2004

Using Relative Novelty to Identify Useful Temporal Abstractions in Reinforcement Learning

Özgür Şimşek
University of Massachusetts - Amherst

Andrew G. Barto
University of Massachusetts - Amherst

Using Relative Novelty to Identify Useful Temporal Abstractions in Reinforcement Learning

Özgür Şimşek OZGUR@CS.UMASS.EDU
Andrew G. Barto BARTO@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01003-9264

Abstract

We present a new method for automatically creating useful temporal abstractions in reinforcement learning. We argue that states that allow the agent to transition to a different region of the state space are useful subgoals, and propose a method for identifying them using the concept of *relative novelty*. When such a state is identified, a temporally-extended activity (e.g., an option) is generated that takes the agent efficiently to this state. We illustrate the utility of the method in a number of tasks.

1. Introduction

Several formalisms have recently been developed by reinforcement learning (RL) researchers that address planning, acting, and learning at multiple levels of temporal abstraction. These include Hierarchies of Abstract Machines (Parr & Russell, 1998; Parr, 1998), MAXQ value function decomposition (Dietterich, 2000), and the options framework (Sutton et al., 1999; Precup, 2000). These formalisms pave the way toward dramatically improved capabilities of autonomous agents, but to fully realize their benefits, an agent needs to be able to create useful temporal abstractions automatically instead of relying on a system designer to provide them.

A number of methods have been suggested for addressing this need. Hengst (2002) proposed constructing a hierarchy of abstractions in problems with factored state spaces. His method orders state variables with respect to their frequency of change and adds a layer of hierarchy for each state variable, where each layer handles a smaller Markov Decision Process (MDP) than the previous layer. Thrun and Schwartz (1995), Pickett and Barto (2002) generate temporal abstractions by finding commonly occurring subpolicies in solutions to a set of tasks. Digney (1998), McGovern and Barto (2001), Menache et al. (2002) take yet a different approach: They identify subgoal states and generate temporally-extended activities that take the agent to these states. Digney's subgoals are states that are visited frequently or that have a high reward gradient. McGovern and Barto's method identifies as subgoals those regions of the state space that the agent visits frequently on successful trajectories but not on unsuccessful ones. Menache et al. define subgoals as the border states of strongly connected areas of the MDP transition graph and find them using a max-flow/min-cut algorithm.

The algorithm we propose also identifies subgoals in the state space and creates temporally-extended activities that take the agent efficiently to these subgoals. Our subgoals are states that allow the agent to transition to a part of the state space that is otherwise unavailable or difficult to reach from its current region. We call them *access states*. Some examples are a doorway between two rooms, an elevator which provides quick access to all floors, and an airport which provides access to various cities through air travel.

While subgoals of this type are abundant in navigational tasks, they are by no means restricted to them. For instance, most sequential tasks require that work on a subtask be completed before the next task can begin, in which case the completion of a subtask provides access to the next subtask. A different example is building a tool, if the tool makes possible a new set of activities for the agent.

These subgoals are useful in two ways. First, they allow more efficient exploration of the state space, by providing more direct access to those regions that the agent does not tend to go to easily (cf. McGovern & Barto, 2001; Menache et al., 2002). Second, they

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004.

allow transfer of knowledge between tasks that are in the same domain but have different reward functions—getting to the doorway is useful regardless of what the agent needs to do in the other room.

In identifying these subgoals we use their defining property—that they allow the agent to transition to a different region in the state space. We detect this transition using the concept of *relative novelty*, a measure of how much short-term novelty a state introduces to the agent. When the algorithm identifies a subgoal, it creates a temporally-extended activity (e.g., an option) that takes the agent efficiently to this state. We call our algorithm the *relative novelty algorithm* (RN) after its method of defining subgoals.

An important property of RN is that it does not make use of the reward function of the overall problem. This has two implications. First, it can be used in a purely exploratory mode, where no overall RL algorithm is executing, or it can be used with any particular RL algorithm. Second, it is not necessary to complete a task in order to create temporal abstractions. For example, if the task is to learn a policy for hitting a goal region in minimum time, abstractions can be created before the goal region is hit for the first time. This, we believe, is a critical property for facilitating RL in large complex tasks.

