# Hierarchy discovery for planning

Momchil

November 12, 2018

## 1 Motivation

People spontaneously organize their environment into clusters of states which constrain planning [1, 2, 3]. Why and how do we do that?

One normative reason is that hierarchical planning is more efficient in time and memory than "flat" planning [4] (e.g. for BFS, $O(\sqrt[L]{N})$ vs. $O(N)$, where $N$ is the number of states and $L$ is the hierarchy depth). This is consistent with people's limited working memory capacity [5].

But hierarchies are a form of lossy compression, and some hierarchies might result in inefficient planning.

[example]

Solway et al. 2014 [1] provide a formal definition of an "optimal" hierarchy, but they do not specify how the brain might discover it. In this work, we propose a Bayesian cognitive model of hierarchy discovery and show that it is consistent with human behavior.

## 2 Background

### 2.1 Preliminaries

We assume a 2-layer hierarchy ($L = 2$). $G = (V, E)$ is the low-level ("flat") graph that is directly observable, where:

- $V$ is the set of vertices (low-level states)
- $E : \{V \times V\}$ is the set of edges

We define $H = (V', E', c, b, p', p, q)$ as the hierarchy that is not directly observable, where:

- $V'$ is the set of low-level vertices (high-level states, or clusters)
- $E' : \{V' \times V'\}$ is the set of high-level edges

- $c : V \to V'$ are the cluster assignments
- $b : E' \to E$ are the bridges
- $p' \in [0, 1]$ is the density of the high-level graph
- $p \in [0, 1]$ is the within-cluster density of $G$
- $q \in [0, 1]$ is the across-cluster density penalty of $G$

Together $(V', E')$ define the high-level (hierarchical or "H") graph. Each low-level state $i$ is assigned to a cluster $k = c_i$. Each high-level edge $(k, l)$ has a corresponding low-level edge (the bridge) $(i, j) = b_{k,l}$, such that $c_i = k$ and $c_j = l$. For simplicity, we assume both graphs are unweighted and undirected.

## 2.2 Model

Informally, an algorithm that discovers useful hierarchies would satisfy the following desiderata:

1. Favor smaller clusters,
2. ...but not too many of them;
3. Favor dense connectivity within clusters,
4. ...and sparse connectivity across clusters,
5. ...with the exception of "bridges" that connect clusters

Intuitively, having too few (e.g. one) or too many clusters (e.g. each state is its own cluster) creates a degenerate hierarchy that reduces the planning problem to the "flat" scenario, and hence medium-sized clusters are best (desiderata 1 and 2). Additionally, the hierarchy ignores transitions across clusters, which could lead to suboptimal paths generated by the hierarchical planner. It is therefore best to minimize the number of cross-cluster transitions (desiderata 3 and 4). The exception is bridges, which correspond to the links between clusters (desideratum 5).

These can be formalized into a generative model for "good" hierarchies:

$$\begin{aligned}
c &\sim CRP(\alpha) && \text{cluster assignments}\\
p' &\sim Beta(1,1) && \text{H graph density}\\
Pr[(k,l) \in E'] &= p' && \text{H graph edges}\\
Pr[b_{k,l} = (i,j) \mid (k,l) \in E', c_i = k, c_j = l] &= \frac{1}{n_k n_l} && \text{bridges}\\
p &\sim Beta(1,1) && \text{within-cluster density}\\
q &\sim Beta(1,1) && \text{cross-cluster density penalty}\\
Pr[(i,j) \in E \mid c_i = c_j] &= p && \text{within-cluster edges}\\
Pr[(i,j) \in E \mid c_i \neq c_j] &= pq && \text{cross-cluster edges}\\
Pr[(i,j) \in E \mid b_{c_i,c_j} = (i,j)] &= 1 && \text{bridge edges}
\end{aligned}$$

Where $n_k = |\{i : c_i = k\}|$ is the size of cluster $k$ and CRP is the Chinese restaurant process, a nonparametric prior for clusterings [6].

