Hierarchical clustering optimizes the tradeoff between compositionality and expressivity of task structures in reinforcement learning

Rex G Liu and Michael J Frank
Carney Institute for Brain Science
Department of Cognitive, Linguistic, & Psychological Sciences
Brown University
Providence, RI 02912
{rex_liu, michael_frank}@brown.edu

Abstract

A hallmark of human intelligence is our ability to compositionally generalise: that is, to recompose familiar knowledge components in novel ways to solve new problems. For instance, a talented musician can conceivably transfer her knowledge of flute fingerings and guitar songs to play guitar music on a piccolo for the first time. Yet there are also instances where it can be helpful to learn and transfer not just individual task components, but entire structures or substructures, particularly whenever these recur in natural tasks (e.g., in bluegrass music one might transfer the joint structure of finger movements and musical scales from one stringed instrument to another). Prior theoretical work has explored how agents can learn and generalize task components or entire latent structures, but a satisfactory account for how a single agent can simultaneously satisfy the two competing demands is still lacking. Here, we propose a hierarchical model-based agent that learns and transfers individual task components as well as entire structures by inferring both through a non-parametric Bayesian model of the task. It maintains a factorised representation of task components through a hierarchical Dirichlet process, but it also represents different possible covariances between these components through a standard Dirichlet process. We validate our approach on a variety of navigation tasks covering a wide range of statistical correlations between task components and show that this hierarchical framework can also be applied to improve generalisation and transfer in hierarchical tasks with goal/subgoal structures.

1 Introduction

Humans have a remarkable ability to quickly solve novel problems by leveraging knowledge acquired from past experiences that bear some meaningful similarity to the problem at hand. One of the key ways in which we do this is by structural generalization: we are able to abstract out latent task structures – independent of the task's particularities, such as its specific context or sensory representation – and transfer them to new problems that may share these structures but instantiate them in unfamiliar ways [1, 2]. For instance, a pianist knows that the fingerings, scales, arpeggios, pedalling, polyrhythms and other techniques she has acquired over years of practice are not specific to the particular piano that she practiced on and can therefore readily perform on a piano in Carnegie Hall, and indeed even transfer many of these skills to different instruments entirely, such as the harpsichord, organ or, to some extent, the glass harmonica. Such capacity for abstraction has also been observed in non-human primates, as first noted by Harlow in his account of "learning set" acquisition [3]. In contrast, artificial agents, in spite of their impressive ability to now outperform humans on a wide range of difficult tasks [4, 5, 6, 7], are notoriously brittle and struggle with similar knowledge transfer [8, 9].

The acquisition and transfer of latent task structures can be understood normatively as a problem of non-parametric Bayesian inference. When solving a new task, the subject will attempt to infer from observations what latent structure governs the task statistics. If the statistics are consistent with a structure the subject

has seen previously, the subject can simply leverage its knowledge of that structure to quickly solve the current task, and transfer learning thus takes place. Otherwise, a new structure can be created and added to the pool of task structures that can be leveraged in the future. This framework has offered an account for how humans are able to generalize over multiple structures without interfering with each other. This framework was originally applied to modelling simple context-dependent Pavlovian conditioning [10], but has been extended to capture learning and generalization of entire instrumental "task sets" of stimulus-response mappings [11, 1], to predictive representations of future states in sequential decision-making tasks [12], and to deeper latent state abstractions in which new tasks can be learned by analogy with previous tasks [13]. Models of hierarchically nested cortico-striatal gating circuits have offered a mechanistic account for how the brain might approximate such structure learning and transfer [14, 11], and human studies have found signatures of latent task structures in the brain [15, 16, 1, 17, 2, 18].

A limitation of the above theoretical framework is that it only considered how agents could learn and reuse task structures as a whole. Similarly, artificial intelligence agents generally treat tasks as monolithic entities without any substructure. In contrast, most naturalistic tasks have a compositional structure, which natural intelligence readily learns and exploits. Indeed, one of the hallmarks of human intelligence is our ability to compositionally generalize, that is, to learn and generalize over individual task components and then recompose them in novel ways to solve novel tasks. A consummate musician who can play the flute and ukelele might reasonably play ukulele music on the piccolo on a first attempt by transferring knowledge of both flute finger movements and ukulele songs.

Recent work has extended the Bayesian framework of structure learning to allow for compositionality and to compare to patterns of human generalization [19, 20]. In this extension, the agent maintains a factorized representation of structural components (for example, the *transition function*, akin to the finger movements, and the *reward function*, akin to the desired songs). When encountering a novel context, the agent attempts to infer each of the structural components independently of the others, allowing it to recompose task structures, akin to playing the piccolo for the first time in the above example.

Note, however, that it is not always appropriate to treat each task component as statistically independent of the others. For instance, a musician playing the sitar is far more likely to play music from the Hindustani tradition than a four-voice keyboard fugue by Bach. Indeed, the relative benefit of jointly vs independently clustering task components depends on their mutual information and noise in the environment [19]. Previous theoretical work therefore considered a mixture of experts or "meta-agent" architecture in which an agent infers whether a task is better described by joint or independent statistics and then generalizes accordingly in new environments [19], as do human subjects [20]. Such a meta-agent is, however, limited: most naturalistic tasks are not simply joint or independent but may require a more powerful algorithm to discover, for example, joint sub-clusters embedded within an otherwise independent structure. For instance, although a large family of stringed instruments can be used to play a wide range of musical genres, there will often be covariances between the genre and the instrument, such as the electric guitar with metal music, the cello with Western classical, or the koto zither with Japanese gagaku. Moreover, the covariances can often have a hierarchical structure to them. For instance, the pairing of music and instruments may change depending on the venue and context in which the musician performs. At a jazz festival, she is not only more likely to play jazz rather than classical music, but the choice of jazz music will depend not only on her chosen instrument but also on whether she is playing in a jazz band, a small group, or by herself; likewise, when she is at a classical music festival, she will more likely play classical instead of jazz music, but the type of classical music will also change according to not only her choice of instrument but also to whether she is playing with a symphony orchestra, a chamber ensemble, or as a soloist.

Here, we develop a novel computational strategy to accommodate a wider range of correlational structures. The meta agent described could only arbitrate between independent and joint structures; its only recourse for tasks that did not conform to these pure structures was simply to blend them (weight one more than the other) at the policy level. We introduce a single, more powerful, clustering agent based on the hierarchical Dirichlet process [21] that can learn the appropriate degree of covariance in a context-dependent fashion. Like the independent agent, our agent maintains a factorized representation over task components, allowing for components to be learnt and generalized independently of each other. Simultaneously, it also estimates the covariance between these components, thereby affording generalization over a wider range of compositional structures. We validate our approach on a variety of tasks with independent, joint, and mixed statistics and show that only the hierarchical agent is able to perform well across all tasks. Finally, we show that the

model's hierarchical structure can be immediately applied to improve generalization and transfer in more complex tasks with hierarchical goal/subgoal structures.

1.1 Related work

Perhaps one of the earliest examples of using a hierarchical generative model to perform multi-task reinforcement learning is [22]. As in our work, they consider the setting where different Markov decision processes (MDPs) are generated by a hierarchical process and thus may share commonalities in the transition or reward function. At the top level, one of a possibly infinite number of MDP classes is chosen, where each class generates MDPs that have similar reward and transition functions. However, the work only considers the ability transfer across task structures as a whole (i.e., the joint reward-transition function). In contrast, by applying Bayesian non-parametrics across the entire model hierarchy (as opposed to just the top-level), our agent is endowed with the ability to generalize MDP components individually while also learning separately the structural relationships between them. To our knowledge, applying hierarchical Bayesian non-parametrics to multi-task reinforcement learning has not been previously explored.

Meta-learning offers another approach to multi-task reinforcement learning [23, 24, 25, 26, 27]. In this approach, the agent is meta-trained over a fixed task distribution and, at meta-test time, must quickly adapt to a novel task sampled from the same distribution. The agent's behaviour at meta-test time can also be understood as performing a form of amortized Bayesian inference over task identity, with knowledge about task commonalities learnt during meta-training serving as an implicit prior to support fast adaptation [25, 27]. However, this approach is limited to a fixed task distribution, making it less suitable, in its current form, for continual online learning, something which Bayesian non-parametrics helps make possible in our approach. Moreover, again, this approach has, to date, still only considered tasks as single entities.

In the cognitive sciences, there has also been a recent tradition of using Bayesian non-parametrics to model the learning of latent task structures. [10] considered the use of a Dirichlet process to model contextual Pavlovian conditioning, where animals must infer the correct relationship between stimuli and outcomes depending on the latent context. [11] extended this to instrumental conditioning, where agents must infer latent "task set" rules that govern the relationship between stimuli, actions, and outcomes. However, none of these cases considered sequential decision-making tasks.

Perhaps the most relevant work is [19], which extended this line of work in several ways. First, they considered model-based sequential decision-making, where an agent must infer the correct task model to support planning. Secondly, they considered tasks that had a compositional structure to them (specifically, dissociable reward and transition functions); an agent that compositionally generalizes could, in principle, leverage this by reusing familiar components to construct new task models. Thirdly, they introduced an agent that learns components independently of each other, a necessary ingredient for compositional generalization. While their agents were able to compositionally generalize when components were independent of each other across tasks, they could be suboptimal when the components were related to each other; in these cases, an agent that is not compositional could actually perform better. What is missing is the ability to learn and transfer higher order structures between components in addition to the individual components. Our work introduces a unified framework where an agent learns and transfers individual components at the same time as higher order structures. Methodologically, this is done by introducing a hierarchical Dirichlet process to model the task's generative process. This framework gives the agent the flexibility needed to adapt to a much wider range of task statistics. One final difference between [19] and our work is that they only considered tasks with non-hierarchical structures whereas we show how our approach can scale to tasks with a goal/subgoal structure.

1.2 Paper layout

This paper is organised as follows. In the first half, we focus on non-hierarchical tasks to illustrate the utility of our framework in simple settings. In section 2, we explain the structure of the tasks we shall consider as well as the architecture of the various agents we shall use. Section 2.1 explains the gridworld tasks we work with, while section 2.2 explains the agent architectures. Section 3 presents the results of our simulations. In section 4, we move on to hierarchical tasks, again presenting the task structure, the agent architectures, and the results of our simulations. We conclude with a discussion of our results in section 5.

2 Task structure and agent architectures

2.1 Gridworld task structure

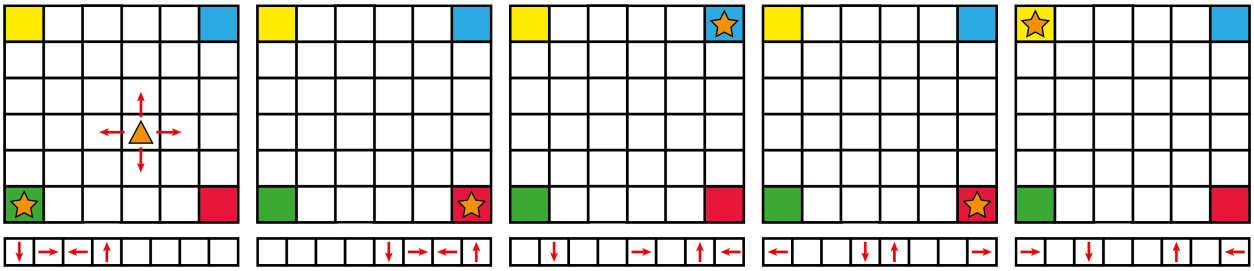


Figure 1: Gridworld task: the agent (triangle) navigates a series of gridworld rooms where it can move in any of the four cardinal directions (red arrows). Each room has a hidden goal state (star) located in one of the colored squares. The agent moves by pushing one of eight possible keys, but the mappings from keys to cardinal movements (bottom) vary from room to room.

We first consider a series of gridworld tasks that are explicitly compositional in structure, based on those described in [19]. The agent has to learn how to navigate towards a goal in a series of gridworld rooms (Fig. 1), where it has to learn both the transition function (which actions will take it from one location to another, analogous to the relationship between frets/strings of a guitar and the notes they produce) and the reward function (the rewarding locations or goals, analogous to the desired song or scale). A flexible agent will be able to recognize when it can reuse a previously observed transition function in the service of reaching a different goal (i.e., if these are independent across rooms), but could also infer when a given transition function is informative about the likely goal location. The agent starts from a random location in each room and must navigate to a hidden goal by moving along the four cardinal directions. To move, the agent selects one of eight possible actions, hereafter referred to as "keys" based on corresponding human experiments in which subjects had to move by pressing keys on a keyboard [20]. Only four of the keys will move the agent while the others will do nothing. Moreover, the mapping from keys to movements (i.e., which key takes the agent north, south, east and west) varies from room to room and is unknown to the agent – it must be learnt from its interactions with the environment. The goal is also hidden from the agent so it must also learn the reward function from its interactions with the rooms. The goal appears in one of four possible locations, but the agent is not endowed with that information either.

