# Online Skill Discovery using Graph-based Clustering

**Jan Hendrik Metzen** JHM@INFORMATIK.UNI-BREMEN.DE
Robotics Group, University Bremen, D-28359 Bremen, Germany

## Abstract

We introduce a new online skill discovery method for reinforcement learning in discrete domains. The method is based on the bottleneck principle and identifies skills using a bottom-up hierarchical clustering of the estimated transition graph. In contrast to prior clustering approaches, it can be used incrementally and thus several times during the learning process. Our empirical evaluation shows that "assuming high connectivity in the face of uncertainty" can prevent premature identification of skills. Furthermore, we show that the choice of the linkage criterion is crucial for dealing with non-random sampling policies and stochastic environments.

## 1. Introduction

Reinforcement Learning (RL) can provide autonomous agents with means to learn to improve their performance with experience. There exist theoretical guarantees for certain RL algorithms that the optimal policy for any discrete Markov Decision Process (MDP) is learned asymptotically. However, scaling RL to real-world problems with large or continuous state spaces remains a major challenge since the amount of experience the agent can collect is limited. One approach to alleviate this problem is Hierarchical RL (Barto & Mahadevan, 2003) which aims at dividing a problem into simpler subproblems, learning solutions for these subproblems, and encapsulate the acquired knowledge into so-called *skills* that can potentially be reused later on in the learning process. It has been shown that skills can help an agent to adapt to non-stationarity of the environment and to transfer knowledge between different but related tasks (Digney, 1998) and can increase the representability of the value function in continuous domains (Konidaris & Barto, 2009).

Presented at $10^{th}$ *European Workshop on Reinforcement Learning (EWRL 2012)*, Edinburgh, Scotland, UK, 2012. Copyright 2012 by the author(s)/owner(s).

One of the major challenges in Hierarchical RL is to identify what might constitute a useful skill, i.e. how the problem should be decomposed. Skills should be reusable, distinct, and easy to learn. The task of identifying such skills is called *skill discovery*.

We propose the new online skill discovery method OGAHC which identifies densely connected regions in the state space using a constrained agglomerative hierarchical clustering of the estimated state transition graph. We show empirically in Section 4.1 that the proposed method is robust with regard to the agent's sampling policy and the stochasticity of the environment. In Section 4.2, we compare the proposed incremental, online skill discovery method to its non-incremental, offline counterpart. Furthermore, we show in Section 5 that using OGAHC for skill discovery in a Hierarchical RL agent outperforms related skill discovery approaches in a multi-task learning domain.

## 2. Background and Related Work

We adopt the *option framework* (Sutton et al., 1999) for Hierarchical RL, in which skills are formalized as *options*: An option $o$ consists of three components: the option's initiation set $I_o \subset S$ which defines the states in which the option may be invoked, the option's termination condition $\beta_o : S \rightarrow [0,1]$ which specifies the probability of option execution terminating in a given state, and the option's policy $\pi_o$ which defines the probability of executing an action in a state under option $o$. In the options framework, the agent's policy $\pi$ may in any state $s$ decide not solely to execute a primitive action but also to call any of the options for which $s \in I_o$. If an option is invoked, the option's policy $\pi_o$ is followed for several time steps until the option terminates according to $\beta_o$.

The option's policy $\pi_o$ is defined relative to an option-specific reward function $R_o$ that may differ from the global external reward function. Learning $\pi_o$ given $I_o$, $\beta_o$, and $R_o$ is denoted as *skill learning* and learning $\pi$ given primitive actions and options is known as *compositional learning*. In this work we focus on *skill discov-*

*ery*, i.e. choosing appropriate $I_o$, $\beta_o$, and $R_o$ for a new option $o$. Autonomous skill discovery is very desirable since the quantities $I_o$, $\beta_o$, and $R_o$ need not be predefined but can been identified by the agent itself and thus, skill discovery increase the agent's autonomy.

Most prior work on autonomous skill discovery is based on the concept of *bottleneck areas* in the state space. Informally, bottleneck areas have been described as the border states of densely connected areas in the state space (Menache et al., 2002) or as states that allow transitions to a different part of the environment (Şimşek & Barto, 2004). A more formal definition is given by Şimşek & Barto (2009) which define bottleneck areas as those states which are local maxima of betweenness—a measure of centrality on graphs—on the transition graph. Once bottleneck areas have been identified, typically one (or several) skills are defined that try to reach this bottleneck, i.e. that terminate with positive pseudo-reward if the bottleneck area is reached, can be invoked in a local neighborhood of the bottleneck, and terminate with a negative pseudo-reward when departing too far from the bottleneck.