In the following sections we describe the particulars of our algorithm and argue for its utility by presenting and discussing empirical results on several examples.

2. The Relative Novelty Algorithm

RN is based on our intuition that access states will be more likely than other states to introduce short-term novelty, i.e., to mediate a transition to a region not visited recently. In the following sections, we describe how RN captures this intuition. We first define novelty and relative novelty, then explain how RN identifies subgoals and generates temporally-extended activities that takes the agent to these subgoals. Throughout our discussion, we will refer to access states as *targets*, and to other states as *non-targets*.

2.1. Novelty

Various concepts of novelty play many roles in both cognitive and computational science. The notion of novelty we use is purposefully very simple, but we do have in mind extensions that benefit from richer formalisms than the discrete state problems that we address here.

Our definition of novelty relates it to how frequently a state is visited since a designated start time. We define the novelty of a discrete state s to be equal to $\frac{1}{\sqrt{n_s}}$, where n_s is the number of times it has been visited. The novelty of a set S of states is $\frac{1}{\sqrt{\bar{n}_S}}$, where $\bar{n}_S$ is the mean number of times states in S have been visited. With this definition, the novelty of a state equals 1 when it is first visited, decays with each succeeding visit, and approaches 0 in the limit.

As mentioned above, we are interested in a short-term measure of novelty. One possible approach is to decay the agent's experience using eligibility traces (Sutton & Barto, 1998). We chose instead to reset visitation frequencies periodically—this is simpler, particularly for episodic tasks in which the end of an episode is a natural point to reset frequencies.

2.2. Relative Novelty

We define the relative novelty of a state in a transition sequence to be the ratio of the novelty of states that followed it (including itself) to the novelty of the states that preceded it. The number of forward and backward transitions to take into account in computing this score is a parameter of the algorithm; we call it the *novelty lag* (l_n). A state will typically have a different relative novelty score each time it is visited.

Our intuition suggests that the distribution of relative novelty scores of targets will be different than that of non-targets. More specifically, we expect targets to have higher scores more frequently. We tested this hypothesis in a simple domain, the two-room gridworld shown in figure 2a. The actions were the usual `north`, `south`, `east`, `west`. We ignored the goal state and had the agent perform a 1000-step random walk 1000 times, starting each at a random grid location.

This domain has a single target state that fits our definition of a subgoal—the doorway between the two rooms. Figure 1 shows the distribution of relative novelty scores for the doorway and for other states, using a novelty lag of 7. The figure reveals that the distributions are indeed different for this domain. Both distributions peak around a relative novelty score of 1, indicating approximately equal novelty scores preceding and following a state, but the target distribution has a heavier tail.

We should note here that putting all non-targets into one bin is a simplification. We expect states that are close to the target to behave more like the target, and in general we expect two given states to differ in their distribution of relative novelty scores.

Our repeated experiments with different novelty lags

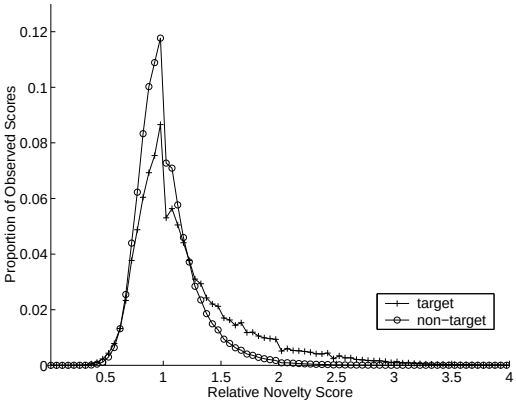

Figure 1. Distribution of relative novelty scores for target and non-target states in the two-room gridworld of Figure 2a.

and room sizes showed a similar discrepancy in relative novelty scores of the doorway and other states. This discrepancy is the basis of our subgoal discovery method, which we discuss in the next section.