Formally, the task can be modelled as a Markov decision process (MDP) consisting of the quintuple $(\mathcal{S}, \mathcal{A}, \gamma, \mathcal{T}, \mathcal{R})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\gamma \in [0, 1]$ is the discount factor, $\mathcal{T}$ is the state transition dynamics, and $\mathcal{R}(s', a, s)$ is the reward function. A state $s \in \mathcal{S}$ consists of a tuple (z, c) where $c \in \mathbb{N}$ is the room's context and $z = (x, y)$ is the agent's position in the room. The action space $\mathcal{A}$ comprises the set of eight possible keys. In our tasks, the agent receives reward only as a function of its position in the room and not how it moves through space. Thus, the reward function $\mathcal{R}(s', a, s)$ is independent of actions and takes the form

$$\mathcal{R}(s', a, s) = \mathcal{R}(z'|c) = \begin{cases} 1 & \text{if a goal is located at } z', \\ 0 & \text{otherwise,} \end{cases} \tag{1}$$

with the conditioning on c reflecting that the goal location depends on the context c (i.e., the room the agent is in). The transition dynamics $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ maps states to successor states conditioned on the key press.

The transition function $\mathcal{T}(s', a, s) = p(s'|s, a)$ expresses the probability of arriving in state s' when taking action a in state s. For our tasks, this can be factorized into two components. The first component expresses state transitions in terms of the cardinal movements, namely $\mathcal{A}_{card} = \{up,\ down,\ left,\ right,\ stay\}$ and abstracts out the transition's dependence on the room's spatial structure. As all rooms share the same structure, this component does not depend on context and therefore takes the form $p(z'|z, a_{card})$. The second component describes how motor actions in the form of key presses map onto spatial movements and

takes the form

$$\Phi(a_{card}|a,c) = \begin{cases} 1 & \text{if key press } a \text{ induces motion along } a_{card} \text{ in context } c, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

This component introduces context-dependence, reflecting the fact that mappings can depend on environmental conditions or other contextual factors; for instance, the note produced on a guitar depends on its tuning, and an aerial drone navigating to a goal will need to adjust its motor outputs depending on wind conditions. Composed together, the full transition function is

$$\mathcal{T}(s',a,s) = p\left(z'|z,a_{card}\right)\Phi(a_{card}|a,c). \tag{3}$$

In these tasks, the agent must learn $\mathcal{R}$ and Φ from its interactions with the rooms (all other MDP components are given). If the goal is known, the agent can plan the optimal path to it in terms of spatial movements. The optimal policy is obtained by choosing movements that maximize the optimal action-value function $Q_c^*(z',a_{card})$,

$$\pi(a_{card}|z,c) = \arg\max_{a_{card}} Q_c^*(z,a_{card}), \tag{4}$$

where $Q_c^*(z',a_{card})$ is the solution to the Bellman optimality equation

$$Q_c^*(z,a_{card}) = \sum_{z'} p(z'|z,a_{card})\left[\mathcal{R}(z'|c) + \max_{a'_{card}} \gamma\, Q_c^*(z',a'_{card})\right]. \tag{5}$$

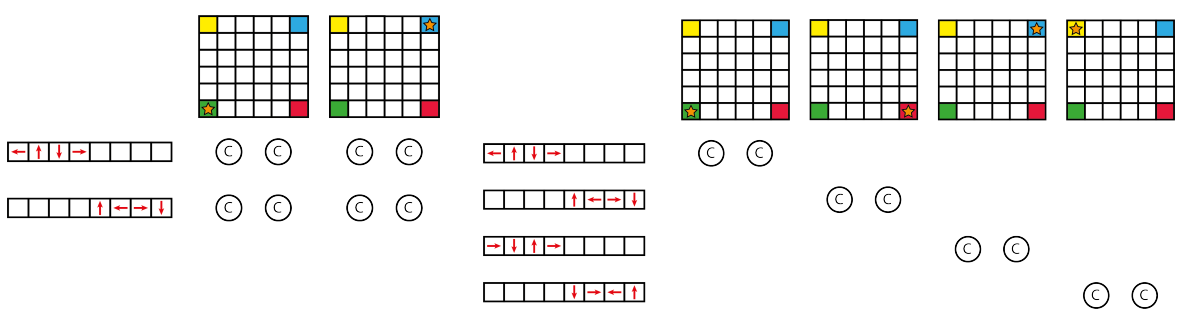
Both $Q_c^*(z',a_{card})$ and $\pi(a_{card}|z,c)$ can be obtained by model-based policy iteration. If Φ is known, the agent can then implement this policy by selecting the appropriate keys. Combining (2) with (4), one can show that this solves the Bellman optimality equation in terms of the key presses $a \in \mathcal{A}$,

$$\pi(a|z,c) = \arg\max_{a} Q_c^*(z,a) = \arg\max_{a} \sum_{a_{card}} \Phi(a_{card}|a,c)\, Q_c^*(z,a_{card}), \tag{6}$$

where, by an abuse of notation, we use $\pi(\cdot|c)$ and $Q_c^*(z,\cdot)$ to denote the policy and action-value functions in terms of cardinal movements or key presses according to whether the argument is an action from $\mathcal{A}_{card}$ or $\mathcal{A}$. The action-value functions for a are easily obtained from those for a_{card}:

$$Q_c^*(z,a) = \sum_{a_{card}} \Phi(a_{card}|a,c)\, Q_c^*(z,a_{card}). \tag{7}$$

Thus, the reinforcement learning problem critically reduces to one of inferring the correct $\mathcal{R}$ and Φ.



(a) Task with independent components

(b) Task with joint statistics

Figure 2: Pairings of mappings and goals (orange star) used in the contexts (C) of the independent and joint statistics tasks.

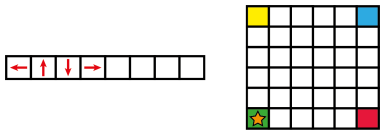
As the agent moves from room to room, functions Φ and $\mathcal{R}$ may repeat independently of each other, and agents that leverage knowledge about familiar Φ and $\mathcal{R}$'s will exhibit an advantage to quickly solve novel

rooms. However, depending on the task statistics, Φ's and $\mathcal{R}$'s can also be correlated, so there will also be a distinct advantage towards agents that can learn and exploit this correlational structure.

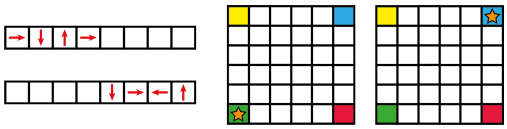
To explore the possibility of transferring different correlational structures, we consider three versions of this task. In the first version, mapping and reward functions are counterbalanced to give statistical independence between them (Fig. 2a). Two distinct mapping functions are paired with two distinct goal locations, giving a total of four unique pairings. Each pairing appears as two distinct room contexts, and each context repeats four times across the entire task. Contexts, representing e.g., a given room, are given explicitly to the agent and are presented in random order. In this task, an agent that can exploit the task's compositional structure by independently clustering and transferring the reward and transition functions will exhibit an advantage, as shown by [19]. In contrast, agents that only transfer the entire model (the joint transition-reward functions) would perform suboptimally when these functions reappear in novel combinations.

In the second version, action mappings are entirely predictive of the goal location (Fig. 2b). This version uses four distinct mappings and four distinct goals, but each mapping is paired with only one of the goals, giving four unique mapping-goal pairings. Again, each pairing appears as two distinct contexts, each context repeats four times, and contexts are presented in random order. This task will favor an agent that can exploit the one-to-one correspondence between goals and mappings. Indeed, an agent that learns goals and mappings as a single structure will outperform one that treats them as independent components [19].

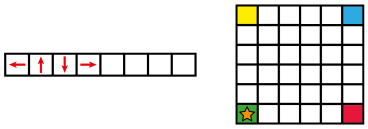
Note that the above task versions represent the two extremes: the mappings and goals were either in a strict one-to-one relationship or were completely independent of each other. The third, arguably more naturalistic, task has a more mixed relationship such that neither an agent that regards the two components as independent nor one that treats them as a joint unit can be optimal.

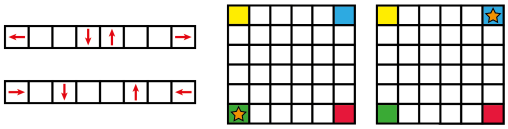


(a) Joint contexts consist of a single mapping-goal pairing, such that the mapping is perfectly predictive of the goal.



(b) Independent training contexts consist of two distinct mappings (different from that in the joint contexts), and two distinct goals (one of which is the same as in the joint context). The mappings are not always predictive of the goals.





(c) Test contexts consist of two types. One context (left) is a joint context and repeats the same mapping-goal pair as in Fig. 3a. The other two contexts (right) are more independent contexts and use two novel mapping functions but the same two goals as in (Fig. 3b).

Figure 3: Task with mixed statistics

In this task, the agent first sees a series of "training" contexts. These belong to two types, one where the mappings and goals are in an one-to-one relation and another where the mappings are not entirely predictive of the goals. We shall refer to first type as *joint* contexts (Fig. 3a) and the second as *independent* contexts (Fig. 3b). After visiting all training contexts, the agent sees a series of "test" contexts next to assess its ability to transfer (Fig. 3c). The entire task is designed so that an agent that treats mapping and reward functions as a joint unit will be optimal in the joint test contexts but not the independent ones, while the reverse will be true of an agent that treats the two as independent components. Only the agent that is introduced below is able to perform optimally across all contexts: it can learn and separate out joint and independent substructures within the task and reuse those substructures whenever they reappear. In contrast, a meta-agent strategy [19] would not be efficient here, as it is only capable producing an overall blending of independent and joint clustering across the task, but cannot discover substructures.

2.2 Clustering agents

2.2.1 Independent, joint, and meta-agents

Past work in both Pavlovian [10] and instrumental conditioning [11, 14] has cast context-dependent generalisation as a clustering problem. In this framework, different contexts are clustered together into a latent generative task state if they share similar task statistics. This mechanism allows an agent to maintain distinct task representations without them interfering with one another, but also allows for rapid transfer of knowledge. Should a novel task share similar statistics with previously encountered tasks, the agent can recognize this similarity to reuse the latent task state and quickly solve the novel task. This framework has been extended to allow for compositional generalisation [19, 20], and to allow for reuse of compressed representations of abstract task states even when reward and transition functions change [13].

Formally, cluster assignment is framed as a Bayesian inference problem where the agent must infer which latent cluster is generating the current task statistics. To allow for the possibility of an infinite number of clusters, a Dirichlet process (DP) prior is used for cluster assignment. This does not require specifying the total number of clusters beforehand, making it a non-parametric Bayesian model.