Since betweenness requires complete knowledge of the transition graph and is computationally expensive, several heuristics have been proposed to identify bottlenecks. One class of heuristics are *frequency-based approaches* that compute local statistics of states. Diverse density (McGovern & Barto, 2001) and relative novelty (Şimşek & Barto, 2004) have been considered in frequency-based heuristics. In diverse density, bottlenecks are identified as those states that are visited often on successful but not on unsuccessful trajectories. Relative novelty defines bottlenecks as those states that allow the agent to transition to an area in the state space that is otherwise difficult to reach from its current region. One disadvantage of frequency-based approaches is that they require repeated-sampling to obtain accurate estimates of the statistics and are thus not very sample-efficient.

An other class of heuristics that may be more sample-efficient are *graph-based approaches*. These heuristics are based on estimates of the transition graph and aim at partitioning this graph into subgraphs that are densely connected internally but only weakly connected with each other. Menache et al. (2002) propose a top-down approach for partitioning the global transition graph based on the max-flow/min-cut heuristic. Şimşek et al. (2005) follow a similar approach but partition local estimates of the global transition graph using a spectral clustering algorithm and use repeated sampling for identifying globally consistent bottlenecks. Due to the repeated sampling the approach shares some properties with the frequency-based approaches. Mannor et al. (2004) propose a bottom-up approach that partitions the global transition graph using agglomerative hierarchical clustering. The approach is offline, i.e. the clustering can be executed only once and thus does not allow us to identify skills at different times during learning. This property limits the utility of the approach since there may be no single optimal point in time for skill discovery.

The main contribution of this paper is a skill discovery method that is based on agglomerative clustering of the transition graph and shares some similarities with the work of Mannor et al. (2004); however, the proposed method is online, i.e. can identify skills at any time.

## 3. Online Graph-based Agglomerative Hierarchical Clustering

We adopt the concept of identifying bottlenecks in the state space as basis for skill discovery. Informally, a set of states forms a bottleneck area if it lies on the boundary[1] of two densely connected areas of the state space that are only weakly connected mutually. We investigate the utility of approaches that identify such bottlenecks based on the sample transition graph using agglomerative hierarchical clustering. Note that we base skill discovery thus solely on the MDP's state transition probabilities $P^a_{ss'}$ and ignore the expected rewards $R^a_{ss'}$ since we consider $R^a_{ss'}$ to encode the task and aim at identifying task-independent skills that can be reused in different tasks.

### 3.1. Sample Transition Graph

For discrete environments, we construct the *sample transition graph* $G = (V, E, w)$, which is a directed, weighted graph that is used as estimate for the actual unknown transition graph, as follows: for a given set of $n$ transitions $\{(s_i, a_i, s'_i)\}_{i=1\ldots n}$ sampled from an MDP, we set $V = \{s_i\} \cup \{s'_i\}$ to the set of all states that have been visited and $E = \{(s_i, s'_i)\}$ to the set of all one-step transitions that have occurred. For each action $a \in A$, we add the attribute $N(s, a, s') = \sum_{i=1}^{n} \delta((s, a, s'), (s_i, a_i, s'_i))$ to any edge $(s, s') \in E$ and the attribute $N(s, a) = \sum_{i=1}^{n} \delta((s, a), (s_i, a_i))$ to any node $s \in V$ where $\delta(x, y) = 1$ if $x = y$ else 0.

Different choices for the edge weights $w(e)$ have been proposed. While Mannor et al. (2004) used essentially uniform edge weights $w_{uni}(e) = 1$, Şimşek et

[1]For a graph $G = (V, E)$, we define the boundary of two disjoint node sets $A, B \subset V$ as
$b_G(A, B) = \{v \in V | \exists v^* \in V : (v, v^*) \in E \cap (A \times B \cup B \times A)\}$

al. (2005) proposed the edge weights $w_{on}((s,s')) = \sum_a N(s,a,s')$, i.e. how often the transition $s \rightarrow s'$ has been observed under the sampling policy. While $w_{uni}$ ignore both the stochasticity of the environment and the action preferences of the agent, $w_{on}$ take both into account (i.e. it is "on-policy" with regard to the sampling policy). We argue that in order to identify properties of $P^a_{ss'}$ like bottlenecks, one should take the stochasticity into account but be independent of the sampling policy. Thus, we propose the edge weights $w_{off}((s,s')) = \sum_a N(s,a,s')/N(s,a)$, which are "off-policy" (note that $N(s,a,s')/N(s,a)$ is the maximum likelihood estimate of $P^a_{ss'}$ and thus independent of the sampling policy).

## 3.2. Linkage criteria

The purpose of the linkage criterion $l$ is to give a quantitative judgment if the boundary of two connected, disjoint subgraphs $A, B \subset G$ forms a bottleneck in $G = (V, E, w)$. The larger the linkage value for the pair $(A, B)$, the more evidence the criterion offers for a bottleneck between $A$ and $B$. By choosing a threshold $\psi$, one can create a binary criterion for "bottleneckness" (by identifying a bottleneck if $l(A, B) > \psi$).