2.3. Subgoal Discovery

We formulate the subgoal discovery task for an online RL agent as a classification problem. With each transition, the agent observes a new relative novelty score for some state s and wishes to classify s as target (T) or non-target (N), based on not only the current score, but *all* scores observed so far for s.

If class-conditional relative novelty distributions are known, this classification task is straightforward using Bayesian decision theory (Duda et al., 2001). Assigning an appropriate cost to two possible types of error—classifying a target as non-target (miss) or a non-target as target (false alarm)—and minimizing total cost gives rise to the following decision rule:

Label state as target if

$$\frac{P\{(s_1,..,s_n)|T\}}{P\{(s_1,..,s_n)|N\}} > \frac{\lambda_{fa}}{\lambda_{miss}} \cdot \frac{P\{N\}}{P\{T\}} \qquad (1)$$

where $(s_1,..,s_n)$ are the relative novelty scores observed for the state, $P\{i\}$ is the prior probability of a state of type i, λ_{fa} is the cost assigned to a false alarm, and λ_{miss} is the cost assigned to a miss.

We further simplify this rule by converting the continuous relative novelty score to a binary feature x, where x equals 1 if score is greater than a threshold (which we call the *relative novelty threshold* and denote by t_{RN}) and 0 otherwise. This is motivated by our observation that the distributions differ mainly in their tail, which suggests that the appropriate choice of a threshold would capture enough information to construct a good classifier. Assuming independent observations of relative novelty for a given state, this allows us to rewrite inequality 1 as follows:

$$\frac{p^{n_1}(1-p)^{n-n_1}}{q^{n_1}(1-q)^{n-n_1}} > \frac{\lambda_{fa}}{\lambda_{miss}} \cdot \frac{P\{N\}}{P\{T\}} \qquad (2)$$

where $p = P\{x = 1|T\}$, $q = P\{x = 1|N\}$, n_1 is the number of observations with $x = 1$, and n is the total number of observations.

And finally, using simple algebra, we obtain the following decision rule:

Label state as target if

$$\frac{n_1}{n} > \frac{\ln\frac{1-q}{1-p}}{\ln\frac{p(1-q)}{q(1-p)}} + \frac{1}{n}\frac{\ln(\frac{\lambda_{fa}}{\lambda_{miss}} \cdot \frac{p(N)}{p(T)})}{\ln\frac{p(1-q)}{q(1-p)}} \qquad (3)$$

This rule is a simple threshold on the proportion of observations where $x = 1$, or equivalently, on the proportion of relative novelty scores that are above t_{RN}. The first term on the right is a constant that depends only on class-conditional probabilities of the observations. The second term includes another constant (which depends on class-conditional probabilities, the priors, and the relative cost of each type of error) and a variable (the number of observations); it is inversely related to the number of observations, thus the influence of the second term decreases with increasing sample size.

There remains an important issue to address: The class-conditional relative novelty distributions are typically not known. One approach we propose here is to estimate them using agent's experience in a small part of the actual task, if it is representative of the whole task and if the subgoals in this region are known. If this is not possible, we propose gathering labeled data using a corpus of environments where the subgoal states are known. We suggest that these environments vary in their size, connectivity structure, and number of access states. This can be done only once, and the resulting estimates and parameter settings may be used for RL tasks for which the first approach is infeasible.

Once the class-conditional relative novelty distributions are estimated, the next task is to determine the value of t_{RN}—each setting of this parameter will yield a different classifier. A classical approach to this task is to guide this decision using a receiver operating characteristic (ROC) curve (Duda et al., 2001), which could

be constructed using the data obtained to estimate the relative novelty distributions.

The class-conditional relative novelty distributions and the value of t_{RN} will determine the values of p and q. All that remains before the classifier is ready for use is determining the value of $\frac{\lambda_{fa}}{\lambda_{miss}} \cdot \frac{P\{N\}}{P\{T\}}$, for which we can only provide some guidelines: 1) The prior probability of a target should be much smaller than that of a non-target, and 2) A false positive should have a much higher cost than a miss—subgoals that are chosen arbitrarily do considerable harm by leading the agent to these states until the agent learns that they are not valuable.