To impose some parsimony in the prior cluster assignment, we assume that those clusters that are most popular across contexts are more likely to apply in a new context [10, 11], consistent with human subject priors during generalization [1]. Specifically, we work with the Chinese restaurant process (CRP) representation of the Dirichlet process. In this framework, a context will be assigned to a pre-existing cluster with a prior probability dependent on the past popularity of that cluster. If contexts $c_{1:n}$ have been assigned to clusters $\{1, \ldots, K\}$, where $K \leq n$, then the prior probability $P(c_{n+1} \in k)$ that new context c_{n+1} belongs to cluster k is

$$P(c_{n+1} \in k) = \begin{cases} \frac{n_k}{n+\alpha} & \text{if } k \text{ is a pre-existing cluster } (k \leq K) \\ \frac{\alpha}{n+\alpha} & \text{if } k \text{ is a new cluster } (k = K+1), \end{cases} \tag{8}$$

where n_k is the total number of contexts already assigned to cluster k and α is a hyperparameter that controls the probability of assigning the context to a novel cluster. As data $\mathcal{D}$ is accumulated, cluster assignment is updated via Bayesian inference,

$$P(c_{n+1} \in k|\mathcal{D}) \propto P(\mathcal{D}|c_{n+1} \in k)P(c_{n+1} \in k).$$

Compositional generalisation requires being able to generalize task components (e.g., transition and reward functions) independently of one another, and a factorized representation of task components is necessary to support this. Previous work [19, 20] has accomplished this by maintaining two independent clusterings of contexts, one that clusters contexts according to whether they share the same transition function Φ and another according to whether they share the same reward function $\mathcal{R}$ (Fig. 4a). In this case, two separate Dirichlet processes are maintained for inferring clusterings according to Φ and $\mathcal{R}$. For a given mapping cluster k_Φ, the likelihood function is

$$P(a_{card}|a, c \in k_\Phi) = \Phi(a_{card}|a, c \in k_\Phi), \tag{9}$$

modelled as a Bernoulli probability function. For a given reward cluster $k_\mathcal{R}$, the reward likelihood function $\mathcal{R}(z'|c)$ is also modelled as a Bernoulli probability function,

$$\mathcal{R}(z'|c \in k_\mathcal{R}) = P(r|z, c \in k_\mathcal{R}), \tag{10}$$

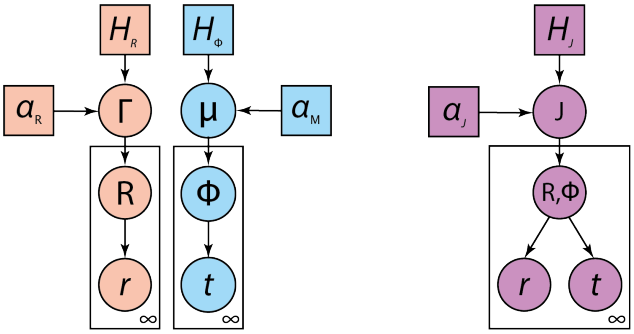
which gives the probability of receiving reward r in state z. Following [19], we shall refer to an agent which clusters contexts in this way as an *independent clustering agent*.

The alternative would be to cluster Φ and $\mathcal{R}$ jointly (Fig. 4b), much like what many previous non-compositional agents do. In this case, the likelihood function for a transition (z, a, a_{card}, z', r) is

$$P(r, z', a_{card}|z, a, c \in k) = \mathcal{R}(z'|c \in k)\, \Phi(a_{card}|a, c \in k), \tag{11}$$

where, again, $\Phi(a_{card}|a, c \in k)$ and $\mathcal{R}(z'|c \in k)$ are modelled as Bernoulli probability functions. As in [19], we shall refer to this as a *joint clustering agent*.

As noted above, an agent that clusters components independently can compositionally generalize to new tasks but will fail to leverage statistical structure when the reward and transition functions are informative of each other. In gridworld tasks with sparse rewards, an agent will generally learn Φ first and only learn $\mathcal{R}$ when it reaches a goal. There will therefore be a distinct advantage towards agents that can predict $\mathcal{R}$ from Φ, when such a predictive relationship exists. In the extreme case where Φ is entirely predictive of $\mathcal{R}$, as in Fig. 2b, the joint clustering agent will outperform the independent clustering agent. Humans are capable of flexibly adapting to both types of task statistics [20], which raises the question of how this flexibility can be captured in artificial agents. To address this, [19] developed a meta-agent that infers whether the task statistics are better modelled with independent or joint clustering models, and then acting accordingly,



(a) Independent clustering agent (b) Joint clustering agent

Figure 4: The independent clustering agent (a) maintains separate clusterings of rewards (red) and mappings (blue). A base distribution $H_\mathcal{R}$ generates a clustering Γ of reward functions $\mathcal{R}$, which in turn generates the rewards r observed in the environment. A separate base distribution H_Φ generates clusterings μ of mappings Φ, which in turn generates the observed transitions t. In the joint clustering agent (b), a single base distribution H_J generates clusterings J of reward functions $\mathcal{R}$ paired with mappings Φ. These, in turn, generate r and t.

$$P(\mathcal{M}_i|\mathcal{D}) \propto P(\mathcal{D}|\mathcal{M}_i = Indep)P(\mathcal{M}_i = Indep) + P(\mathcal{D}|\mathcal{M}_i = Joint)P(\mathcal{M}_i = Joint), \tag{12}$$

where $\mathcal{M}_i \in \{Indep, Joint\}$ denotes the model type. The main limitation with this approach is that all tasks get projected onto either joint or independent statistics, yet most naturalistic tasks have statistics that fall somewhere in between. In particular, tasks may actually include both joint and independent substructures existing alongside each other, and such an agent would fail to discover and reuse these substructures. Further, the meta-agent requires maintaining two separate clustering agents and a third mechanism for arbitrating between them.

Here, we develop a novel *hierarchical agent*, using the hierarchical dirichlet process, that includes an expanded hypothesis space covering a broader range of statistical structures between Φ and $\mathcal{R}$ over which it can perform inference, and which includes independent and joint clustering as special cases.

2.2.2 Hierarchical agent

The key idea behind the hierarchical agent is to have it learn and transfer task *components* as well as any statistical relationship between those components (generative *structures* that may produce them jointly). In our navigation task, the different rooms can be regarded as belonging to different room types that generate different statistics in Φ and $\mathcal{R}$. The idea behind the hierarchical agent (Fig. 5a) is to invert this generative model and cluster contexts not just according to mapping and reward functions but also according to the room types that generated them. In this way, the hierarchical agent clusters contexts not just according to task components but also according to generative structures.

The hierarchical agent introduces, at the highest level, a DP for clustering contexts by "room types" (note the same framework could be used for any type of MDP, but here we focus on rooms in gridworlds). Each room cluster defines its own pair of DPs for clustering reward and mapping functions. These room-specific DPs model distributions over Φ and $\mathcal{R}$ that are in turn conditional on the room type. Because different room types can induce different distributions over Φ and $\mathcal{R}$, statistical associations between Φ and $\mathcal{R}$ may be prevalent in some room types (that is, Φ may be predictive of $\mathcal{R}$). The agent can also leverage information about the prevalence of each mapping and reward function across the entire environment (all rooms); indeed, it additionally maintains a pair of DPs outside the rooms that cluster reward and mapping functions across the environment, and the room-specific clusters are assumed to be drawn from these environment-wide clusters, hence forming a hierarchical process.

Note that this is a fundamentally different solution than the meta-agent, which only tries to infer whether an entire task domain is independent or joint, but cannot discover substructures within tasks. As an analogy, in the musical instrument example, the hierarchical agent can learn about types of instruments in which physical movements and desired songs tend to co-occur, while simultaneously learning about other instrument types in which these are independent of each other.

The hierarchical agent consists of several components:

i) a DP, $P(c \in k_{room_type})$, that clusters contexts according to room types. $P(c \in k_{room_type})$ is governed according to (8) with hyperparameter α_{room_type}.

ii) within each room cluster, a pair of DPs, $P(c \in k_\Phi | c \in k_{room_type})$ and $P(c \in k_\mathcal{R} | c \in k_{room_type})$, for clustering Φ and $\mathcal{R}$ conditioned on the room type. These are also governed by (8) with hyperparameters α_M and α_G respectively. Each reward and mapping cluster determines a corresponding likelihood function, $P(\mathcal{D}|c \in k_\Phi)$ and $P(\mathcal{D}|c \in k_\mathcal{R})$, given by (9) and (10) respectively.

iii) the agent also models environment-wide statistics for Φ and $\mathcal{R}$, from which the room-specific clusters are assumed to be drawn. These are modeled by a pair of hierarchical DPs (hDPs), $P(c \in k_\Phi)$ and $P(c \in k_\mathcal{R})$, where the room-specific DPs act as children within this hDP. Whenever a child CRP generates a new cluster to assign a context to, it draws that cluster from its parent CRP.

Formally, the entire hierarchical model (Fig. 5a) is given by

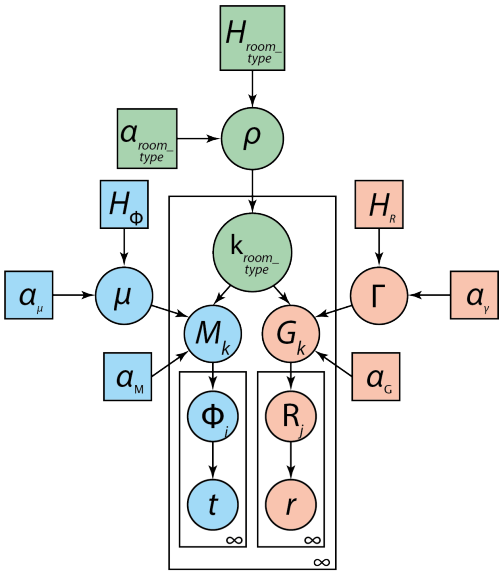
$$
\begin{aligned}
\rho &\sim DP(H_{room_type}, \alpha_{room_type}), & k_{room_type} &\sim \rho, \\
\mu &\sim DP(H_\Phi, \alpha_\mu), & M_k | k_{room_type}, \mu &\sim DP(\mu, \alpha_M), \\
\Gamma &\sim DP(H_\mathcal{R}, \alpha_\gamma), & G_k | k_{room_type}, \Gamma &\sim DP(\Gamma, \alpha_G), \\
\Phi_i | k_{room_type} &\sim M_k, & a_{card} | a, k_{room_type} &\sim \Phi_i = \Phi(a_{card} | a, k_\Phi = i), \\
R_j | k_{room_type} &\sim G_k, & r | k_{room_type} &\sim \mathcal{R}(r | z, k_\mathcal{R} = j),
\end{aligned}
$$

where $\mathcal{R}(r|z, k_\mathcal{R} = j)$ and $\Phi(a_{card}|a, k_\Phi = i)$ are again Bernoulli probability functions.

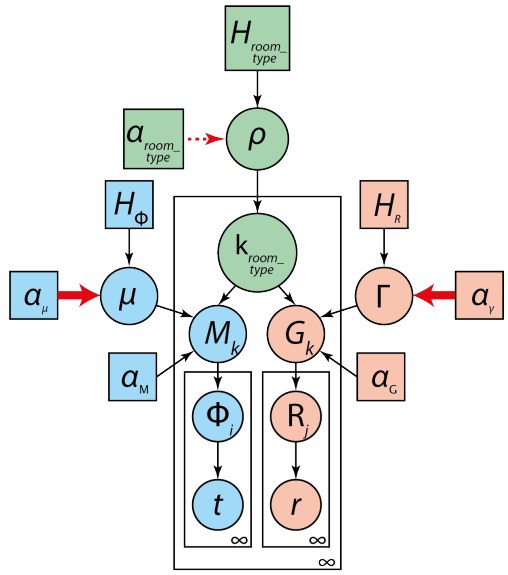
The separate environment-wide hDPs for mapping and reward functions ensures the agent maintains a factorized representation of Φ and $\mathcal{R}$. Without this factorisation, the agent would not be able to learn and transfer task components independently of each other. The hierarchy also allows different clusters across different children DPs to share statistical strength if they originate from the same parent cluster. This captures the intuition that a task component can appear across a variety of structures, and an agent that learns something about that component in one structure should be able to immediately transfer that knowledge to all other structures using the same component.[1]

It is important to note that room clusters offer a generalisation of structures beyond the purely joint structures defined in the joint agents. Indeed, on one extreme, these clusters can comprise specific combinations of task components – i.e., pairings of Φ and $\mathcal{R}$ (Fig. 6, yellow clusters) – as in joint agents. Yet in the intermediate case, the agent might recognise a degree of variability in the choice of task components that the structure uses (Fig. 6, orange clusters). For instance, in room cluster 4 of Fig. 6, the structure is nearly joint. It will always use the dark purple Φ but will sometimes pair this with the dark red $\mathcal{R}$ but other times with the pink $\mathcal{R}$. Thus one can consider this like a noisy joint structure. Finally, on the other extreme, the agent may recognise a room cluster where the components are completely unpredictive of each other. In this cluster, the task components would truly be clustered independently of each other (Fig. 6, brown cluster). Additionally, even within a room cluster, some of the Φ or $\mathcal{R}$ clusters can be more popular than others, as exemplified by the independent room cluster in Fig. 6b.

