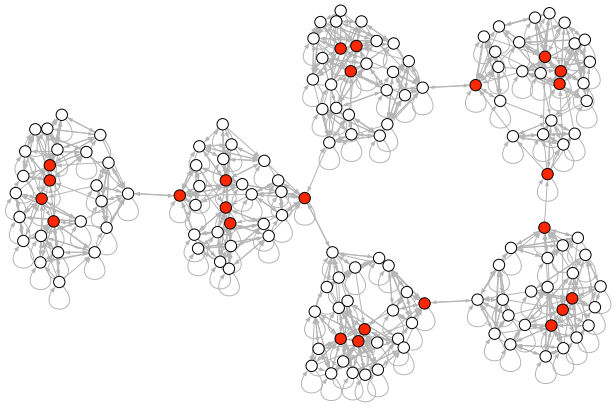

Figure 8: The state transition graph of the small playroom, showing identified subgoals as shaded nodes.

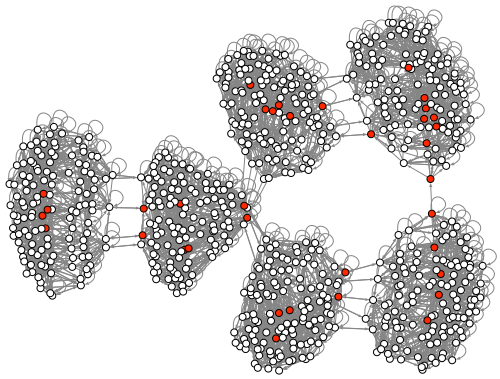

Figure 9: The state transition graph of the large playroom, showing identified subgoals as shaded nodes.