As base for linkages, we define the (directed) *connectivity mass* $c$ of two subgraphs $A$ and $B$ as follows:

$$c(A, B) = \sum_{e \in E \cap (A \times B)} w(e).$$

The connectivity mass gets large for large, densely connected subgraphs with big edge weights. Mannor et al. (2004) proposed a linkage criterion for bottleneck identification in transition graphs that is defined as follows (using our notation and $|A|$ being the number of vertices in subgraph $A$):

$$M(A, B) = \frac{\min(|A|, |B|) \log(\max(|A|, |B|))}{c(A, B) + c(B, A)}.$$

The linkage is based on uniform edge weights such that the denominator is equal to the number of edges between $A$ and $B$ in $G$. The linkage thus assigns large values to subgraphs of similar sizes that are only weakly interconnected.

Şimşek et al. (2005) have proposed the $\hat{N}_{cut}$ criterion which is defined as follows (using our notation):

$$\hat{N}_{cut}(A, B) = \frac{c(A, B) + c(B, A)}{c(A, V) + c(B, A)} + \frac{c(B, A) + c(A, B)}{c(B, V) + c(A, B)}.$$

The $\hat{N}_{cut}$ criterion is an approximation of the sum of probabilities that the agent transitions in one time step from a state in subgraph $A$ to a state in subgraph $B$ or vice versa. The authors used a top-down spectral clustering algorithm to find graph cuts that minimize $\hat{N}_{cut}$; in order to use it as a linkage criterion where large values indicate a bottleneck, we shall use $-\hat{N}_{cut}$. The authors used the criterion with on-policy edge weights; we claim that it should rather be used with off-policy weights (see Section 4.1).

**Algorithm 1** Constrained agglomerative clustering

```
Input: graph $G = (V, E, w)$, constraint set $C$, linkage criterion $l$, threshold $\psi$
Initialize partition: $P = \{\{v\} | v \in V\}$
# Merge only clusters $p_i$ that are connected in $G$
$C = C \cup$ lambda $p_1, p_2 : (p_1 \times p_2) \cap E \neq \emptyset$
loop
    # Merge-candidates that fulfill all constraints
    $M = \{(p_1, p_2) | (p_1, p_2) \in (P \times P) \wedge \bigwedge_{c \in C} c(p_1, p_2)\}$
    # Find merge candidates with minimal linkage
    $p_1^*, p_2^* = \arg\min_{p_1, p_2 \in M} l(p_1, p_2)$
    if $l(p_1^*, p_2^*) > \psi$: return $P$
    # Merge $p_1^*$ and $p_2^*$
    $P = (P \setminus \{p_1^*, p_2^*\}) \cup \{p_1^* \cup p_2^*\}$
end loop
```

## 3.3. Constrained Agglomerative Clustering

In order to identify all bottlenecks of $G$, we aim to determine a partition $P^*$ of minimal cardinality of the graph nodes into disjoint sets $p_i$ (called clusters) such that each induced subgraph does not contain a bottleneck. By construction, the boundary of any connected pair of these subgraphs forms a bottleneck (otherwise, the two subgraphs would have been merged because of the minimal cardinality objective). Formally:

$$P^* = \arg\min_{P \in \mathcal{P}(V)} |P| \qquad \text{s.t.} \max_{p_i \in P, q_i \subset p_i} l(p_i \setminus q_i, q_i) \leq \psi,$$

with $\mathcal{P}(V)$ being the set of all possible partitions of $V$. Alternatively, we may also fix the number of clusters to $k$ and identify a partition $P^*$ with $|P^*| = k$ such that the maximal intra-cluster linkage $l$ is minimized:

$$P^* = \arg\min_{|P| = k} \max_{p_i \in P, q_i \subset p_i} l(p_i \setminus q_i, q_i).$$

Since finding the optimal solutions for these problems is $\mathcal{NP}$-hard, we use an approximate approach similar to the one proposed by Mannor et al. (2004) which is based on agglomerative hierarchical clustering (see Algorithm 1). This algorithm starts by assigning each node into a separate cluster and afterwards merges greedily the pair of clusters with minimal linkage until no pair remains with a linkage below $\psi$. As proposed by Mannor et al., only clusters which are connected in

$G$ can be merged. Additionally, the algorithm allows the specification of further constraints that determine which clusters may be merged (see Section 3.4).

## 3.4. Online Clustering

The clustering approach presented in Section 3.3 is an offline approach, i.e. it can be executed only once when "enough" transitions have been gathered to estimate the sample transition graph. The specific point in time when clustering is performed presents a trade-off: on the one hand, one would like to perform the clustering as early as possible in order to maximize the impact of the discovered skills during learning; on the other hand, performing the clustering too early may result in spurious clusters that are due to considerable deviations of the estimated transition graph from the true transition graph.