In summary, we propose the following procedure for identifying subgoals:

1. (Off-line) Estimate the class-conditional relative novelty distributions using agent's experience, if possible, in a small part of the actual task, and otherwise in a corpus of environments. Determine the value of t_{RN} using a ROC curve analysis. Compute p, q given the class-conditional relative novelty distributions and t_{RN}. Determine $\frac{\lambda_{fa}}{\lambda_{miss}} \cdot \frac{P\{N\}}{P\{T\}}$.

2. (On-line) Evaluate decision rule 3 periodically, possibly with each new state transition, considering *all* of the relative novelty scores observed for a given state.

There are some details to this relatively simple procedure. First, it is essential to periodically reset visitation counts. This is important because the type of novelty we seek is defined *relative to the agent's recent experience*—it is irrelevant whether a state is novel to the agent overall. In an episodic task, a natural way to reset visitation counts is to do it at the beginning of each episode. But it can also be done after a certain number of transitions, which may be beneficial if the episodes are long, or if they vary greatly in length. Second, when computing visitation frequencies, self-transitions (i.e., transitions from a state to itself) should be ignored; otherwise the assumption of independent observations would be severely violated, as two consecutive scores for the same state will be highly correlated. And finally, there is a time lag between the actual state visitation and the relative novelty computations because of the novelty lag.

2.4. Generating Temporal Abstractions

In defining temporal abstractions, we adapt the options framework (Precup, 2000; Sutton et al., 1999). 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 $\beta : I \rightarrow [0, 1]$ denotes the option's termination condition, with $\beta(s)$ giving the probability that the option terminates in state $s \in I$.

When a new subgoal is identified, RN generates an option whose policy efficiently takes the agent to this subgoal from any state in the option's initiation set.

The option's initiation set consists of those states that were visited shortly before the subgoal state registered a relative novelty score higher than t_{RN}. How many past transitions to include in this set is determined by a parameter, the *option lag* (l_o).

Following McGovern and Barto (2001), McGovern (2002), Menache et al. (2002), the option's policy is specified through an RL process employing action replay (Lin, 1992) and a reward function specific to the subgoal for which the option was created (corresponding to what Dietterich, 2000, called a *pseudo reward function*). The reward function that RN uses causes a policy to be learned that makes the agent reach the subgoal state in as few time steps as possible while remaining in the option's initiation set. This is achieved by giving a large positive reward for reaching the subgoal, a large negative reward for exiting the initiation set, and a small negative reward for every transition.

And finally, the option's termination condition specifies that the option terminates with probability 1 if the agent reaches the subgoal, or if the agent leaves the initiation set; otherwise, it terminates with probability 0.

3. Experimental Results

We first present results in two simple gridworld domains. These domains are helpful in illustrating the behavior of the algorithm in easily visualized form. We then show results in a more complex task—the taxi task introduced by Dietterich (2000).

The off-line part of the algorithm was conducted only once, using the data obtained from the random walk in the two-room gridworld discussed earlier in Section 2.2. Using an ROC curve analysis, t_{RN} was set to 2, which lead to a p value of 0.0712 and a q value of 0.0056. Other parameter settings were as follows: $\frac{\lambda_{fa}}{\lambda_{miss}} = 100$, $\frac{p(N)}{p(T)} = 100$, $l_n = 7$, $l_o = 10$. No limit was set on the number of options that could be generated; and no filter was employed to exclude certain

states from being identified as subgoals. In all of our experiments, the agent used Q-learning with ϵ-greedy exploration with $\epsilon = 0.1$. The learning rate (α) was kept constant at 0.05; initial Q-values were 0.

3.1. Two-Room Gridworld

Our first example is the two-room gridworld in figure 2a. As noted above, this domain was used to estimate the class-conditional relative novelty distributions by having the agent perform repeated random walks. We realize that presenting the performance of the RN algorithm in this domain is akin to building and evaluating a classifier on the same data set; but we believe that it is important to include it here, as performance on this task shows how well the algorithm can do given almost perfect estimates of the class-conditional relative novelty distributions.

