# Subgoal Discovery for Hierarchical Reinforcement Learning Using Learned Policies

**Sandeep Goel and Manfred Huber**

Department of Computer Science and Engineering
University of Texas at Arlington
Arlington, Texas 76019-0015
{goel, huber}@cse.uta.edu

**Abstract**

Reinforcement learning addresses the problem of learning to select actions in order to maximize an agent's performance in unknown environments. To scale reinforcement learning to complex real-world tasks, agent must be able to discover hierarchical structures within their learning and control systems. This paper presents a method by which a reinforcement learning agent can discover subgoals with certain structural properties. By discovering subgoals and including policies to subgoals as actions in its action set, the agent is able to explore more effectively and accelerate learning in other tasks in the same or similar environments where the same subgoals are useful. The agent discovers the subgoals by searching a learned policy model for state that exhibits certain structural properties. This approach is illustrated using gridworld tasks.

## Introduction

Reinforcement learning (RL) (Kaelbling, Littman, and Moore, 1996) comprises a family of incremental algorithms that construct control policy through real-world experimentation. A key scaling problem of reinforcement learning is that in large domains an enormous number of decisions are to be made. Hence, instead of learning using individual primitive actions, an agent could potentially learn much faster if it could abstract the innumerable micro-decisions, and focus instead on a small set of important decision. This immediately raises the question of how to recognize hierarchical structures within learning and control systems and how to learn strategies for hierarchical decision making. Within the reinforcement learning paradigm, one way to do this is to introduce subgoals with their own reward functions, learn policies for achieving these subgoals, and then include these policies as actions. This strategy can facilitate skill transfer to other tasks and accelerate learning. It is desirable that the reinforcement learning agent discover the subgoals automatically. Several researchers have proposed methods by which policies learned for a set of related tasks are examined for commonalities (Thrun and Schwartz, 1995) or are probabilistically combined to form new policies (Bernstein, 1999). However, neither of these RL methods introduces subgoals. In other work, subgoals are chosen based on information about the frequency a state was visited during policy acquisition or based on the reward obtained. Digney (Digney 1996, 1998) chooses states that are visited frequently or states where the reward gradient is high as subgoals. Similarly, McGovern (McGovern and Barto, 2001a) uses diverse density to discover useful subgoals automatically. However, in the case of more complicated environments and rewards it can be difficult to accumulate and classify the sets of successful and unsuccessful trajectories needed to compute the density measure or frequency counts. In addition, these methods do not allow the agent to discover subgoals that are not explicitly part of the tasks used in the process of discovering them. In this paper, the focus is on discovering subgoals by searching a learned policy model for certain structural properties. This method is able to discover subgoals even if they are not a part of the successful trajectories of the policy. If the agent can discover these subgoal states and learn policies to reach them, it can include these policies as actions and use them for effective exploration as well as to accelerate learning in other tasks in which the same subgoals are useful.

## Reinforcement Learning

In the reinforcement learning framework, a learning agent interacts with an environment over a series of time steps $t$ = 0, 1, 2, 3, … At any instant in the time the learner can observe the state of the environment, denoted by $s \in S$ and apply an action, $a \in A$. Actions change the state of environment, and also produce a scalar pay-off value (reward), denoted by $r \in \Re$. In a Markovian system, the next state and reward depend only on the preceding state and action, but they may depend on these in a stochastic manner. The objective of the agent is to learn to maximize the expected value of reward received over time. It does this by learning a (possibly stochastic) mapping from states to actions called a policy, $\Pi : S \rightarrow A$ i.e. a mapping from states $s \in S$ to actions $a \in A$ . More precisely, the objective is to choose each action so as to maximize the expected return:

$$R = E[\sum_{i=0}^{\infty} \gamma^i r_i] \qquad (1)$$

where $\gamma \in [0,1)$ is a discount-rate parameter and $r_i$ refers to the pay-off at time *i*. A common approach to solve this problem is to approximate the optimal state-action value function, or Q-function (Watkins, 1989), $Q: S \times A \rightarrow \Re$ which maps states $s \in S$ and actions $a \in A$ to scalar values. In particular, $Q(s, a)$ represents the expected discounted sum of future rewards if action $a$ is taken in state $s$ and the optimal policy is followed afterwards. Hence *Q*, once learned, allows the learner to maximize *R* by picking actions greedily with respect to *Q*:

$$\Pi(s) = \arg\max_{a \in A} Q(s,a) \qquad (2)$$

The value function $Q$ is learned on-line through experimentation. Suppose that during learning the learner executes action $a$ in state $s$, which leads to a new state $s'$ and the immediate pay-off $r_{s,a}$. In this case Q-learning uses this state transition to update $Q(s, a)$ according to:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r_{s,a} + \gamma \max_a Q(s',a)) \qquad (3)$$

The scalar $\alpha \in [0,1)$ is the learning rate.