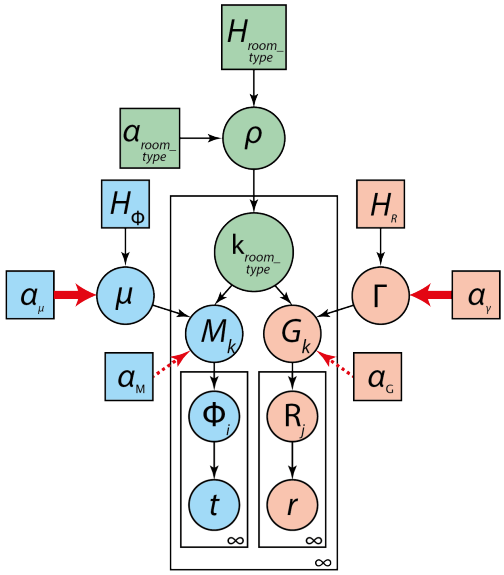
[1] We note that the parent CRPs also express a different type of prior compared to their children. The children CRPs are much like their counterparts in the non-hierarchical agents – in the absence of any data, clusters that were more popular in a given room type will be favoured for other novel contexts associated with that room type. A parent CRP, instead, has a bias towards diversity. It will prefer clusters that appear across a wider range of children CRPs over ones that might be far more frequent overall but special to a smaller number of children CRPs. This captures the intuition that a task component for a novel structure is more likely to be one that is generic to all structures rather than idiosyncratic to a few.



(a) Hierarchical agent consisting of a DP of room types (green) and hDPs of mappings (blue) and rewards (red). Base distributions H_Φ and $H_\mathcal{R}$ generate environment-wide (marginal) clusterings, μ and Γ, of mappings Φ and rewards $\mathcal{R}$, respectively. Within this overall environment there are also clusterings ρ of room clusters k_{room_type} (generated by base distribution H_{room_type}) defining statistical associations between mappings M_k and rewards G_k that apply for this room-type. These generate clusters of mappings Φ and reward functions $\mathcal{R}$ which in turn generate observed transitions t and rewards r in specific rooms.



(b) The hierarchical agent can act like an independent clustering agent by using just one room cluster. This can be encouraged by using a small value for α_{room_type} (dashed red arrow) and large values for α_μ and α_γ (thick red arrows).



(c) The hierarchical agent can also act like a joint clustering agent by using just one reward and mapping cluster per room cluster. This can be encouraged by using small values for α_M and α_G (dashed red arrows) and large values for α_μ and α_γ (thick red arrows).

Figure 5

Moreover, the hierarchical agent confers a generalization advantage over the joint clustering agent even amongst the joint room clusters, because it retains the ability to recognize individual components. In particular, the agent can readily leverage knowledge of familiar task components to construct a novel joint structure that makes use of familiar task components but in a novel combination; only the structure would have to be learnt from scratch. As an example, suppose the agent had already learnt room clusters 1 and 2 in Fig. 6 and was adding in room cluster 3; this cluster reuses the blue Φ from room cluster 1 and the pink $\mathcal{R}$ from room cluster 2, so the agent can readily transfer these components over, even if the combination is novel. In contrast, the joint agent is unable to support such combinatorial transfer and would have to relearn even familiar components again when they reappear in novel ways.

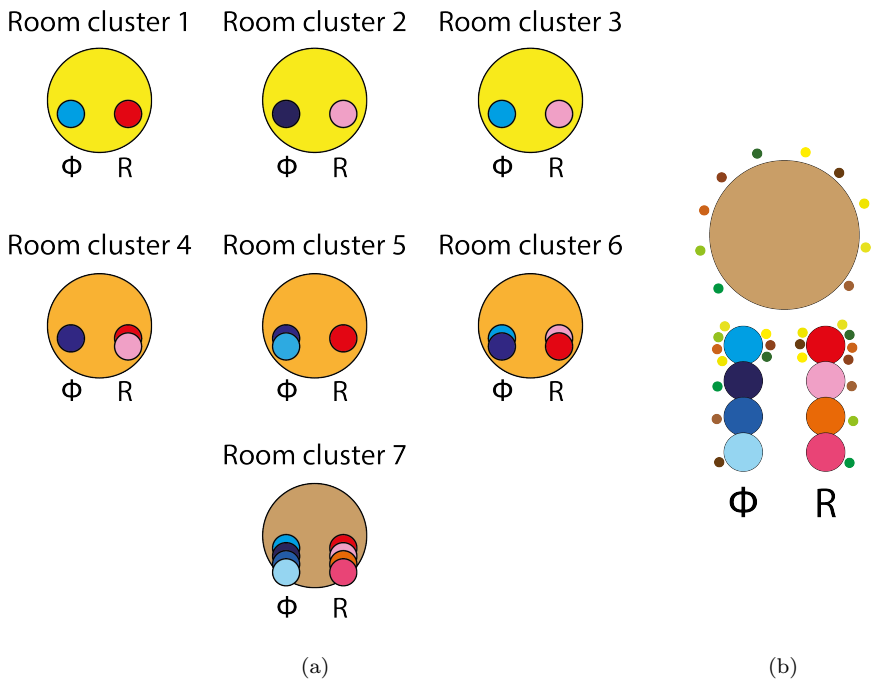


Figure 6: (a) Room clusters (large circles) can capture a variety of structures from purely joint structures (yellow clusters) to joint structures with some variability (orange clusters) to structures with completely independent components (brown cluster). Within the room clusters, small circles represent further clusters of the children CRPs for Φ and $\mathcal{R}$, and chains of small circles represent entire CRPs with multiple clusters. (b) Even within an independent room cluster, one of the mappings or reward functions could be more popular than the others. Here, a variety of distinct contexts (small colored circles) are assigned to this room cluster. Each context is also assigned a Φ cluster (small colored circles surrounding the Φ circles) and a $\mathcal{R}$ cluster (small colored circles surrounding the $\mathcal{R}$ circles). For this cluster though, the top blue Φ cluster has been assigned far more contexts than all the other Φ clusters; likewise for the top $\mathcal{R}$ cluster.

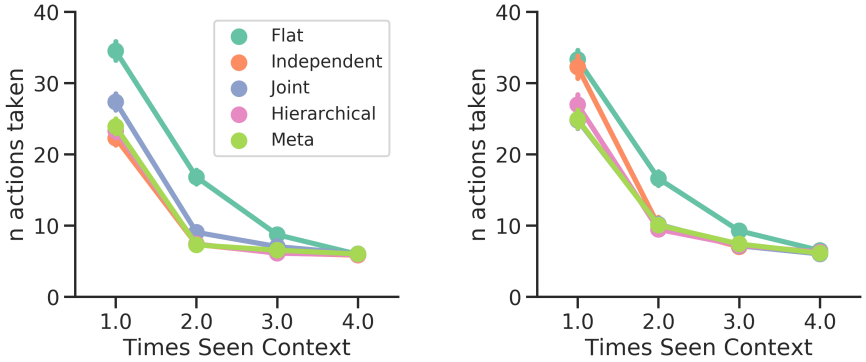
Finally, we also note that the hierarchical agent subsumes the hypothesis spaces of both the independent (Fig. 5b) and joint (Fig. 5c) clustering agents, making the hierarchical agent strictly more general than either of the two. Indeed, by an appropriate choice of hyperparameters, we can actually force the hierarchical agent to behave exactly like either of the two agents, showing that they are special cases. We can force the hierarchical agent to mimic the independent agent by setting α_{room_type} to zero and setting α_M and α_G to their counterparts in the independent agent. Setting α_{room_type} to zero forces the agent to assign all contexts to only one room cluster, in which the reward and mapping functions across all gridworlds would get clustered independently of each other. Conversely, we can force the hierarchical agent to mimic the joint agent by setting α_M and α_G to zero, so that only one mapping and reward cluster gets created inside a room cluster. Each room cluster would then act like a joint cluster, and α_{room_type} would act like the joint clustering agent's α hyperparameter. For both agents, we would also need to set α_μ and α_γ to be infinite so that the children CRPs always get a new cluster from the parent CRPs, rather than reusing a cluster already existing in the parent. The hyperparameters have been summarised in Table 1.

	Independent	Joint
α_{room_type}	0	arbitrary
α_M	arbitrary	0
α_G	arbitrary	0
α_μ	∞	∞
α_γ	∞	∞

Table 1: Hyperparameters for forcing the hierarchical agent into either the independent or joint clustering agents.

3 Non-hierarchical task results

We simulated our agents on the gridworld task structures described above. For comparison, we also simulated an agent that does not transfer any prior knowledge. In each new context, it will learn both Φ and $\mathcal{R}$ from scratch. One can think of this as a joint or independent agent where contexts c map one-to-one onto clusters k; that is, $k = c$. Following [19], we call this the *flat* agent (to contrast it with the other agents that are hierarchical, in the sense that they cluster latent states which then define the mapping and reward functions).



(a) Independent task. The hierarchical agent's performance is equivalent to that of the independent agent and is occluded by it.

(b) Joint task. The joint agent's performance is nearly equivalent to that of the meta-agent and is occluded by it.

Figure 7: Average number of steps taken to reach the goal in a context plotted against the number of times the agent has seen that context. Each point is averaged across all distinct contexts and across 150 simulations. The first point corresponds to the agent's performance when it is seeing a context for the first time, providing a metric for how effectively the agent leverages prior knowledge. A lower number of actions taken implies the agent transfers knowledge more effectively to reach the goal.

As the agent visits more contexts, the complexity of possible clusterings grows exponentially, making exact inference computationally intractable. We have therefore approximated inference by particle filtering (see Supplement for details).

It is important to note that, ultimately, the only difference between agents lies in how they cluster contexts. It is the structure of this hypothesis space that determines the Bayesian priors used to support transfer, and it is this alone that is responsible for any performance differences between the agents. At the level of individual reward and mapping clusters, all agents operate in exactly the same way.

3.1 Task with independent statistics

We first simulated our agents in the task with independent statistics (Fig. 2a). As described above, each gridworld is associated with a context, and each context repeats four times across the entire task. Fig. 7a shows how many steps each agent needed, on average, to reach the goal in a context as a function of how many times the agent has seen that context.

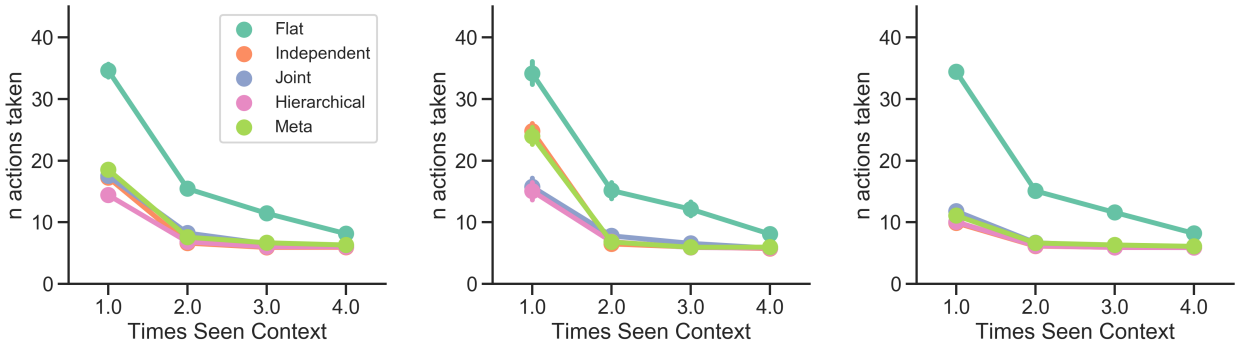
As expected, the independent clustering agent shows the most effective transfer, needing the fewest actions on average to reach the goal in a novel context. It can leverage the knowledge from previous contexts about the likely Φ and $\mathcal{R}$ functions (in contrast to the flat agent) without assuming that knowledge of one is indicative of the other (in contrast to the joint agent). As shown in [19], the meta-agent can also transfer effectively, as its hypothesis space subsumes that of the independent agent, so the near-optimal performance implies that these agents are correctly finding hypotheses that model Φ and $\mathcal{R}$ as independent components. The same property emerges from the hierarchical agent here, confirming that it can also easily identify independent structure even though (unlike the meta-agent) it is not given this structure as one of two *a priori* hypotheses.

3.2 Task with joint statistics

We next examined the task with joint statistics between Φ and $\mathcal{R}$ (Fig. 2b). Here, the joint clustering agent transfers most effectively (Fig. 7b), while the independent agent is almost as suboptimal as the flat agent. The hypothesis spaces of both the meta-agent and the hierarchical agent subsume that of the joint agent as

well, so their near-optimal performance suggests that they are successfully converging on hypotheses that capture the joint statistics between Φ and $\mathcal{R}$.

3.3 Task with mixed statistics



(a) All training contexts (b) Joint training contexts (c) Independent training context

Figure 8: Average number of steps taken by each agent in the training contexts plotted against the number of visits to that context.