It would thus be highly desirable to use an online clustering algorithm instead, which can be invoked several times during learning. In order to make such an online clustering algorithm useful for skill discovery, it has to fulfill the following properties: a) Subsequent executions of the clustering should be consistent, i.e. bottlenecks identified in one invocation of the clustering should persist in subsequent ones. b) The less reliable the estimate of the transition graph in a certain area of the state spaces is, the less likely the clustering should identify bottlenecks in this area. In this section, we present the *Online Graph-based Agglomerative Hierarchical Clustering* (OGAHC) method (see Algorithm 2), which is tailored towards these demands.

**Algorithm 2** OGAHC

```
**Input:** linkage criterion $l$, parameters $\rho$, $\psi$, $m$
Initialize: partition $P = \emptyset$, graph $G = (\emptyset, \emptyset)$,
**loop**
  # Act for $m-1$ steps and observe trajectory
  $(s_1, a_1, s_2, \ldots, a_{m-1}, s_m) = \text{ACT}(\text{AGENT, ENV})$
  # Update transition graph with trajectory
  $\text{UPDATE}(G, (s_1, a_1, s_2, \ldots, a_{m-1}, s_m))$
  # Pseudo transitions for under-explored nodes
  $G' = \text{SMOOTH}(G, \rho)$
  # Constraints for keeping partitions consistent
  $C = \emptyset$
  **for** $(p_A, p_B) \in P \times P$ with $p_A \neq p_B$ **do**
    # Must not merge two clusters with elements that had not been merged in last iteration
    $C = C \cup \{$**lambda** $p_1, p_2 :$
          $(p_1 \cup p_2) \cap p_A = \emptyset \vee (p_1 \cup p_2) \cap p_B = \emptyset\}$
  **end for**
  # Partition $G'$ using constrained agglomerative clustering (CAC)
  $P = \text{CAC}(G', C, l, \psi)$
**end loop**
```

The main contributions of the OGAHC method are the following: a) In order to guarantee consistency of subsequent clusterings, an increasing set of constraints that needs to be obeyed by the clustering is maintained: for each pair of clusters obtained in one clustering, one constraint is created that prevents that any two nodes that have been placed in different of these two clusters are merged into the same cluster in subsequent clusterings. b) In order to prevent premature identification of bottleneck areas, OGAHC builds on an assumption that may be framed as "assume high connectivity in the face of uncertainty". Technically, instead of working on the maximum likelihood estimate $G$ of the transition graph, the clustering is performed on a modified transition graph $G'$. In $G'$, for each $N(s,a) < \rho$, $N(s,a,s')$ is incremented by $(\rho - N(s,a))/k$ for any $s'$ that is among the $k = \max(5, \rho)$ nearest neighbors of $s$, and $N(s,a)$ is set to $\rho$. $\rho$ is a free parameter of the algorithm that determines if the algorithm is more liberal, i.e. tends to identify bottlenecks early in the learning process despite the uncertainty in the sample transition graph, or—for larger values of $\rho$— more conservative. This process is denoted as $\text{SMOOTH}(G, \rho)$ in the pseudo code.

The assumption of "high connectivity in the face of uncertainty" prevents bottlenecks from being identified prematurely since the linkage value is typically decreased by adding pseudo transitions and thus, fewer clusters are formed in early invocations of the algorithm. Over time, fewer graph nodes are under-explored, fewer pseudo transitions are added, the connectivity decreases, the typical linkage of subgraphs increases, and thus, more clusters are formed. Note that specifying the number of clusters $k$ beforehand instead of $\psi$ is not practical in online clustering since it does not allow the implicit increase of granularity of the clustering over time.

## 3.5. Skill Prototype Generation

Given the partitioning $P$ obtained using OGAHC, one skill is generated for reaching each identified bottleneck area. We set the bottleneck area of two clusters $A$ and $B$ to the boundary of the corresponding subgraphs of $G$. The skill prototype $(I_{AB}, \beta_{AB}, R_{AB})$ that is generated for the bottleneck between $A$ and $B$

is then defined as follows:

$$I_{AB} = (A^* \cup B^*) \setminus b_G(A,B)$$
$$\beta_{AB}(s) = 0 \text{ if } s \in I_{AB} \text{ else } 1$$
$$R_{AB}((s,a,r,s')) = r \text{ if } s' \in (A^* \cup B^*) \text{ else } r_p + r,$$

where $A^* = \{v \in V | \exists v' \in A : (v, v') \in E\}$, and $r_p$ is a parameter of the algorithm that determines the penalty for failing to fulfill a skill's objective. Further skill prototypes are generated for reaching identified terminal states $s_t$ from the adjacent clusters $A$:

$$I_{As_t} = A^* \setminus \{s_t\}$$
$$\beta_{As_t}(s) = 0 \text{ if } s \in I_{As_t} \text{ else } 1$$
$$R_{As_t}((s,a,r,s')) = r \text{ if } s' \in A^* \text{ else } r_p + r,$$

Note that the skill prototypes may change over time when $A$ and/or $B$ change because of re-clustering; however, a skill and its corresponding bottleneck area can never disappear.