## Subgoal Extraction

An example that shows that subgoals can be useful is a room to room navigation task where the agent should discover the utility of doorways as subgoals. If the agent can recognize that a doorway is a subgoal, then it can learn a policy to reach the doorway. This policy can accelerate learning on related tasks in the same or similar environments by allowing the agent to move between the rooms using single actions. The idea of using subgoals however is not confined to gridworlds or navigation tasks. Other tasks should also benefit from subgoal discovery. For example, consider a game in which the agent must find a key to open a door before it can proceed. If it can discover that having a key is a useful subgoal, then it will more quickly be able to learn how to advance from level to level (McGovern and Barto, 2001b).

In the approach described in this paper, the focus is on discovering useful subgoals that can be defined in the agent's state space. Policies to those subgoals are then learned and added as actions. In a regular space (regular space here refers to a uniformly connected state space) every state will have approximately the same expected number of direct predecessors under a given policy, except for regions near the goal state or close to boundaries (where the space is not regular). In a regular and unconstrained space, if the count of all the predecessors for every state under a given policy is accumulated and a curve for these counts along a chosen path is plotted, the expected curve would behave like the positive part of a quadratic, and the expected ratio of gradients along such a curve would be a positive constant. In the approach presented here, a subgoal state is a state with the following structural property: the state space trajectories originating from a significantly larger than expected number of states lead to the subgoal state while its successor state does not have this property. Such states represent a "funnel" for the given policy. To identify such states it is possible to evaluate the ratio of the gradients of the count curve before and after the subgoal state. Consider a path under a given policy going through a subgoal state. The predecessors of the subgoal state along this path lie in a relatively unconstrained space and thus the count curve for those states should be quadratic. However, the dynamics changes strongly at the subgoal state. There will be a strong increase in the count and the curve will become steeper as the path approaches a subgoal state. On the other hand, the increase in the count can be expected to be much lower for the successor state of the subgoal as it again lies in a relatively unconstrained space. Thus the ratio of the gradients at this point will be high and easily distinguishable. Let $C(s)$ represent the count of predecessors for a state $s$ under a given policy, and $C_t(s)$ is the count of predecessors that can reach s in exactly t steps:

$$C_1(s) = \sum_{s \neq s'} P(s|s', \Pi(s')) \qquad (4)$$

$$C_{t+1}(s) = \sum_{s \neq s'} P(s|s', \Pi(s'))C_t(s') \qquad (5)$$

$$C(s) = \sum_{i=1}^{n} C_i(s) \qquad (6)$$

where $n$ is such that $C_{n+1} = C_n$ or $n =$ *number of states*, whichever is smaller. The condition $s \neq s'$ prevents the counting of one step loops. $P(s \mid s', \Pi(s'))$ is the probability of reaching state *s* from state *s'* by taking action $\Pi(s')$ (in a deterministic world the probability is 1 or 0). If there are loops within the policy, then the counts for the states in the loop will become very high. This implies that, if no precautions are taken, the gradient criteria used here might also identify states in the loop as subgoals.

To calculate the ratio along a path under the given policy, let $C(s_1)$ be the predecessor count for the initial state of the path and $C(s_t)$ be the count for the state the agent will be in after executing $t$ steps from the initial state. The slope of the curve at step $t$, $\Delta_t$ can be computed as:

$$\Delta_t = C(s_t) - C(s_{t-1}) \qquad (7)$$

To identify subgoals, the gradient ratio $\frac{\Delta_t}{\Delta_{t+1}}$ is computed if $\Delta_t > \Delta_{t+1}$ (If $\Delta_t < \Delta_{t+1}$ then the ratio is less then 1 and state does not fit the criterion. Avoiding the computation of the ratio for such points thus saves computational effort). If the computed ratio is higher then a specified threshold, state $s_t$ will be considered a potential subgoal. The threshold here depends largely on the characteristics of the state space but can often be computed independent of the particular environment.

The subgoal extraction technique presented here has been illustrated using a simple gridworld navigation problem. Figure 1 shows a four-room example environment on a 20x20 grid. For these experiments, the goal state was placed in the lower right portion and each trial started from same state in the left upper corner as shown in Figure1.