The first two tasks had either perfectly independent or perfectly joint statistics, meaning that the meta-agent, which arbitrates between a joint clustering or independent clustering agent, can perform near optimally on both tasks [19]. However, the optimality of the meta-agent breaks down when the task statistics are mixed (Fig. 3). Here, some contexts are "joint" in that a given goal is always paired with a given mapping. Other contexts are more independent, where a given mapping might be paired with either of two goals.

The agents first passed through a series of training contexts consisting of 4 joint contexts and 30 independent ones, in random order. Fig. 8a shows that the hierarchical agent transferred knowledge more effectively than the other agents in novel contexts, consistent with its more expressive hypothesis space. In contrast, the meta-agent performs poorly: while it can mix its policy across its independent and joint sub-agents, it cannot recognize specific sub-structures in the task (e.g., that some clusters should be treated as strictly joint). In this particular setting, the independent agent performed comparatively better, simply because it could leverage the fact that there was one goal that was most popular across all contexts, so if it went to this goal first in a new context, it would arrive at the correct goal more often than not.

Notably, despite its overall adequate performance, the independent agent is unable to discover joint sub-structures that are present in the joint contexts (Fig. 8b). Here, both the hierarchical and joint agents performed the best by a significant margin. Demonstrating its enhanced expressivity, the hierarchical agent learned to assign all joint contexts to one room cluster with joint structure, and all the independent contexts to another room cluster with more independent structure (Fig. 9). In contrast, the independent agent and the meta-agent transferred less effectively in the joint contexts. The independent agent has a prior that favours the most popular goal. In this case, the most popular goal overall had never been associated with a joint context. The independent agent cannot recognize this structure and hence it will generally attempt the wrong goal first in a joint context. The meta-agent fails for a similar reason: the prevalence of the independent training contexts induces a bias to prefer the independent agent, which again visits the wrong goal.

3.3.1 Test contexts

To more directly assess differences in generalization, after training, all agents were evaluated on three test contexts. The first context simply involved another repeat of the mapping-reward pairing seen in the joint training contexts. This context revealed the stronger ability of the joint and hierarchical agents to transfer learnt knowledge of joint statistics (Fig. 10a). Indeed, their performance did not improve much when they saw the same context a second time, suggesting that they were already near optimal on their first try. In

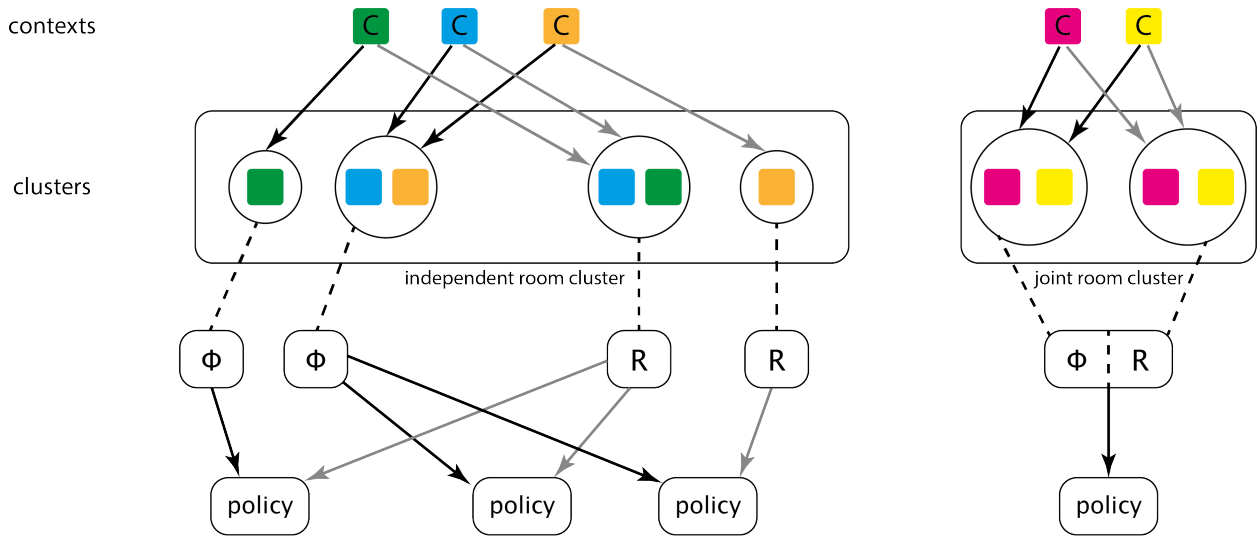


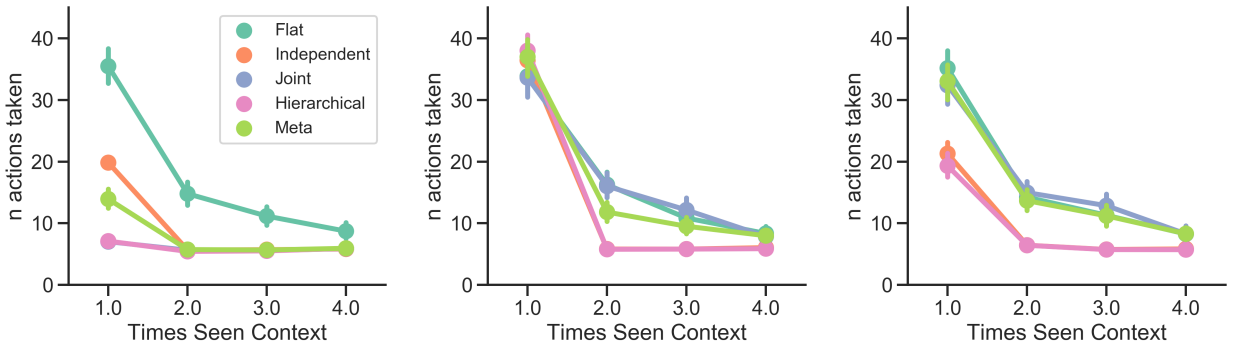
Figure 9: Hierarchical agent's clusterings after training in the task with mixed statistics. The agent discovers two room structures: an independent room type in which Φ and $\mathcal{R}$ are clustered independently, and a joint room type in which each context that forms part of the cluster is linked to a single Φ and $\mathcal{R}$ pair. There are additional parental clusters that the room clusters inherit from but which we have omitted in the figure for simplicity. A more complete figure with the parental clusters has been included in the Supplement.

contrast, the independent agent displayed less optimal transfer, as its prior would again favour the incorrect goal initially. It could still outperform the flat agent, however: after reaching the wrong goal, it would quickly recover by trying the correct goal next, as this is the only other goal the prior knows about. The meta-agent's performance lies somewhere between the independent agent's and the joint agent's. Apart from the flat agent, all agents were optimal in their second visit to the context, indicating that by the end of their first visit, they had inferred the correct cluster assignment and could therefore readily transfer the correct knowledge from then on.

The second context retained one of the previous (but less popular) goals, while introducing a novel mapping. As such, none of the agents could transfer knowledge effectively on the first visit to this context (Fig. 10b); all agents needed to learn the novel mapping from scratch. Nevertheless, the independent and hierarchical agents can transfer goal information independently of mappings, and hence have potential to show transfer. But because the independent agent favours the most popular goal, it actually exhibits negative transfer. In contrast, the hierarchical agent can assign the novel context to one of two possible room clusters. It could favour the independent room cluster, in which case it would also show negative initial transfer. Or it could favour the joint room cluster, adding an additional mapping to it. Because it can't know which cluster it is in before it observes the goal, it does not show positive transfer on the first visit. But the independent and hierarchical agents were able to solve the context optimally from the second visit onward, indicating that they had learned the novel mapping and determined the correct cluster assignment by the end of the first visit and were henceforth transferring the knowledge of the correct goal. In contrast, the joint and flat agents were still trying to relearn even familiar goals anew. The meta-agent's performance again lay somewhere between that of the joint and independent agents.

Finally, the third test context introduced another novel mapping, this time paired with the most popular goal (Fig. 10c). This context provided an even clearer demonstration of both the independent and hierarchical agents' ability to generalise task components independently, as shown by their better performance on the very first visit to this context. Even though the agents had no exposure to the novel mapping, they could still transfer the reward function. In this case, the independent agent's prior favours the correct (most popular) goal. The hierarchical agent can, again, assign this context to either the joint or independent room cluster, but as the independent room cluster was more popular across the environment, it prefers this cluster.[2] In

[2]While in this task, the hierarchical agent preferred the more popular goal in the test independent contexts, we note that the preference is not due solely to goal popularity but actually involves competing preferences between the room-type CRP and



(a) Joint test context. The joint agent's performance is equivalent to that of the hierarchical agent and is occluded by it.

(b) Independent test context with the less popular goal. The independent agent's performance is equivalent to that of the hierarchical agent and is occluded by it.

(c) Independent test context with the more popular goal.

Figure 10: Average number of steps taken by each agent in the test contexts plotted against the number of visits to that context.

contrast, the joint agent again shows no transfer and mimics the flat agent's performance, while the meta-agent seems to have settled on using a joint agent to solve this context (likely because the independent clusters are not perfectly independent and overall a joint strategy performs reasonably well).

We note that while some contexts favoured the joint agent and others the independent agent, only the hierarchical agent consistently showed the better performance across all contexts, demonstrating the greater flexibility of its hierarchical hypothesis space of context clusterings.

Finally, in the Supplement, we provide additional violin plots showing the distribution in the total steps needed by each agent to solve each of the tasks we have described above.

4 Scaling compositionality to hierarchical tasks

Above we showed that the hierarchical agent can discover and transfer substructures across a range of task statistics, and is thus more expressive and general than any of the other agents. But in the simulation examples, to reach the goal in any given context, agents had to discover only just a single goal and transition function. In contrast, naturalistic tasks often have a hierarchical goal/subgoal structure to them. In this section we leverage this hierarchical power and apply it to task structures that themselves have hierarchical structure.

Returning to the music festival example, suppose a jazz musician were playing at a new festival for the first time. The overall goal of putting on a successful concert can be broken down into the subgoals of successfully improvising through each song. The musician might first choose her set list by selecting songs likely to be crowd pleasers based on the genre of music and the other musicians at the festival. For any given song, she might then try taking chord progressions ("sub-subgoals") from piano songs and improvising on them with the saxophone. Finally, the specific sequence of finger movements to play in a given chord progression may itself be inherited from her knowledge playing the flute. In each case (song choice, chord choice, movements), effective transfer requires the musician to borrow knowledge from a different set of learned structures. As another example, suppose one were travelling to a foreign land to conduct some business. At the start of a meeting in a conference room, the businessperson will realise that, in this context, they should first be introducing themselves and thanking their hosts. This decision about the general way to behave is inherited from their experience across many other business meetings in other cities. But the specific social norms that apply as to *how* to introduce, thank and greet the hosts should be informed from

the room-specific children CRPs, with the former favoring independent room structures while the latter preferring joint room structures. We have left a fuller exploration of this interaction to the Supplement, where we show theoretically and empirically that it is possible for the hierarchical agent to favor the less popular goal instead. Apart from this, the overall behavior of the various agents remain unchanged.