## 4. Cluster Accordance Analysis

This section presents an empirical evaluation of the quality of the partitions generated by different clustering approaches in 50 MDPs. These MDPs have been created randomly as follows: the state space of all MDPs has been set to a two dimensional grid of 400 states ($S = \{0, 1, \ldots, 19\}^2$) and the action space to contain four discrete actions ($A = \{(-1, 0), (1, 0), (0, -1), (0, 1)\}$). For any state transition, a reward of $-1$ is given, and an episode starts always in $s_0 = (0, 0)$ and terminates in $s_t = (19, 19)$.

The MDP's state transition probability $P^a_{ss'}$ depends on an implicit connectivity graph $G_c$. This graph is created based on a partitioning of the states that is generated by drawing $k = 7$ states (the "centers") uniform randomly from $S$ and assigning each state to the cluster of its closest center (breaking ties randomly). A graph edge is added between any pair of states that are neighbors in the 4-neighborhood and in the same cluster. One additional edge per cluster-pair $(p_i, p_j)$ is added between a randomly drawn pair of neighbors where one is in cluster $p_i$ and the other in $p_j$.[2] For any state-action pair $s, a$, let the deterministic successor state be $d(s, a) = s + a$ if $(s, s + a)$ is an edge in $G_c$ and else $d(s, a) = s$. We set $P^a_{ss'} = 1 - (8/9)\chi$ for $s' = d(s, a)$ and $P^a_{ss'} = (1/9)\chi$ for any other state $s'$ from the 9-neighborhood of $s$. $\chi$ is a parameter of the MDP that determines its stochasticity. Note that for $\chi = 0$, the connectivity of $G_c$ controls the connectivity of the sample transition graph, while for $\chi > 0$, it influences only its edge weights.

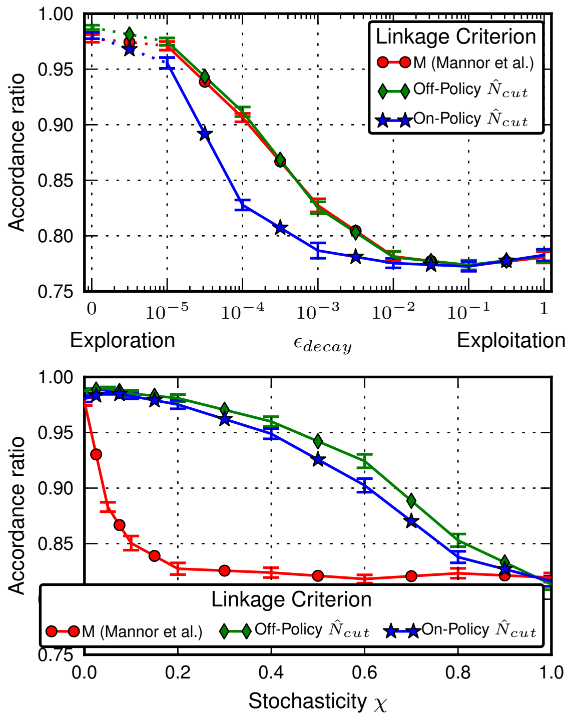

*Figure 1.* Accordance ratios for different linkage criteria in randomly generated MDPs for different degrees of exploration and different stochasticity of the environment.

For the given MDP, the optimal policy is computed offline and a trajectory consisting of $n$ transitions is sampled following an $\epsilon$-greedy policy which takes the action provided by the optimal policy with probability $1 - \epsilon$ and a random action otherwise. $\epsilon$ is set initially to 1 and then decayed after each step by $\epsilon = \epsilon(1 - \epsilon_{decay})$. Based on the sampled transitions, a partition of the states is determined and compared to the ground truth partition. The agreement of the two partitions $P_1$ and $P_2$ of $S$ is measured using the accordance ratio

$$acc(P_1, P_2) = \frac{1}{|S|^2} \sum_{s,\bar{s} \in S} \delta(\delta(P_1(s), P_1(\bar{s})), \delta(P_2(s), P_2(\bar{s}))),$$

i.e. the ratio of element-pairs on which $P_1$ and $P_2$ agree on assigning them to the same or to different clusters ($\delta(x, y) = 1$ if $x = y$ else 0). Statistical hypothesis testing has been conducted using Student's independent two-samples t-test. Figures show the standard error of the mean.

### 4.1. Linkage criterion

In a first experiment, we compare different linkage criteria in an offline clustering setting for $k = 7$ and

[2]Visualizations of the 50 partitions and connectivity graphs are contained in the supplementary material.