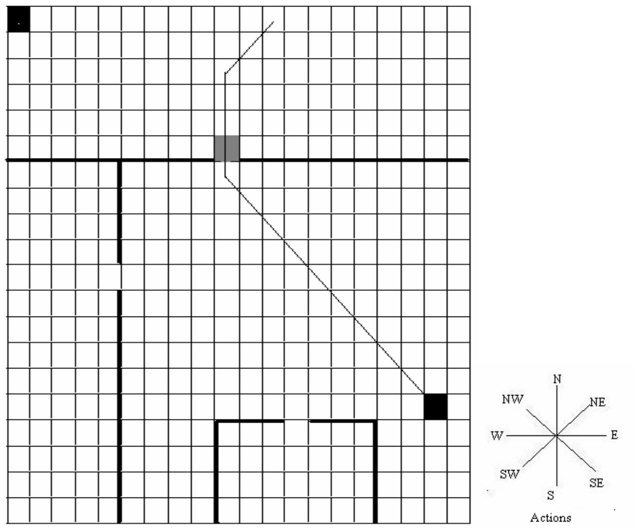

Figure 1. Set of primitive actions (right) and gridworld(left) with the initial state in the upper left corner, the goal in the lower right portion and a random path under the learned policy.

The action space consists of eight primitive actions (*North, East, West, South, North-west, North-east, South-west, and South-east*). The world is deterministic and each action succeeds in moving the agent in the chosen direction. With every action the agent receives a negative reward of -1 for a straight action and -1.2 for a diagonal action. In addition, the agent gets a reward of +10 when it reaches the goal state. The agent learns using Q-learning and ε-greedy exploration. It starts with ε=0.90 (which means 90% of the time it tries to explore by choosing a random action) and gradually decreases the exploration to ε=0.05. In this experiment the predecessor count for every state is computed exhaustively using equations 4, 5, and 6. However, for large state spaces counts can be approximated using Monte Carlo sampling methods. The agent then evaluates the ratio of gradients along the count curve by choosing random paths, and picks the states in which the ratio is higher then the specified threshold as subgoal states. For this experiment the count curve along one of the randomly chosen paths through a subgoal state is shown in Figure 2. The path chosen is indicated in Figure 1 and the subgoal state is highlighted both in Figure 1 and Figure 2. The value for the gradient ratio at step 4 (which is in regular space) is 1.444 while it is 95.0 at step 6 (which is a subgoal state).

To show that the gradient ratio in the unconstrained portion of the state space and at a subgoal state are easily distinguishable, histograms for the distribution of these ratios in randomly generated environments, are shown in Figure 3.

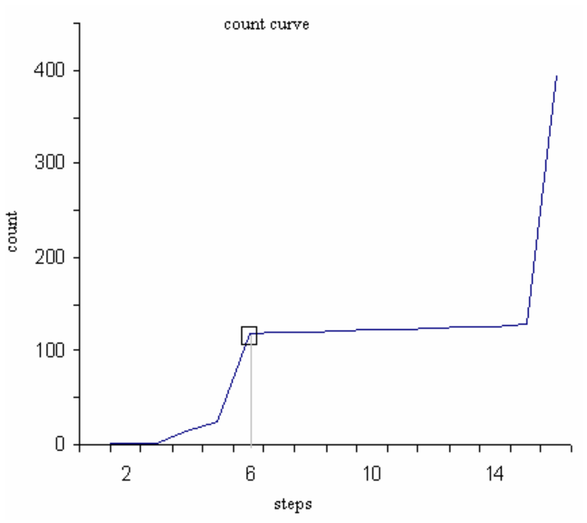

Figure 2. Count curve along a randomly chosen path through a subgoal state under the learned policy.

The Histogram shows data collected from 12 randomly generated 20x20 gridworlds with randomly placed rooms and goals. Each run learns a policy model for the respective task using Q-learning and computes the counts of predecessors for every state using equations 4, 5, and 6. Gradient ratios for 40 random paths in each environment are shown in the histogram.

The subgoal states that the agent discovered in this experiment are shown in Figure 4. The subgoal state leading to the left room is identified here due to its structural properties under the policy and despite the fact that it does not lie on the successful paths between the start and the goal state. The agent did not discover the doorway in the smaller room as a subgoal state because the number of state for which the policy leads through the subgoal is small compared to the other rooms and hence the count for this subgoal state is not influenced significantly by the structural property of the state.