Hierarchy discovery can then be framed as inverting the generative model (in similar spirit to how k-means clustering can be understood as inference over a Gaussian Mixture model). The posterior probability of $H$ is:

$$\begin{aligned}
P(H|G) &\propto P(G|H)P(H) && (1)\\
&= P(E|c,b,p,q)P(p)P(q)P(b|E',c)P(E'|p')P(p')P(c) && (2)
\end{aligned}$$

## 2.3 Tasks

Like prevous authors [1, 3], we assume the agent faces a sequence of tasks in $G$, where each task is to navigate from a starting state $s$ to a goal state $g$. We assume the agent prefers shorter routes.

Informally, the hierarchy discovery algorithm might account for tasks by clustering together states that frequently co-occur in the same task, since hierarchical planning is optimal within clusters.

Defining $tasks = \{task_t\}$ and $task_t = (s_t, g_t)$, we can extend the generative model with:

$$\begin{aligned}
p'' &\sim Beta(1,1) && \text{cross-cluster task penalty}\\
Pr[s_t = i] &= \frac{1}{N} && \text{starting states}\\
Pr[g_t = j \mid s_t = i] &\propto \begin{cases} 1 & \text{if } c_i = c_j \\ p'' & \text{otherwise} \end{cases} && \text{goal states}
\end{aligned}$$

Where $N = |V|$ is the total number of states. We denote the observable data as $D = (tasks, G)$. The posterior then becomes:

$$\begin{aligned}
P(H|D) &\propto P(D|H)P(H) && (3)\\
&= P(tasks|G,H)P(G|H)P(H) && (4)\\
&= \left[\prod_t P(g_t|s_t, p'', G, H)P(s_t|G,H)\right] P(p'')P(G|H)P(H) && (5)
\end{aligned}$$

Where the last two terms are the same as in Eq. 1.

## 2.4 Rewards

To model reward learning, we assume each state delivers stochastic rewards and the agent aims to maximize the total reward [7].

Informally, the hierarchicy discovery algorithm might account for rewards by clustering together states that deliver similar rewards. This is consistent with the tendency for humans to cluster based on perceptual features [3] and would be rational in an environment with autocorrelation in the reward structure.

We can incorporate these intuitions into the generative model as:

$$\begin{aligned}
\theta_k &\sim \mathcal{N}(0, 100) && \text{average cluster rewards}\\
\mu_i &\sim \mathcal{N}(\theta_{c_i}, 100) && \text{average state rewards}\\
r_{i,t} &\sim \mathcal{N}(\mu_i, \sigma_r^2) && \text{rewards}
\end{aligned}$$

Where $k \in V'$ and $i \in V$. $\theta_k$ is the average reward delivered by states in cluster $k$, $\mu_i$ is the average reward delivered by state $i$, $r_{i,t}$ is the actual reward delivered by state $i$ at time $t$, and $\sigma_r^2$ is the variance of that reward. For simplicity, we assume $\sigma_r^2 = 0$, i.e. constant rewards for each state ($r_{i,t} = \mu_i$).

## 2.5 Inference

### 2.5.1 Offline

We approximate Bayesian inference over $H$ using Metropolis-within-Gibbs sampling [8] (a kind of MCMC) which updates each component of $H$ in turn by sampling from its posterior conditioned on all other components in a single Metropolis-Hastings step. The proposal distribution for continuous components is a Gaussian random walk. The proposal distribution for cluster assignments $c_i$ is the conditional CRP prior (algorithm 5 in [9]).

Our approach can also be interpreted as stochastic hill climbing with respect to a utility function defined by the posterior. This has been previously used to find useful hierarchies for robot navigation [4].