$\chi = 0$. To analyze the effect of non-uniform exploration of the agent, we have varied $\epsilon_{decay}$. Large values of $\epsilon_{decay}$ correspond to an agent which starts to exploit more early and as a consequence, obtains a more biased estimate of the transition graph. The upper plot in Figure 1 shows a comparison of the partitions obtained after $n = 25000$ transitions for the $M$-linkage and the $\hat{N}_{cut}$-linkage with both on-policy and off-policy edge weights as discussed in Section 3.2. The main result shown in the figure is that the on-policy $\hat{N}_{cut}$ linkage obtains significantly worse partitions than the other two linkages for intermediate values of $\epsilon_{decay}$ ($p < 0.008$ for $\epsilon_{decay} \in \{10^{-5}, 10^{-4}, 10^{-3}\}$). This can be attributed to the fact that the on-policy $\hat{N}_{cut}$ linkage bases its partitioning not solely on the environment's transition probabilities but also on the agent's action selection which is undesirable if the agent selects action non-uniformly.

In a second experiment ($n = 25000$, $k = 7$, $\epsilon_{decay} = 0$), we have varied the stochasticity $\chi$ of the environment. The lower plot in Figure 1 shows that the $M$-linkage obtains significantly worse results than the off-policy $\hat{N}_{cut}$ linkage if the environment gets slightly non-deterministic ($p < 0.0001$ for $0.05 \leq \chi \leq 0.8$). This is due to the use of uniform edge weights in the $M$-linkage; these weights do not allow to differentiate between probable and less probable transitions, which is apparently important for skill discovery in non-deterministic environments. In summary, the results show that the off-policy $\hat{N}_{cut}$-linkage is the most robust linkage criterion; the following experiments have been conducted with this linkage accordingly.

## 4.2. Online Graph-Clustering

In this subsection, we compare the OGAHC algorithm with its offline counterpart (essentially the approach proposed by Mannor et al. (2004)). We have used the same 50 MDPs as before and set $n = 30000$, $\epsilon_{decay} = 0$, $\chi = 0$, and $\psi = -0.075$. For OGAHC, the clustering has been updated every 500 steps and different choices of $\rho$ have been evaluated. The offline clustering has been performed after $m \in \{500, 1000, \ldots, 30000\}$ transitions, taking all $m$ transitions that have been acquired so far into account at once.

The upper plot in Figure 2 shows how the accordance ratio of the identified partitions changes over time. It can be seen that smaller values of $\rho$ achieve higher accordance ratios in the early phase since they tend to identify clusters more early. However, these clusters are potentially suboptimal since they are based on a sample transition graph that was estimated based on a small number of transitions and has thus a high

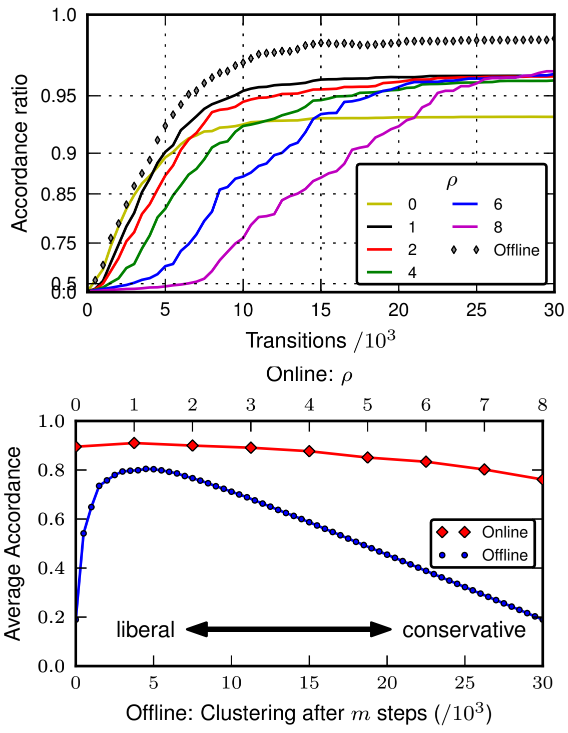

*Figure 2.* Accordance ratios and average accordance obtained for different values of $\rho$ in OGAHC and by offline clustering. Please note that the y-axis in the upper graphic has been non-linearly transformed to improve readability.

sampling error. Accordingly, it can be seen that for e.g. $\rho = 0$ the accordance ratio plateaus on a lower level than for larger values of $\rho$. Comparing OGAHC to offline clustering, one can see that at any specific point in time, the online clustering tends to be slightly worse. The accordance of the offline clustering represents an upper boundary for the performance of the online clustering; performance degradation can either be due to the enforced consistency with prior clusterings or to the influence of $\rho$.