To show that the method for discovering subgoals discussed above is not confined to gridworlds or navigation tasks, random worlds with 1600 states were generated. In these worlds fixed numbers of actions were available in each state. Each action in the state *s* connects to a randomly chosen state *s'* in their local neighborhood. Then the count metric was established and gradient ratios were computed for these spaces with and without a subgoal. The results showed that the gradient ratios in the unconstrained portion of the state space and at a subgoal state are again easily distinguishable.

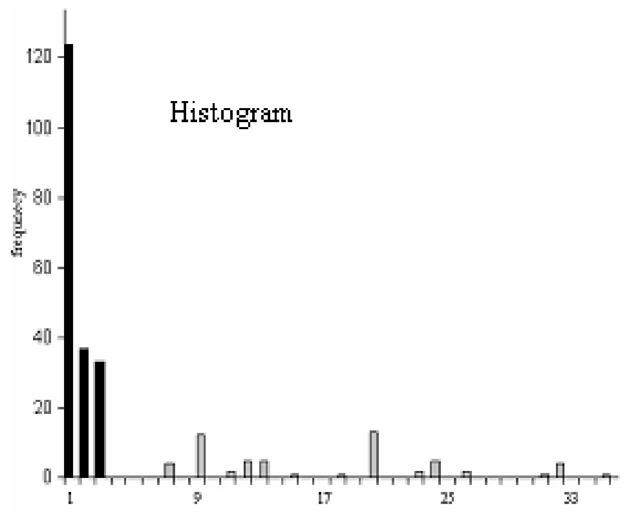

*Figure* 3. Histogram for the distribution of the gradient ratio in regular space (dark bars) and at subgoal states (light bars).

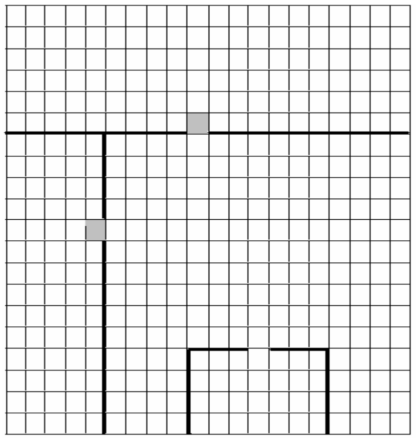

Figure 4. Subgoals states discovered by the agent (light gray states)

## Hierarchical Policy Formation

The motivation for discovering subgoals is the effect that available policies that lead to subgoals have on the agent's exploration and speed of learning in related tasks in the same or similar environments. If the agent randomly selects exploratory primitive actions, it is likely to remain within the more strongly connected regions of the state space. A policy for achieving a subgoal region, on the other hand, will tend to connect separate strongly connected areas. For example, in a room-to-room navigation task, navigation using primitive movement commands produces relatively strongly connected dynamics within each room but not between rooms. A doorway links two strongly connected regions. By adding a policy to reach a doorway subgoal the rooms become more closely connected. This allows the agent to more uniformly explore its environment. It has been shown that the effect on exploration is one of two main reasons that extended actions can be able to dramatically affect learning (McGovern, 1998).

### Learning policies to subgoals

To take advantage of the subgoal states, the agent uses Q-learning to learn a policy to each of the subgoals discovered in the previous step. These policies, which lead to respective subgoal states (subgoal policies) are added to the action set of the agent.

**Learning hierarchical policies.** One reason that it is important for the learning agent to be able to detect subgoal states is the effect of subgoal policies on the rate of convergence to a solution. If the subgoals are useful then learning should be accelerated. To ascertain that these subgoals help the agent to improve its policy more quickly, two experiments were performed where the agent learned a new task with and without the subgoal policies. The same 20x20 grid-world with three rooms was used to illustrate the results. Subgoal policies were included in the action set of the agent (Subg1, Subg2). The task was changed by moving the goal to left hand room as shown in Figure 5. The agent solves the new task using Q-learning with an exploration of 5%.

The action sequence under the policy learned for the new task, when its action set included the subgoal policies is (Subg2, South-west, South, South, South, South) where Subg2 refers to the subgoal policy which leads to the state as shown in Figure 5. Figure 6 shows the learning curves when the agent was using the subgoal policies and when it was using only primitive actions. The learning performance is compared in terms of the total reward that the agent would receive under the learned policy at that point of the learning process. The curves in Figure 6 are averaged over 10 learning runs. Only an initial part of data is plotted to compare the two learning curves; with primitives only the agent is still learning after 150,000 learning steps while with subgoal policies the policy has already converged. After 400,000 learning steps the agent without subgoal policies also converges to the same overall performance. The vertical intervals along the curve indicate one standard deviation in each direction at that point.