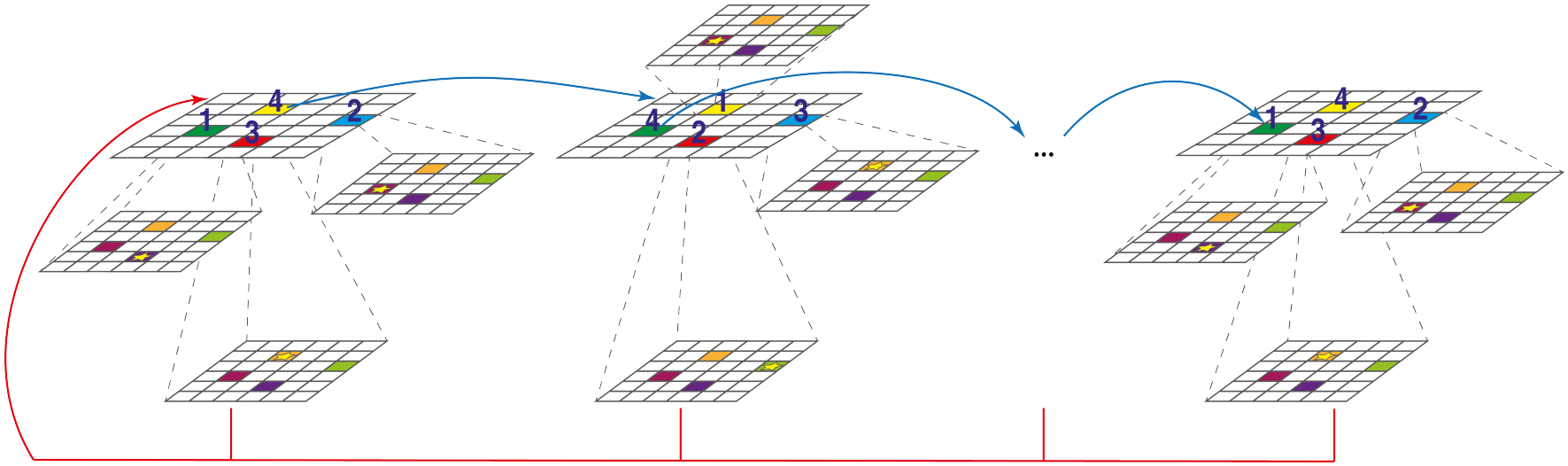


Figure 11: Diabolical hierarchical gridworlds. The agent has to navigate through a series of rooms, each consisting of an upper level and three sublevels. To advance to the next room, the agent must first complete a sequence of sub-levels in the correct order (numbered squares symbolizing doors). The first three doors lead to the room's three sublevels, while the fourth door allows the agent to advance to the next room (or ends the task if it is the final room), but only if the agent has first visited and correctly completed all three sublevels in the correct order. Each sublevel has four possible exits, but only one (orange star) will be correct and allow the agent to return to the upper level. Mappings, door sequences, and correct sub-level exits can differ across rooms and sub-levels, but some can be reused to afford transfer. The task is diabolical in the sense that if the agent visits any door out of order or goes to the wrong sublevel exit, it is sent back to the beginning of the task (red arrows), making mistakes extremely costly.

the local culture, and they would need to adjust their greeting "policy" accordingly, whether it be to shake hands or bow or, in the context of a global pandemic, bump elbows.

In each of these cases, solving the overall task would require implementing a policy that also has hierarchical structure, where the overall policy can usually be broken up into subpolicies for the different subtasks, and the different subtasks can be recombined in novel ways to quickly solve novel macro tasks. Developing approaches to learn such hierarchical policies has been the focus of *hierarchical reinforcement learning* [28, 29]. However, such approaches have not considered how to transfer distinct components of knowledge across levels of the hierarchy depending on both global and local contexts. In the next series of tasks, we explored the possibility of scaling our approach up to tasks with such hierarchical, compositional structure.

4.0.1 Hierarchical sequences of gridworld rooms

We considered two types of hierarchical tasks. The first is a modification of the "diabolical" rooms task from [19] to include hierarchical structure. In the original task, the agent had to navigate through a sequence of gridworld rooms, where the transition function (mappings) and reward function (goals; i.e., which door to enter to reach the next room) in any given room might be similar to one of the previous rooms, in a compositional manner. The diabolical nature of the task refers to the problem that if in any room the agent navigates to the wrong door, it will return back to the start of the first room, making mistakes extremely costly. There is thus a large advantage towards agents that can transfer prior knowledge compositionally; this advantage grows with the number of rooms and the size of each one [19].

The hierarchical version of the task introduced here further amplifies the complexity and the diabolical aspect to match the musical festival analogy above. The agent must again navigate through a sequence of gridworld environments (Fig. 11). However, the gridworlds are organised into a set of hierarchically structured rooms. Each room consists of an upper level and three sublevels, where solving each sublevel will return the agent back to the upper level (reminiscent of "underground" levels in the "Super Mario Bros" video game or "dungeons" in "Zelda"). The upper level has four doors, and the agent must visit each of them in the correct order. The first three doors lead to each of the sublevels. The fourth door leads to the next room (or ends the task if it is the final room), but only if the agent has first completed the first three sublevels in that room.

Each sublevel itself comprises another room with four exits, but where only one is correct and returns to the upper level. Again, any mistake (within upper or sub levels) will send the agent back to the beginning of the entire task, hence again putting pressure on the agent to leverage prior knowledge about rooms and sublevels effectively. To give the agent some chance of reaching the goals in finite time, it is given information about the locations of the doors and exits (but of course not which ones are correct). This means the reward function will now model the reward probability at the four possible goal locations only. Further, to avoid having a non-Markovian reward function, the agent's state information in the upper level includes how far along has it progressed in the door sequence.

In second version of the task, to better examine transfer in individual rooms, we remove the diabolical part of the task. Each room is now associated with a context, each context repeats four times across the task, and contexts appear in random order. The agent still knows the locations of the four possible doors or exits, but still has to learn which ones to go to in which order, and how to select actions to move in the cardinal directions (i.e., the mappings).

We considered both diabolical and non-diabolical versions of the tasks, varying the task statistics from independent to joint in each case. All diabolical tasks used a total of 16 upper level rooms, and all non-diabolical tasks used 16 contexts, meaning that the agent visited 64 rooms in total (including the three sublevels for each room).

For both diabolical and non-diabolical versions, we manipulated the generative statistics of the environment to generate three task types.

1. Independent statistics. This task used a total of four distinct mappings, four distinct door sequences, and all four possible sublevel exits. To generate independent statistics, the applicable mappings, door sequences in the upper rooms, and exits in the sublevels, were all randomly paired with each other; that is, knowledge of the transition function or reward function in any level was not informative of the others in that level or the other levels. Nevertheless, to ensure that there was a benefit to leveraging prior knowledge, some mappings, door sequences, and sublevel exits were more prevalent than others.

Thus an agent could improve performance by, for example, reusing popular mappings in a room or sublevel that it had encountered in combination with a different door sequence.

2. Joint statistics. In this case, the upper level mapping was entirely predictive of both the door sequence and the mappings and exits in each of the room's sublevels. Again, some of the mappings, and by extension the doors and sublevels they predict, were more prevalent than others. Transfer could be improved by recognizing and reusing this joint association.

3. Mixed statistics. There were three rooms (or contexts) where the statistics were strictly joint, four where the upper level mappings were predictive of doors and sublevel exits but not sublevel mappings, and nine where all task components were independent of each other.

4.0.2 Agent architectures for the hierarchical tasks

We ran a flat agent (as in the earlier simulations, the flat agent had no clustering or generalization abilities, but was still a model-based agent), an independent clustering agent, and a hierarchical agent on the hierarchical tasks.[3] As above, the agents are endowed with knowledge of the general structure of the tasks: in this case, that there are rooms and sublevels, goals and mappings to be learned for each. Thus we extended the agents to allow for clustering of these structures, thereby generalizing them. Indeed, our hierarchical tasks can be reduced to the non-hierarchical ones if the first door in the sequence for each upper level led directly to the next room; our extended agents for the hierarchical tasks could then be directly run on the non-hierarchical tasks.

The independent agent clusters each task component independently of the others (Fig. 12). One DP is used to cluster mappings across the entire task hierarchy, and another is used to cluster exits for all sublevels. The agent also maintains distinct DPs to cluster reward functions for each position in a door sequence.[4]

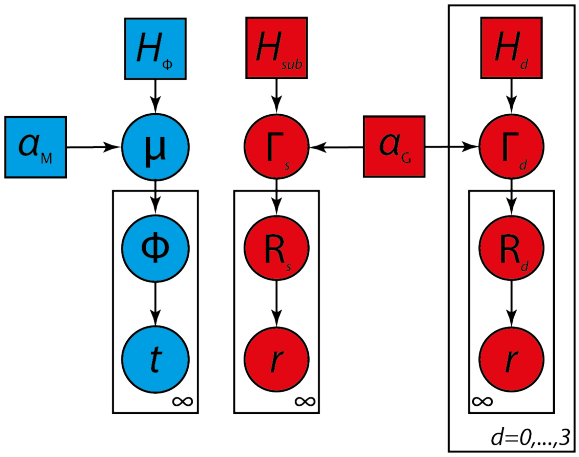


Figure 12: Independent clustering agent for the hierarchical environment. The blue DP clusters mappings, with the base distribution H_Φ generating clusterings μ of mappings Φ, which in turn generates the observed transitions t. The red DPs independently cluster goals for doors and sublevel exits. A base distribution H_{sub} generates clusterings Γ_s of exit reward functions R_s, which in turn generates observed rewards r. A set of four independent DPs, one for each position in a door sequence, generate clusterings Γ_d of door reward functions R_d from base distributions H_d. R_d in turn generates observed rewards r.

The hierarchical agent, in contrast, can simultaneously represent independent and joint structure, as it clusters task components at multiple levels of hierarchy (Fig. 13). Just as in the non-hierarchical task, it maintains one clustering ρ of room clusters k_{room_type} that capture higher-order structural information. Sublevel structures, s, are identical to the room structures of the previous hierarchical agent.

[3]Unlike previously, we did not include a fully joint agent as jointness can be defined on multiple levels, such as across the entire room, at the level of individual upper and sublevels, at the level of sublevel sequences within a room, and at the level of exit sequences amongst a room's sublevels, to name just a few possibilities. The task structure involves many subcomponents which affords many possibilities of jointness, and selecting one or a subset of agents that assume different types of jointness would involve a certain degree of arbitrariness. For this reason, we have chosen not to include a joint agent baseline for these tasks.

[4]On the one hand, maintaining independent clusterings between the different door positions would allow the agent to capture the differential prevalences of different doors in different sequence positions. In this way, the agent does still capture some higher order structural information. But on the other hand, this separate clustering prevents information about the same door from being pooled together whenever it appears in a different sequence position. One could also pool generic goal information together from upper level doors and sublevel exits, because doors and exits are located in identical positions within their respective gridworlds and should therefore have similar reward functions. But this separate clustering of doors and exits also prevents that. To pool information from different parts of the task hierarchy together as well as model higher order structural information, one would have to cluster task components in a more hierarchical manner, something not possible with the independent agent.

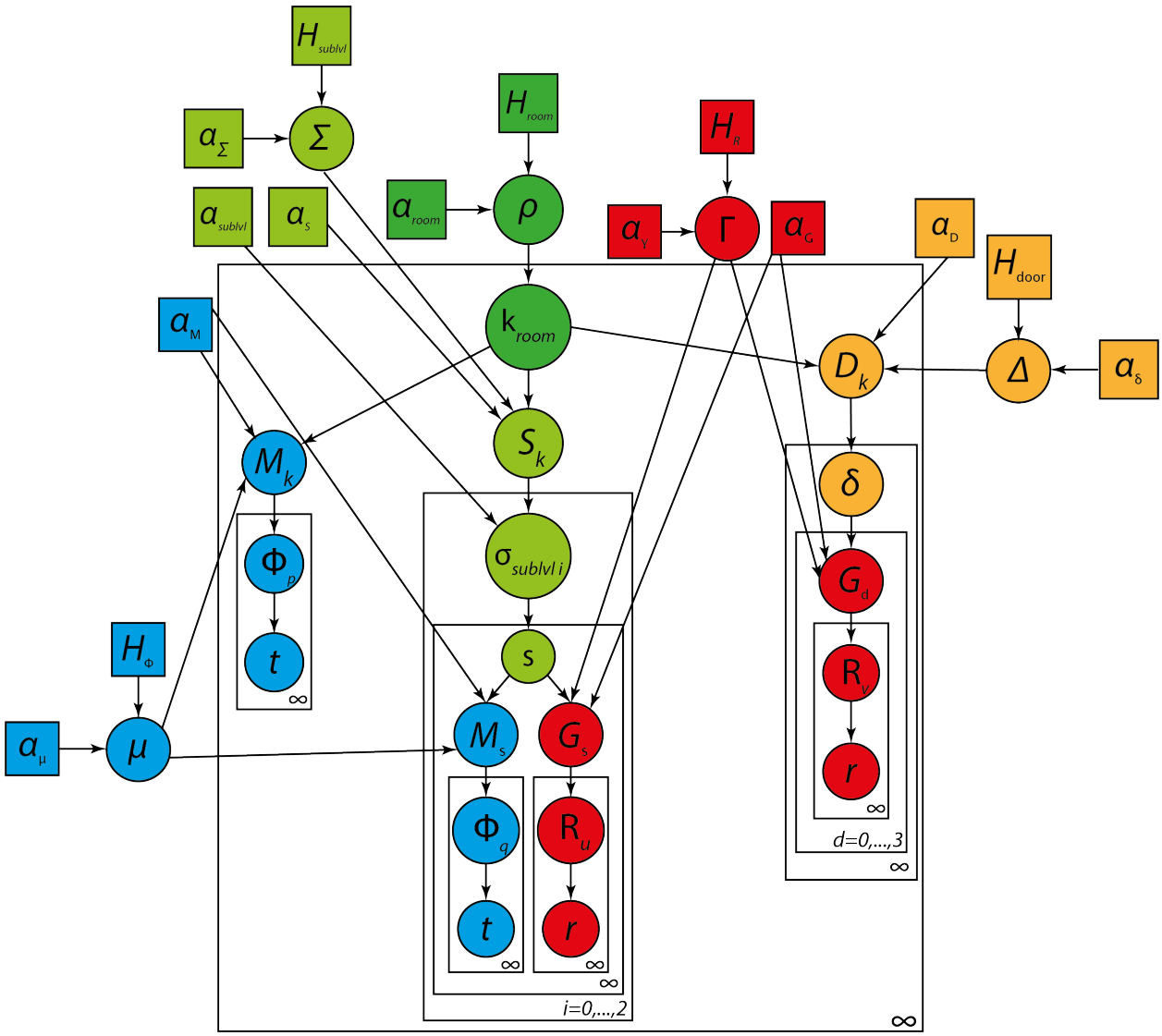
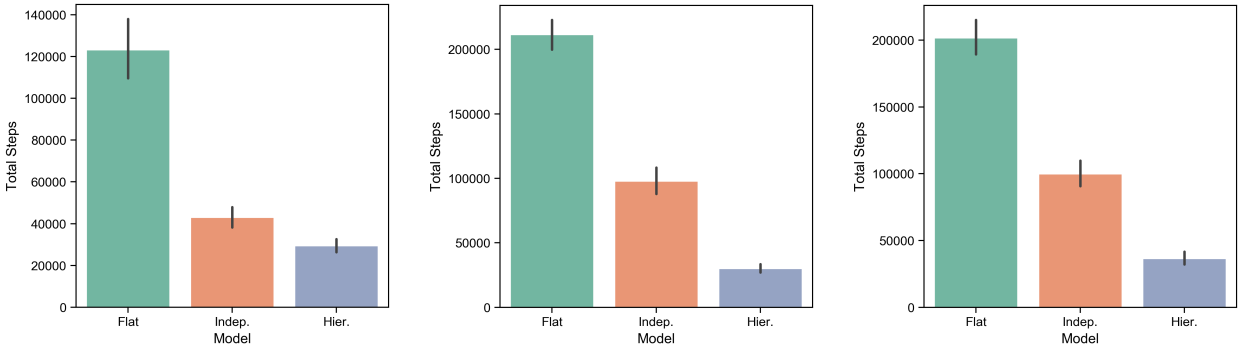