In the lower plot of Figure 2, the average accordance ratio over the 30000 steps is shown for online and offline clustering for different values of $\rho$ and $m$. It can be seen that for both small and large values of $m$, the average accordance of the offline clustering is low. For small $m$, the average accordance is low because the clustering is based on too little transitions and cannot be improved later on; for large values of $m$, the average accordance is low because there is no clustering at all for a long initial period. Intermediate values ($m \approx 4500$) obtain a higher average accordance of approx. 0.81. In contrast, the online clustering depends less on the specific choice of $\rho$. The optimal value for $\rho$ is 1, which results in an average accordance of ap-

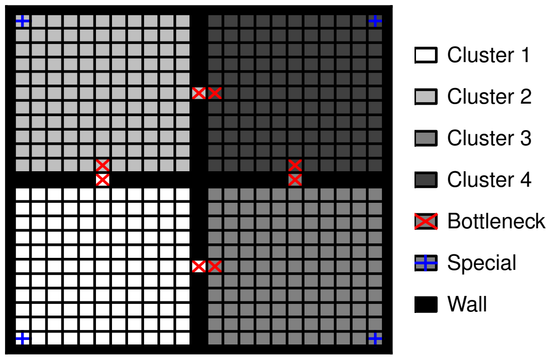

*Figure 3.* Structure, ground-truth partitioning, and corresponding bottlenecks of the multi-task maze world.

prox. 0.91; however, for any value $0 \leq \rho \leq 6$, OGAHC achieves a significantly higher average accordance than the offline clustering for $m = 4500$ ($p < 0.019$). Thus, even without fine-tuning $\rho$, the online clustering can outperform the offline clustering with optimally chosen $m$ with regard to the average clustering quality. In the next section, we will investigate if the same holds true for the learning performance of a Hierarchical RL agent using OGAHC for skill discovery compared to agents using other approaches.

## 5. Multi-task Learning

In this section, we evaluate the utility of the OGAHC method within a Hierarchical RL agent in a multi-task learning scenario. The scenario is a $23 \times 23$ maze world consisting of four rooms with four "special" states (see Figure 3). In each time step, the agent obtains a reward of $-1$. The agent has to learn to solve 12 different tasks in this environment; each task is defined by a pair $(s_0, s_t)$ of special states where $s_0$ is the start state and $s_t$ is the goal state of the task. In each episode, a task is chosen at random and the specific task is communicated to the agent as a state space dimension. This dimension is used by the agent for policy learning but ignored during skill discovery and skill learning. The acquired skills can thus be transferred between tasks and may give a useful exploration bias for the agent.

The agent uses a 2-level hierarchy: The lower level consists of the acquired skills and a special option discussed below. On the upper level, the agent may choose among these skills but not choose a primitive action directly, i.e. the action space is abstracted not augmented. 1-step intra-option Q-Learning (Precup, 2000) is used for skill learning; no experience replay is conducted to avoid intermixing the contributions of skill discovery and experience replay (see Jong et al. (2008) for a discussion). A special low-level option prototype is provided to the agent: this option can be invoked in any state, terminates in any state with probability $\beta = 0.01$, and uses the external reward signal. This option allows the agent thus to learn a monolithic global policy. The reasons for this option are twofold: on the one hand, it guarantees that the agent can learn a global optimal policy eventually despite the abstraction of the action space. On the other hand, it makes the agent less susceptible to the broken exploration symmetry that is caused by temporal abstraction (see (Jong et al., 2008)) than pure augmentation of the action space.

We compare OGAHC to the Offline Clustering approach proposed by Mannor et al. (2004) and the Local Graph Partitioning (LGP) approach by Şimşek et al. (2005). As baseline, we evaluate the agent using no skill discovery (i.e. learning a monolithic policy using the "special" option), and the agent that is provided with the ground-truth partitioning (see Figure 3) from the beginning. $|\psi|$ has been set to 0.075 for all approaches. For LGP, the required hit-ratio has been set to $t_p = 0.2$, the option lag to $l_o = 20$, the window length to $h = 1000$, and the update frequency to 250. These parameters have been chosen based on preliminary experiments. The discount factor of the agent has been set to $\gamma = 0.99$, the learning rate to $\alpha = 1.0$, the value functions have been initialized optimistically to 0.0, and the penalty for failing to fulfill a skill's objective has been set to $r_p = -100$. For OGAHC, the parameter $\rho$ has been varied between 0 and 30, for LGP the minimum number of state observations $t_o$ has been varied between 1 and 30, and the Offline Clustering has been conducted after $m$ steps in which no novel state has been encountered (where $m$ has been varied between 5000 and 50000). Each setting has been evaluated 15 times for 1000 episodes.