The agent started each episode on a random square in the west room; the goal was the grid square on the Southeast corner of the grid. The four primitive actions—`north`, `south`, `east`, `west`—moved 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 was blocked, the agent remained in the same location. The agent received a reward of 1 at the goal state, and a reward of 0 at all other states. The discount factor was 0.9.

Figure 2b shows a visual representation of the location and frequency of the subgoals identified in 30 runs. The color of a square in this figure corresponds to the number of times it was identified as a subgoal, with lighter colors indicating larger numbers. The state that was identified as a subgoal most frequently was the doorway—in 25 of the 30 runs. Mean number of subgoals identified per run was 1.6; 96% of the subgoals were within two steps of the doorway.

Figure 2c shows the mean number of steps taken to reach the goal state, with and without RN. The figure reveals that RN identified useful subgoals and showed a marked improvement in performance within 10 episodes.

3.2. Six-Room Gridworld

Our second example is the gridworld environment in Figure 3a, which was used by Menache et al. (2002) to demonstrate the utility of their Q-Cut algorithm. The grid dynamics were the same as before. Start and goal states were as shown in the figure. The target states in this domain are the six doorways between the rooms. Figures 3b and 3c show the results of 30 runs. Mean number of subgoals identified per run was 15.3; of the subgoals identified, 30% were target states and 24% were states one transition away from the targets. As in the previous example, the options generated drastically improved the agent's performance.

3.3. Taxi Task

The taxi task has been a popular illustrative problem for RL algorithms since its introduction by Dietterich (2000). The task is to pick-up and deliver a passenger to her destination on a 5×5 grid depicted in Figure 4a. There are four possible source and destination locations for the passenger: the grid squares marked by R, G, B, Y. The source and destination are randomly and independently chosen in each episode. The initial location of the taxi is one of the 25 grid squares, picked uniformly random. At each grid location, the taxi has a total of six primitive actions: `north`, `east`, `south`, `west`, `pick-up`, `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. Reward is -1 for each action, an additional $+20$ for passenger delivery, and an additional -10 for an unsuccessful `pick-up` or `put-down` action.

This domain has 500 states: 25 grid locations $\times$ 5 passenger locations (including in-taxi) $\times$ 4 destinations. The sequential nature of the task gives rise to access states that denote completion of a subtask: going to the passenger location, and picking up the passenger—two other subtasks, going to the destination and dropping off the customer, also give rise to meaningful subgoals but these are too close to the end of an episode for computing relative novelty (because of the novelty lag). There are also navigational access states in this domain—grid squares (2,3) and (3,3)—as the walls limit navigation quite a bit. Both types of access states are repeated for a number of values of other state variables (e.g., grid square (3,3) is an access state when destination is R, G, B, or Y); as a consequence, there are a total of 72 access states; 32 of these complete a subtask, 40 of them are navigational.

We evaluated the performance of RN in 100 runs. Figure 4b shows a visual representation of the grid location of the subgoals, ignoring the other two state variables. Mean number of subgoals identified per run was