### 2.5.2 Online

To make trial-by-trial predictions, we separately used an importance sampling [10] approximation together with particle filtering [11, 12]. We approximate the posterior with a set of $M$ samples (particles) $[H^{(1)}, ..., H^{(M)}]$ drawn from a proposal distribution $Q(H)$. Each particle is assigned an importance weight according to:

$$w^{(m)} \propto \frac{P(D|H^{(m)})P(H^{(m)})}{Q(H^{(m)})} \tag{6}$$

Where $D$ is the data observed so far. When a new data point $d$ is observed at time $t$, the weights are updated to:

$$w_t^{(m)} \propto P(d|H^{(m)})w_{t-1}^{(m)} \tag{7}$$

The sum of the weights is then normalized to $\sum_m w_t^{(m)} = 1$. A draw from $[H^{(1)}, ..., H^{(M)}]$ with multinomial probabilities $[w^{(1)}, ..., w^{(M)}]$ then approximates a draw from the posterior of $H$, with convergence to the true posterior as $M \to \infty$. Our proposal distribution $Q(H)$ was simply the prior $P(H)$, and thus all weights were initialized to $w_0^{(m)} = 1/M$.

Since the initial samples might turn out to be a poor fit to the data $D$, we introduced a rejuvination step [13] that applies several iterations of Metropolis-within-Gibbs to each particle.

## 2.6 Decision making

Here we describe the linking assumptions we use to make predictions about human choices based on $H$.

### 2.6.1 Hierarchical planning

The optimal algorithm to find the shortest path between a pair of low-level vertices $(s, g)$ in $G$ is breadth-first search (BFS) [14] whose time and memory complexity is $O(N)$ (assuming $O(1)$ vertex degrees, i.e. $|E| = O(N)$). We use a natural extension of BFS to hierarchical graphs [4] (hierarchical BFS or HBFS) that leverages $H$ to find paths more efficiently than BFS (approximately $O(\sqrt{N})$ time and memory). Intuitively, HBFS works by first finding a high-level path between the clusters of $s$ and $g$, $c_s$ and $c_g$, and then finding a low-level path within the cluster of $s$ between $s$ and the first bridge on the high-level path.

In particular, HBFS first finds a high-level path $(k_1, ..., k_{m'})$ between $c_s$ and $c_g$ in the H graph $(V', E')$ (note that $k_1 = c_s$ and $k_{m'} = c_g$). Then it finds a low-level path $(v_1, ..., v_m)$ between $s$ and $i$ in $G[S]$, where $(i, j) = b_{k_1,k_2}$ is the first bridge on the high-level path, $S = \{v : c_v = c_s\}$ is the set of all low-level vertices in the same cluster as $s$, and $G[S]$ is the subgraph induced in $G$ by $S$. HBFS then returns $v_2$, the next vertex to move to from $s = v_1$.

In an efficient hierarchy, the number of clusters will be $|V'| = O(\sqrt{N})$ and the size of each cluster $k$ will also be $n_k = O(\sqrt{N})$, resulting in $O(\sqrt{N})$ time and memory complexity for HBFS. Note that actually traversing the full low-level path from $s$ to $g$ in $G$ still takes $O(N)$ time; HBFS simply computes the next step, ensuring the agent can progress towards the goal without computing the full low-level path in advance. HBFS can straightforwardly extend to deeper hierarchies with $L > 2$, with the corresponding complexity becoming $O(\sqrt[L]{N})$.

The pseudocode for HBFS is in [Box 1]. Note that we are not making any specific commitments to the cognitive plausibility of HBFS and that any other hierarchical planner would make similar predictions.

### 2.6.2 Edge unveiling

Drawing on the active learning framework from the causal inference literature [15, 16], we assume the agent will chose to learn about ("unveil") edges of $G$ in a way that provides maximal information about $H$. Maximizing information about $H$ is equivalent to minimizing uncertainty about $H$, which can be quantified as the entropy of $H$:

$$\mathbb{H}(H|D) = -\sum_{H} \int P(H|D) \log P(H|D) dH \tag{8}$$

Where the sum is over the discrete components of $H$ and the integral is over the continuous components of $H$. $D$ is the data observed so far. We use $\mathbb{H}$ to denote the entropy of a mixed random variable with discrete and continuous components [17].