Figure 13: Hierarchical agent's generative model for the hierarchical environment. This consists of several interacting DPs and hDPs, including a DP for room clusters (dark green), an hDP for sublevel clusters (light green), an hDP for door sequence clusters (orange), an hDP for mappings (blue), and an hDP for goals (red). Base distributions H_{room_type}, H_{sublvl}, H_{door}, H_R, and H_Φ generate task-wide clusterings, respectively, of rooms ρ, sublevels Σ, door sequences Δ, goals Γ, and mappings μ. A room cluster k_{room_type} is drawn from ρ and contains room-specific clusterings of sublevels S_k, mappings for the upper level M_k, and door sequences D_k. Distribution D_k generates door sequence clusters δ, which in turn consists of four further clusterings $\{G_d\}$, one for each door position, with each G_d clustering goals representing doors R_v. Distribution S_k generates a further three clusterings $\{\sigma_{sublvl\,i}\}$ of sublevel structures s specific to each position in the door sequence. Each sublevel structure s has further clusterings of mappings M_s and goals G_s specific to the sublevel. Finally, mapping clusters Φ and goal clusters R, wherever they appear in the hierarchy, generate observed transitions t and rewards r.



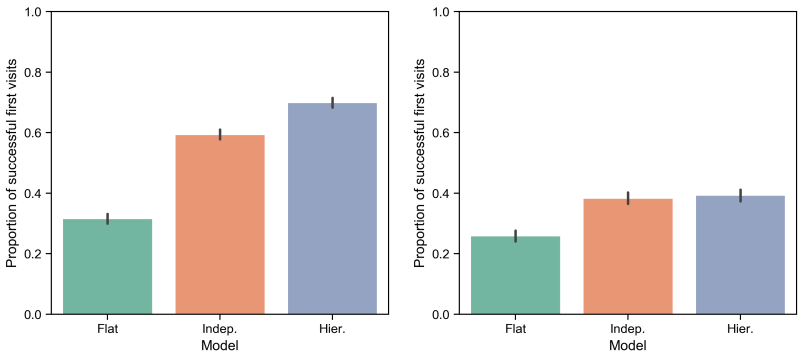
(a) Independent task statistics (b) Joint task statistics (c) Mixed task statistics

Figure 14: Hierarchical diabolical rooms tasks. Total number of steps each agent needed to solve each task. Results are averaged across 50 runs.

Each structure consists of one DP for clustering sublevel reward functions and a separate DP for clustering sublevel mappings. These DPs draw their clusters directly from the parent goal and mapping DPs, μ and Γ. At the parental level, the agent also maintains task-wide clusterings of door sequence structures (Δ), sublevel structures (Σ), mappings (μ), and goals (Γ). A door sequence structure δ is another composite structure that consists of four DPs, $\{G_d\}$, for clustering reward functions for the positions in a door sequence. These DPs draw their goal clusters directly from the parent goal DP, Γ, which allows goal information to be shared across doors and sublevel exits.

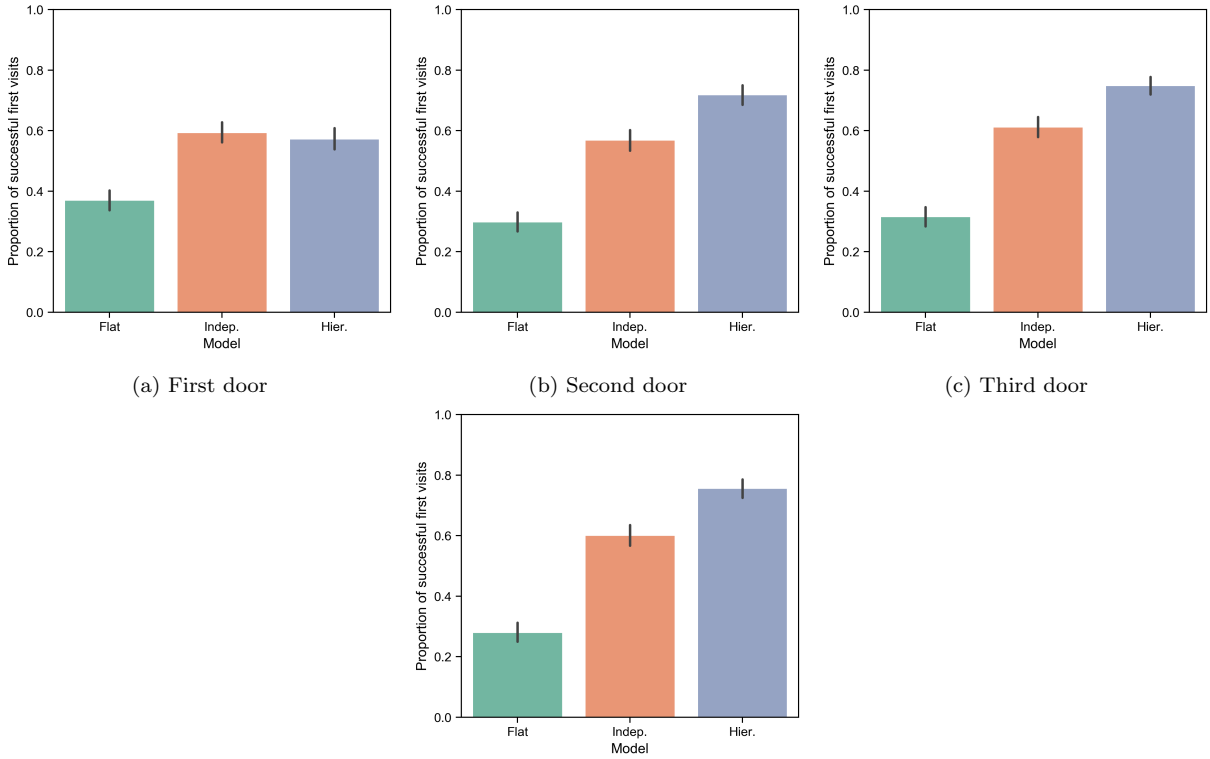
Each room cluster k_{room_type} maintains a set of children DPs that also draw task components from their parental counterparts. One of these clusters door sequences D_k, another clusters upper level mappings M_k, and a third clusters sublevel structures S_k. This sublevel DP serves as a parent for a further three children DPs $\{\sigma_{sublvl\,i}\}$ that cluster sublevel structures for each of the three sublevels in the room. These children DPs model the conditional probability of seeing certain mappings or reward functions given some latent room type or sublevel type.

Another commonly-used approach to hierarchical reinforcement learning makes use of a type of subpolicy known as options [28, 29]. The options framework, however,is hierarchical only at the level of the policy. Unlike our approach, it does not cluster subpolicies or task components such as latent Φ or $\mathcal{R}$, and it is therefore incapable of supporting any form of transfer (but see, e.g. [30] for one example of option reuse). Indeed, to use an option, one must first define an initiation set of states where the option can be entered into, whereas in our approach, any familiar subpolicies can be used in novel, unfamiliar contexts with the appropriate choice determined through Bayesian inference. Additionally, the set of possible options must be specified at the start of the task and is immutable thereafter, whereas our approach allows the agent to acquire novel subpolicies in an online, continual manner.



(a) Upper level doors (b) Sublevel goals

Figure 15: Probability of successful first visits



(a) First door

(b) Second door

(c) Third door

(d) Fourth door

Figure 16: Probability of successful first visits to upper level doors according to sequence position

4.0.3 Diabolical tasks

In all three diabolical tasks, the independent and hierarchical agents greatly outperformed the flat agent baseline, needing far fewer steps on average to solve these tasks (Fig. 14), indicating that they successfully leveraged experience from previous rooms to solve new ones more quickly. Moreover, the hierarchical agent significantly outperformed the independent agent, as it is able to exploit higher order statistical structures in the rooms while retaining the independent components. Surprisingly, the hierarchical agent even somewhat outperformed the independent agent in the task with independent statistics. As we will see below, this improvement was simply due to imperfect independence of the generative statistics: unlike in the non-hierarchical task, the combinatorial explosion in complexity arising from the multiple task components made it difficult to ensure complete independence of task statistics within only a limited number of rooms. The better performance of the hierarchical agent shows that it is sensitive enough to pick up on any residual structure that may lurk within the task statistics.

If the hierarchical agent performs better than the independent agent, it must be able to predict the correct goal whenever it sees a gridworld room for the first time. Fig. 15 shows the fraction of times each of the agents predicted the correct goal in the first opportunity to do so (i.e, for a novel position in a door sequence or the correct exit for a novel sublevel). We see that, as expected, the flat agent's performance is near chance level of 0.25, but the independent and hierarchical agents' performances are much higher. Interestingly, the difference between the independent and hierarchical agents emerges only in anticipating the correct doors from the second door onwards (Fig. 16). This result suggests that the hierarchical agent is exploiting residual structure in the door sequences.

To better understand this, we quantify the information theoretically available to an agent about structure between task components. Let us suppose the agent has already seen the following sequences of mappings and doors in the upper levels,

A 1 3 4 2	B 3 1 2 3
A 2 2 4 3	B 1 2 1 4
A 3 2 1 3	C 4 1 3 1
A 4 3 4 1	
A 4 1 4 1	

where letters denote different mappings and numbers denote doors. When an agent enters a new room, it will learn the mapping first before it gets any goal information. If it learns that the mapping is A, it will not be able to tell whether the first door will be 1, 2, 3, or 4, as all possibilities follow mapping A. But if it learns the first door is 1, 2, or 3, it can immediately infer the rest of the sequence, since knowing any of these is sufficient to narrow down the possibilities to just one unique sequence. But if it sees door 4, it will need to learn the identity of the next door before it can narrow down the possibilities to a unique sequence. Alternatively, if the agent learned the mapping was B, it would only need to know the identity of the first door before it could figure out the entire sequence. And if it learned the mapping was C, it could immediately infer the entire door sequence right away, although such a coincidence of a mapping paired with just one door sequence was highly unlikely in our tasks. The amount of information that a partial sequence s tells an agent about the complete sequence can be quantified using the Shannon information

$$I(s) = -\ln p(s),$$

where the probability is computed with respect to the dictionary of all sequences. Since dictionaries can be of varying sizes across different simulations, we normalise this with respect to the total information obtained when the entire sequence is learned; that is,

$$i(s) = -\frac{\ln p(s)}{\ln \mathcal{N}},$$

where $\mathcal{N}$ is the total number of unique sequences.