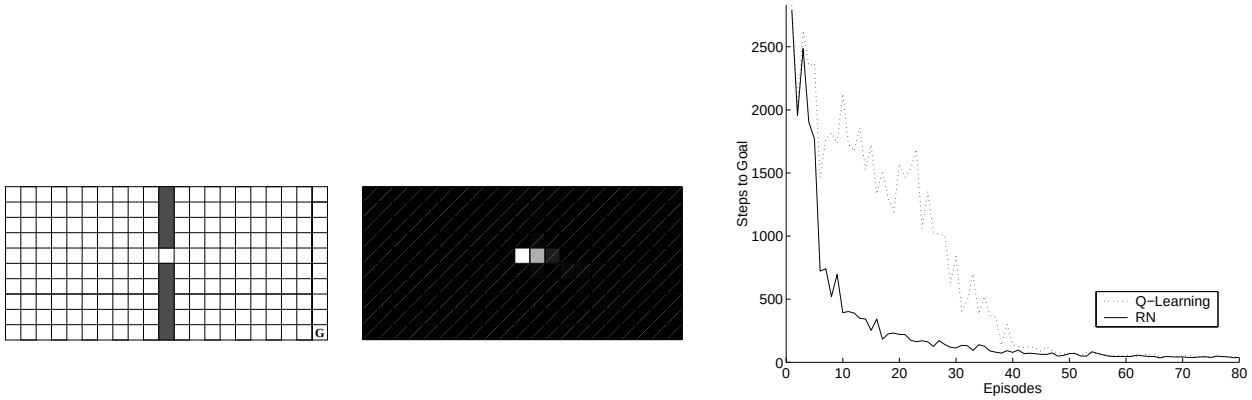

Figure 2. (a) Two-Room gridworld environment, (b) Subgoals identified, (c) Mean steps to goal.

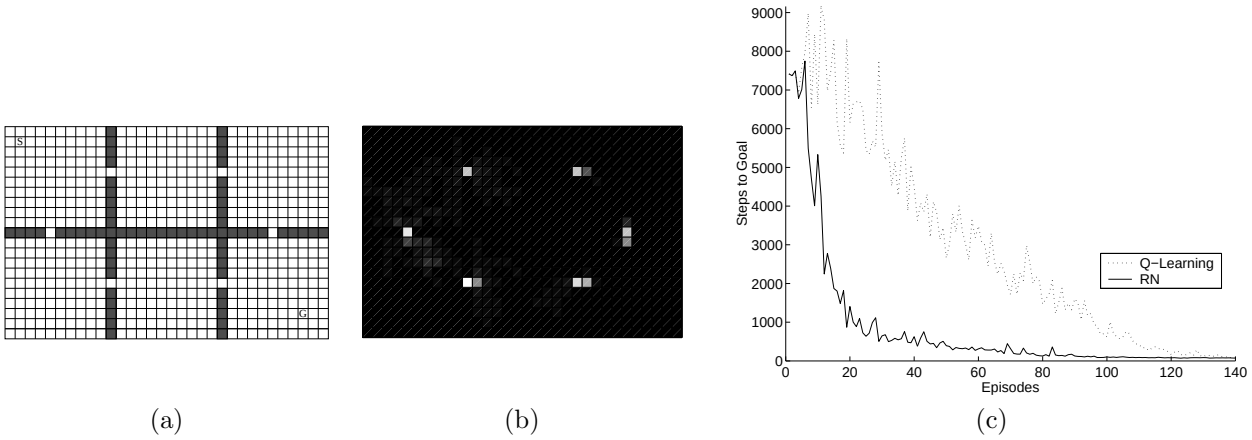

Figure 3. (a) Six-Room gridworld environment, (b) Subgoals identified, (c) Mean steps to goal.

25.8. Of these, 78% were subtask subgoals that correspond to getting to the passenger location and picking up the passenger. Another 14% were states that are one transition away from these. Navigational bottlenecks accounted for 4% of the subgoals. Altogether, these add up to 96% of the subgoals identified. Mean number of steps to complete the task is shown in Figure 4c, which reveals a dramatic improvement in mean performance when RN was used.

4. Discussion

This work is closely related to a number of methods proposed earlier in the literature, most notably to Menache et al. (2002), Hengst (2002), McGovern (2002), and McGovern and Barto (2001), all of which identify subgoal states and define temporally-extended activities that take the agent efficiently to these states. In assessing the relative merits of our method, it would be useful to empirically compare the algorithms in a wide variety of domains; we leave this for future work, but provide a qualitative comparison here.