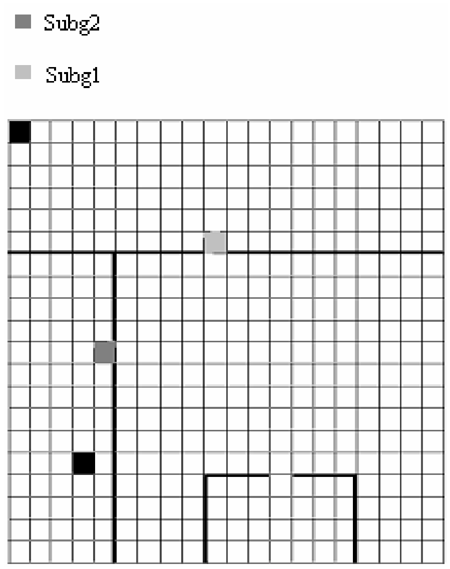

*Figure* 5. New task with goal state in the left hand room.

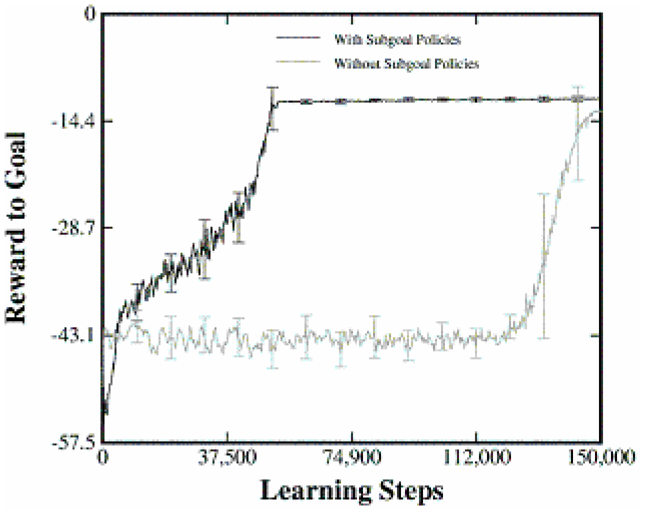

Figure 6. Comparison of learning speed using subgoal policies and using primitive actions only.

## Conclusions

This paper presents a method for discovering subgoals by searching a learned policy model for states that exhibit a funneling property. These subgoals are discovered by studying the dynamics along the predecessor count curve and can include states that are not an integral part of the initial policy. The experiments presented here shows that discovering subgoals and including policies for these subgoals in the action set can significantly accelerate learning in other, related tasks. While the example shown here are gridworld tasks, the presented approach for discovering and using subgoals is not confined to gridworlds or navigation tasks.

## Acknowledgements

This work was supported in part by NSF ITR-0121297 and UTA REP/RES-Huber-CSE.

## References

Bernstein, D. S. (1999). Reusing old policies to accelerate learning on new MDPs (Technical Report UM-CS-1999-026). Dept. of Computer Science, Univ. of Massachusetts, Amherst, MA.

Digney, B. (1996). Emergent hierarchical structures: Learning reactive/hierarchical relationships in reinforcement environments. *From animals to animats 4*: *SAB 96*. MIT Press/Bradford Books.

Digney, B. (1998). Learning hierarchical control structure for multiple tasks and changing environments. *From animals to animats 5*: *SAB 98.*

Kaelbling, L. P., Littman, M. L., and Moore, A. W. ‘‘Reinforcement Learning: A Survey,” *Journal of Artificial Intelligence Research*, Volume 4, 1996

McGovern, A., and Barto, A. G. (2001a). Automatic Discovery of Subgoals in Reinforcement learning using Diverse Density. Proceedings of the 18th International Conference on Machine Learning, pages 361-368.

McGovern, A. (1998). Roles of macro-actions in accelerating reinforcement learning. Master’s thesis, U. of Massachusetts, Amherst. Also Technical Report 98-70.

McGovern, A., and Barto, A. G. (2001b). Accelerating Reinforcement Learning through the Discovery of Useful Subgoals. Proceedings of the 6th International Symposium on Artificial Intelligence, Robotics and Automation in Space.

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

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. Machine Learning 3: 9-44.

Thrun, S. B., & Schwartz, A. (1995). Finding structure in reinforcement learning. *NIPS 7* (pp. 385-392). San Mateo, CA: Morgan Kaufmann.

Watkins, Christopher J.C.H. (1989). Learning from delayed rewards. PhD thesis, Dept. of Psychology, Univ. of Cambridge.