If the posterior is approximated by a set of samples $[H^{(1)}, ..., H^{(M)}]$ with normalized importance weights $[w^{(1)}, ..., w^{(M)}]$, we can approximate the entropy as:

$$\mathbb{H}(H|D) \approx -\sum_{m} w^{(m)} \log P(H^{(m)}|D) \tag{9}$$

We use $a = (i, j)$ to denote the action of unveiling edge $(i, j)$, i.e. observing whether $(i, j) \in E$. Since there is no way to know in advance what the outcome would be, the agent has to minimize the expected entropy over the two possible outcomes:

$$\begin{aligned} \mathbb{H}(H|D, a) = \mathbb{H}(H|D \cup (i, j) \in E) Pr[(i, j) \in E|D] \qquad (10) \\ + \mathbb{H}(H|D \cup (i, j) \notin E) Pr[(i, j) \notin E|D] \qquad (11) \end{aligned}$$

Where we can use the sampling approximation to compute the probability of each outcome by marginalizing over $H$:

$$Pr[(i,j) \in E|D] = \sum_H \int Pr[(i,j) \in E|H] P(H|D) dH \tag{12}$$
$$\approx \sum_m Pr[(i,j) \in E|H] w^{(m)} \tag{13}$$

Where $Pr[(i,j) \in E|H]$ is $p$ or $pq$, according to the generative model. $Pr[(i,j) \in E|H]$ is computed analogously.

The agent then chooses the action that minimizes the expected entropy:

$$a = \arg\min_a \mathbb{H}(H|D,a) \tag{14}$$

## Bibliography & References Cited

[1] Alec Solway, Carlos Diuk, Natalia Córdova, Debbie Yee, Andrew G. Barto, Yael Niv, and Matthew M. Botvinick. Optimal behavioral hierarchy. *PLOS Computational Biology*, 10(8):1–10, 08 2014.

[2] Anna C. Schapiro, Timothy T. Rogers, Natalia I. Cordova, Nicholas B. Turk-Browne, and Matthew M. Botvinick. Neural representations of events arise from temporal community structure. *Nature Neuroscience*, 16(4):486–492, Apr 2013.

[3] Jan Balaguer, Hugo Spiers, Demis Hassabis, and Christopher Summerfield. Neural mechanisms of hierarchical planning in a virtual subway network. *Neuron*, 90(4):893–903, 2016.

[4] Juan A Fernández and Javier González. *Multi-hierarchical representation of large-scale space: Applications to mobile robots*, volume 24. Springer Science & Business Media, 2013.

[5] George A Miller. The magic number seven plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63:91–97, 1956.

[6] Samuel J Gershman and David M Blei. A tutorial on bayesian nonparametric models. *Journal of Mathematical Psychology*, 56(1):1–12, 2012.

[7] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[8] Gareth O Roberts and Jeffrey S Rosenthal. Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349–367, 2009.

[9] Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

[10] Christian P Robert. Casella: Monte carlo statistical methods. *Springerverlag, New York*, 3, 2004.

[11] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In *Sequential Monte Carlo methods in practice*, pages 3–14. Springer, 2001.

[12] Pratiksha Thaker, Joshua B Tenenbaum, and Samuel J Gershman. Online learning of symbolic concepts. *Journal of Mathematical Psychology*, 77:10–20, 2017.

[13] Nicolas Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, 2002.

[14] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[15] Kevin P Murphy. Active learning of causal bayes net structure. 2001.

[16] Simon Tong and Daphne Koller. Active learning for structure in bayesian networks. In *International joint conference on artificial intelligence*, volume 17, pages 863–869. Citeseer, 2001.

[17] Chandra Nair, Balaji Prabhakar, and Devavrat Shah. On entropy for mixtures of discrete and continuous variables. *arXiv preprint cs/0607075*, 2006.