Menache et al. (2002) define their subgoals to be the border states of strongly connected regions of the MDP transition graph. In essense, our access states are an alternative description of these, and we may say that Q-Cut and RN seek to identify the same states as subgoals. The difference between the algorithms is how they search for them. Q-Cut takes a global approach, and identifies as subgoals those states that perform a minimum cut of the state transition graph, viewing the transition graph as a whole and recursively partitioning it into smaller pieces. Our method, in contrast, takes a local approach. It examines only the most recent experiences of the agent, and *approximates* finding a minimum cut of the corresponding transition graph (though never building the graph). We use a local search procedure because an access state is defined locally, with reference to two neighboring regions, in-

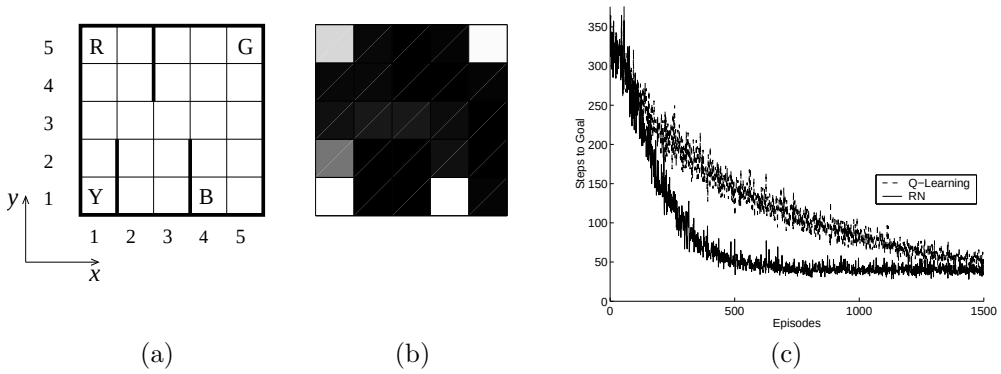

Figure 4. (a) The taxi domain, (b) Subgoals identified (showing only the grid location variable), (c) Mean steps to goal.

dependently of the rest of the transition graph. An access state may or may not be part of a global cut—getting out of one's house is an access state, but is not necessarily part of a global cut in the context of one's entire state space. Because of this fundamental difference in their search mechanisms, Q-Cut and RN are expected to differ in the subgoals they identify, although they search for the same type of subgoals.

Q-Cut and RN also differ in the computational cost of their subgoal discovery methods. The minimum cut procedure employed by Q-Cut has a time complexity of $O(n^3)$, where n is the number of states, while the subgoal discovery method of RN has a time complexity of $O(1)$—RN examines only the most recent experiences of the agent, so the computational cost does not grow with the number of states.

The *region exits* of Hengst (2002) are also related to our access states, in that some domains have states that satisfy both definitions.[1] One example is the subtask-completion subgoals in the taxi task discussed earlier—these were identified more efficiently by HEXQ than by RN, though in this domain RN also identified useful navigational subgoals which HEXQ does not search for—but in most tasks, region exists and access states will not be identical.

The main difference between our subgoals and those of McGovern (2002) and McGovern and Barto (2001) is the role of the reward function. The methods of McGovern et al. require that their subgoals occur frequently on successful trajectories, but not on unsuccessful ones. We do not pay attention to the reward function, arguing that access states are useful regardless of their relation to the actual goal of the agent. For instance, this would certainly be true in an environment where the goal state is not stationary. In that case, by using the access states it has already identified in the environment, an agent would be able to accelerate learning in the new task, even if the solution path of the new task is completely different than that of the old one.

RN is also related to E^3 (Kearns & Singh, 1998) and R-MAX (Brafman & Tennenholtz, 2002) algorithms, in that all three algorithms influence, implicitly or explicitly, the exploration behaviour of the agent. E^3 and R-MAX do this by guiding the agent's action choices based on how well an individual state is known, while RN does it by providing opportunities for the agent to exit its current region.

The algorithm introduced here is a simple implementation of the general idea of using a measure of short-term novelty to identify states that may form useful target states for a collection of temporally-extented activities. The intuition is that if the ease of reaching such states is improved, the agent's access to unexplored regions of the state space will improve, thus leading to more efficient exploration. A key aspect of the algorithm is that the process of identifying subgoals is not dependent on the reward function of the overall task. Indeed, there may be no such reward function. This implies that the method can facilitate transfer among multiple tasks with disparate reward functions and that it can provide potentially useful abstract actions before any particular task has been solved (in cases where "solving a task" has a well-defined meaning). This property is essential if an automatic abstraction method is to be useful in extending the utility of RL to complex real-world tasks.