The results of the experiment are depicted in Figure 4. The agent provided with predefined skill prototypes obtains an average reward per episode of $\bar{r}_e = -296.3 \pm 1.6$ and the agent using no skills of $\bar{r}_e = -395.5 \pm 0.9$. The maximal gain of average reward that can be achieved by biasing exploration using skills is thus approx. +100. Using LGP for skill discovery does not achieve an average reward of more than $\bar{r}_e = -350$ for any choice of $t_o$. Closer inspection showed that for small values of $t_o$, the resulting partitioning was far from being optimal while for larger values of $t_o$, the skills have been introduced too late to provide a useful exploration bias. Using the Offline Clustering approach, the optimal time for performing the clustering is after $m^* = 30000$ steps without observing any novel state which results in

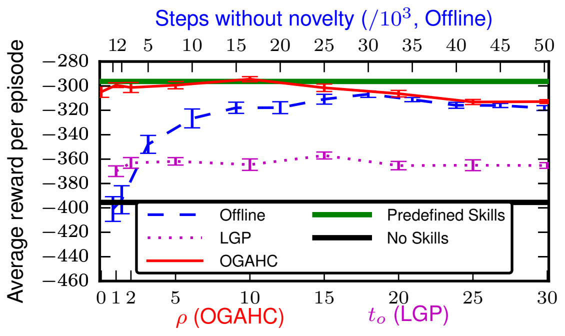

*Figure 4.* Reward per episode (averaged over the first 1000 episodes) for different skill discovery strategies. $\rho$ and $t_o$ are varied over the same value range (bottom x-axis). See supplementary material for more detailed learning curves.

$\bar{r}_e = -306.8 \pm 2.6$. Thus, performing Offline Clustering at the right time does already realize 9/10 of the possible reward gain. However, for the same reasons as in Section 4.2 the specific choice of $m$ is crucial: performing clustering too early ($m \ll m^*$) results in suboptimal partitions while performing it too late ($m \gg m^*$) limits the benefits of the acquired skills for learning.

The OGAHC approach achieves $\bar{r}_e > -305$ for any choice of $\rho \leq 15$ and $\bar{r}_e = -294.5 \pm 2.1$ for $\rho = 10$. The approach allows thus to acquire more than 9/10 of the possible reward gain (and thus more than obtained by Offline Clustering for any choice of $m$) for a broad range of values for $\rho$, making the specific choice of $\rho$ less important. For the optimal choice $\rho^* = 10$, OGAHC achieves an average reward that is on a par with what can be obtained with the predefined skill prototypes. This can be explained by the observation that for this choice of $\rho$, OGAHC discovers skills that are close to the optimal ones very early during learning (typically during the first 3 episodes). Since skill learning takes several episodes, the predefined skill prototypes offer the agent an efficient exploration bias starting after approx. 5 episodes; thus OGAHC can be on a par with the predefined skill prototypes. In summary, the results show that using OGAHC for skill discovery allows to identify meaningful skills at an early stage of learning without requiring to fine-tune the parameter $\rho$ intensely.

## 6. Conclusion and Future Work

We have presented the novel online skill discovery method OGAHC which is based on identifying bottleneck areas in the state transition graph by means of an agglomerative hierarchical clustering. Our empirical studies suggest that the approach can reliably identify useful skills in a very early stage of learning and allows the agent thus to explore the environment more efficiently than with other skill discovery methods or without utilizing skills at all. Future work is to extend the approach to MDPs with continuous state spaces and to evaluate it on real-world problems.

## References

Supplementary material. URL `http://www.informatik.uni-bremen.de/~jhm/files/ewrl_2012_ogahc_supplement.pdf`.

Barto, A. G. and Mahadevan, S. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, October 2003.

Digney, B. L. Learning hierarchical control structures for multiple tasks and changing environments. In *Proceedings of the 5th Conference on the Simulation of Adaptive Behavior*, pp. 321–330. MIT Press, 1998.

Jong, N. K., Hester, T., and Stone, P. The utility of temporal abstraction in reinforcement learning. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 299–306, 2008.

Konidaris, G. and Barto, A. G. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems*, volume 22, pp. 1015–1023, 2009.

Mannor, S., Menache, I., Hoze, A., and Klein, U. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the 21st International Conference on Machine Learning*, pp. 560–567, 2004.

McGovern, A. and Barto, A. G. Automatic discovery of subgoals in reinforcement learning using diverse density. In *In Proceedings of the 18th International Conference on Machine Learning*, pp. 361–368, 2001.

Menache, I., Mannor, S., and Shimkin, N. Q-Cut - dynamic discovery of sub-goals in reinforcement learning. In *Proceedings of the 13th European Conference on Machine Learning*, pp. 295–306, 2002.

Precup, D. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts, Amherst, MA, 2000.

Şimşek, Ö. and Barto, A. G. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *Proceedings of the 21st Interna-*

*tional Conference on Machine Learning*, pp. 751–758, 2004.

Şimşek, Ö. and Barto, A. G. Skill characterization based on betweenness. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, pp. 1497–1504, 2009.

Şimşek, Ö., Wolfe, A. P., and Barto, A. G. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd International Conference on Machine Learning*, pp. 816–823. ACM, 2005.

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