A weakness of the algorithm is the empirical and heuristic setting of its parameters. This is far from ideal, and an important direction for future research is to allow the agent to adaptively set these parameters. More importantly, in domains that lack the symmetric

[1] Region exits are state-action pairs, but we refer to the state component in this discussion.

structure of the tasks we used as examples here, there may be large differences in the distributions of relative novelty scores in different parts of state space—for example if the effective dimensionality is different in different regions—which will require that the algorithm be sensitive to these differences to be effective. In its current form, the algorithm is not able to handle these differences. A related question we have not addressed here is the sensitivity of the algorithm to the settings of the parameters.

Refining the definition of relative novelty for high-dimensional or continuous state spaces is another important direction for future research. The definition we use here is for discrete-state problems only and therefore has limited applicability.

Given the simple problems we have experimented on and the questions we leave unanswered, we can not claim that our method will scale to more complex tasks—future research is needed to determine whether the method will be generally effective—but we can conclude from the results we have presented here that our approach has the potential for facilitating learning on more challenging tasks.

Finally, we comment that various concepts of novelty are closely linked to motivation and reward in animals (e.g., Kakade & Dayan, 2001). The use of novelty measures to drive the automatic creation of hierarchical behavior architectures may provide useful computational interpretations of novelty-related animal behavior.

Acknowledgments

We would like to thank Daniel Bernstein, Mohammad Ghavamzadeh, Sridhar Mahadevan, Balaraman Ravindran, Michael Rosenstein, and three anonymous reviewers for their useful comments and suggestions. This work was supported by the National Science Foundation under Grant No.ECS-0218123 to Andrew G. Barto and Sridhar Mahadevan. 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

Brafman, R. I., & Tennenholtz, M. (2002). R-MAX - A general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*.

Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, *13*, 227–303.

Digney, B. (1998). Learning hierarchical control structure for multiple tasks and changing environments. *From Animals to Animats 5: The Fifth Conference on the Simulation of Adaptive Behaviour*. The MIT Press.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*. New York: Wiley.

Hengst, B. (2002). Discovering hierarchy in reinforcement learning with HEXQ. *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 243–250). Morgan Kaufmann.

Kakade, S., & Dayan, P. (2001). Dopamine bonuses. *Advances in Neural Information Processing Systems* (pp. 131–137). MIT Press.

Kearns, M., & Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 260–268). Morgan Kaufmann.

Lin, L. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, *8*, 293–321.

McGovern, A. (2002). *Autonomous discovery of temporal abstractions from interaction with an environment*. Doctoral dissertation, University of Massachusetts, Amherst.

McGovern, A., & Barto, A. G. (2001). Automatic discovery of subgoals in reinforcement learning using diverse density. *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 361–368). Morgan Kaufmann.

Menache, I., Mannor, S., & Shimkin, N. (2002). Q-Cut - Dynamic discovery of sub-goals in reinforcement learning. *Proceedings of the Thirteenth European Conference on Machine Learning* (pp. 295–306). Springer.

Parr, B. R. (1998). *Hierarchical control and learning for markov decision processes*. Doctoral dissertation, Computer Science Division, University of California, Berkeley.

Parr, R., & Russell, S. (1998). Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems* (pp. 1043–1049). MIT Press.

Pickett, M., & Barto, A. G. (2002). PolicyBlocks: An algorithm for creating useful macro-actions in reinforcement learning. *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 506–513). Morgan Kaufmann.

Precup, D. (2000). *Temporal abstraction in reinforcement learning*. Doctoral dissertation, University of Massachusetts Amherst.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Sutton, R. S., Precup, D., & Singh, S. P. (1999). Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, *112*, 181–211.

Thrun, S., & Schwartz, A. (1995). Finding structure in reinforcement learning. *Advances in Neural Information Processing Systems* (pp. 385–392). MIT Press.