To see whether the hierarchical agent was exploiting residual regularities in the door sequences left in the independent task, we performed several tests on the fractional information content of the partial sequences. We divided the partial sequences into two sets, one where the hierarchical agent succeeded at guessing the correct door but the independent agent failed, and one where both agents succeeded or the hierarchical agent failed. We found that when the hierarchical agent was more successful, the fractional information content of the partial sequences was significantly higher ($p = 1.5 \times 10^{-13}$). We also ran a logistic regression to predict the probability that the hierarchical agent will be more successful at guessing the door from the fractional information available (Fig. 17). A likelihood ratio test showed that this model was significantly better than one where the probability was the same across all levels of fractional information ($p = 1.35 \times 10^{-11}$). These results indicate that the agent is likely exploiting structure in the door sequences, which it may do by having more joint-like representations in its door sequence clusters.

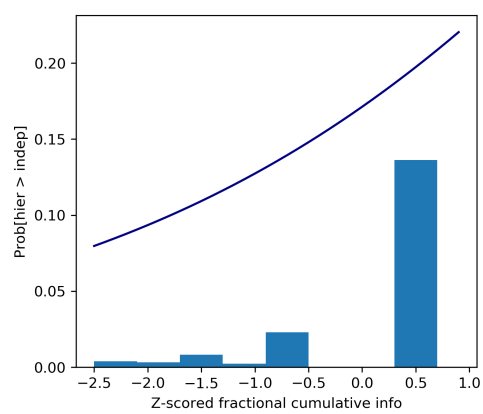


Figure 17: Probability of the hierarchical agent being more successful at guessing the correct door plotted against the (z-scored) fractional information content available about door sequences. Bars show the fraction of times the hierarchical agent was more successful for different bins of fractional information; the navy line shows the logistic fit. Data was aggregated across all rooms and all 50 simulations.

The Supplement provides additional histograms showing the distribution in the total number of steps each agent needed to solve each of the tasks described above.

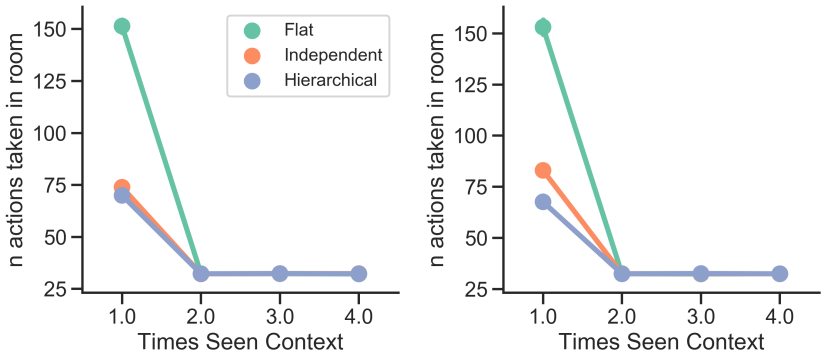
4.0.4 Non-diabolical tasks

As another means of diagnosing agent transferability, we subjected them to non-diabolical versions of the hierarchical task. Across all types of task statistics, the hierarchical and independent agents demonstrated significant positive

transfer in novel contexts compared to the flat agent baseline, with the hierarchical agent being the better of the two (Figs. 18a-b, 19a). In the task with independent statistics, the difference between the hierarchical and independent agent was less pronounced.[5] In the task with mixed statistics, we have also broken down the results into the different types of contexts and levels (Fig. 19). We find that the performance advantage of the hierarchical agent over the independent agent is present in all types of gridworlds, whether it be an upper level or sublevel or whether the context belongs to one of joint, mixed, or independent statistics. Finally, the Supplement provides additional violin plots showing the distribution in the number of steps each agent needed to solve these tasks.

5 Discussion

Taken together, our simulation results demonstrate the adaptive advantage of hierarchical clustering for generalization within a model-based reinforcement learning setting. Previous work has demonstrated the need to independently store and cluster task components of a Markov decision process to support flexible compositional generalisation [19]. Yet there are cases when such a strategy can actually lead to suboptimal performance, especially when there are higher order relationships binding task components together; and in such cases, an agent that generalises jointly can actually perform better. One previous approach was to use a meta-agent that arbitrates between joint and independent clustering strategies [19]; but such an agent is unable to recognise joint and independent sub-clusters existing alongside each other in the same task. Thus it has remained an open question of how an agent can balance the need to compositionally generalise while also recognising that there are situations when transferring joint structures will actually serve the agent better. One of the principal ideas put forth by this work is that an agent must be able to do both simultaneously: it must be able to learn task components independently, but it must also be able to learn the higher order structures that combine them. Our hierarchical agent is intended to satisfy these competing demands in a principled manner. Its hierarchical structure maintains a clear delineation between the independent learning of task components, as implemented by parental DPs, and the learning of structures that combine these components together in a variety of novel ways, as implemented by room clusters. This set-up naturally allows different structures to share statistical strength by pooling knowledge about shared task components, supporting compositional generalisation. Additionally, by using structure-conditioned distributions over components, our framework also allows for a more probabilistic version of structures where component identity can be stochastic. As validated by our simulations, our hierarchical agent framework leads to better and more flexible transfer across a wider range of task statistics, including those that have hierarchical goal-subgoal structure.

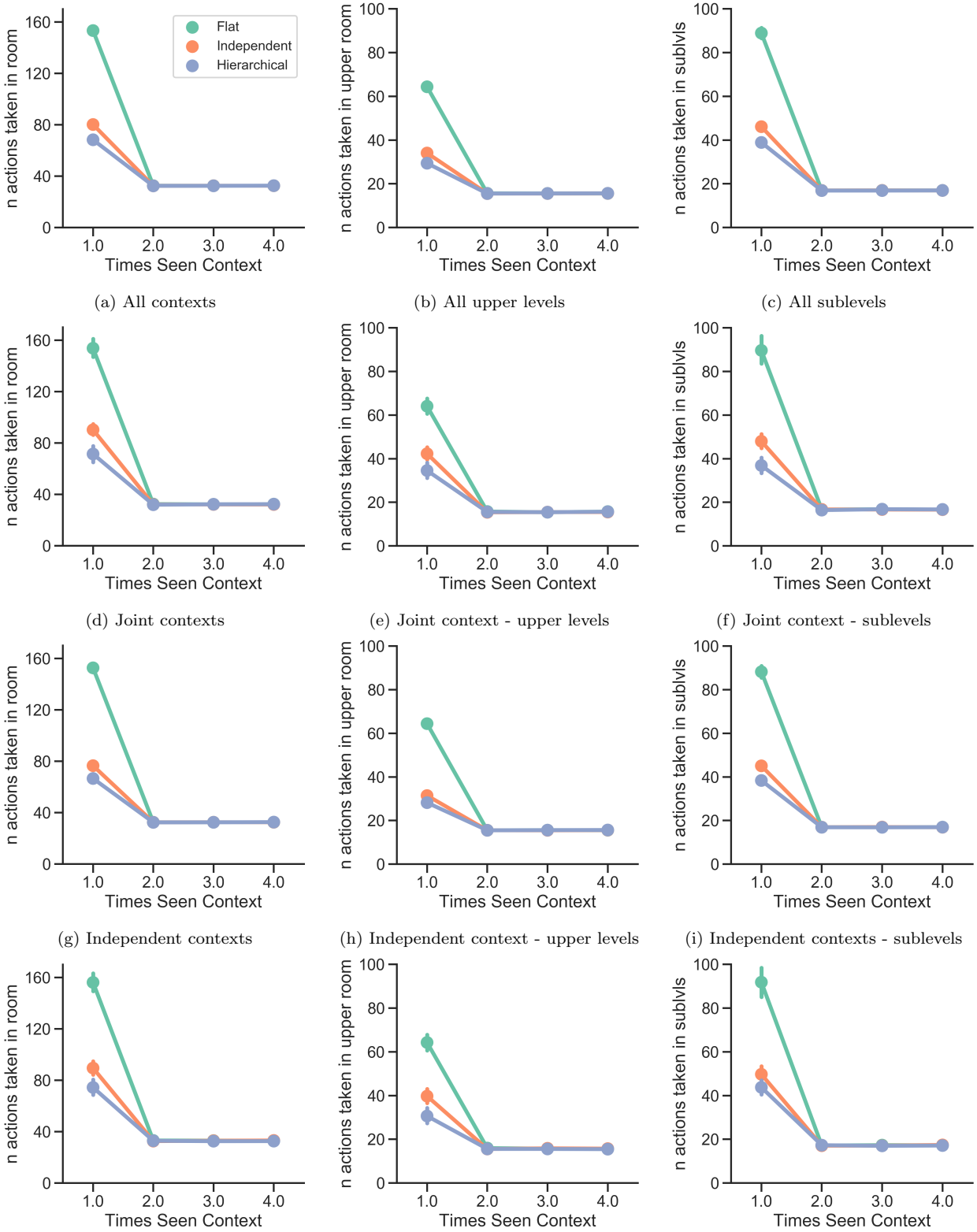


(a) Independent hierarchical task (b) Joint hierarchical task

Figure 18: Hierarchical task results

Although we have focused on compositions of rewards and transitions, our framework is quite general and can be applied to any type of compositional, hierarchical structure. For example, the transition function itself might have further substructure where transition functions covering one region of state space might be composed with another transition function covering a different region (retaining our music example, one could apply the finger picking movements of a guitar but using chord finger placements on a banjo).

[5] This may be because the diabolical nature of the previous tasks amplified any performance advantage one agent has over the other, because over repeated resets, any small performance difference in individual gridworlds will eventually accumulate into a large final difference by the task's end.



(a) All contexts (b) All upper levels (c) All sublevels

(d) Joint contexts (e) Joint context - upper levels (f) Joint context - sublevels

(g) Independent contexts (h) Independent context - upper levels (i) Independent contexts - sublevels

(j) Mixed contexts (k) Mixed context - upper levels (l) Mixed contexts - sublevels

Figure 19: Hierarchical task with mixed statistics

Conceivably, these compositions could have independent, joint, or mixed statistics, and our framework would be able to discover and transfer such transition function substructures. On a higher level of abstraction, high-dimensional MDPs, as one finds in sensory-rich naturalistic tasks, can often be compressed to discover regions of state spaces that are equivalent for the sake of planning. Discovery of such lower dimensional latent states allows an agent to transfer to novel MDPs with entirely novel reward and transition functions, as long as they retain the abstract latent structure [13]. Indeed, this work showed that one could discover useful abstractions in a guitar playing task (e.g., that all fret positions that yield a "C" note are equivalent) that allows an agent to much more rapidly learn new scales. This framework would be yet more powerful if it could compose and transfer substructures without requiring the previous abstraction to hold in all regions of the state space.

Previous work has also cast the learning of latent task components and structures as a problem of non-parametric Bayesian inference [22, 10, 11, 12, 13, 31], and our work continues in this same vein; but Bayesian inference must now be performed over both individual latent components and a potentially infinite variety of structures composed of these components. Bayesian non-parametrics also offers one possible solution to two of the challenges inherent in continual learning. Continual learning requires the agent to continually adapt to an ever-evolving task distribution. However, many approaches to multi-task learning, such as meta-learning [23, 24, 25, 26, 27], require that the task distribution be static, making them less suitable for the continual, online learning setting without some modification. Specifically, the agent will learn a set of features reflecting their relative prevalence across multiple tasks, allowing the agent to adapt quickly to new tasks when sampled from the same distribution. But should the distribution change significantly, as may be the case in the continual learning setting, this bias can lead to a form of negative transfer as the best set of relevant features may change. As we have seen, Bayesian non-parametrics offers one possible solution to this problem by allowing for priors over shared task features that can continually adapt and update as more tasks are encountered. Continual learning also requires that learning of novel tasks not interfere with knowledge of familiar ones, and Bayesian non-parametrics offers a solution by assigning different tasks to different task clusters.

Yet, there are also limitations to our approach which future work will need to address. Currently, our approach relies on signalled context changes (e.g., that the agent has entered a new room), whereas such changes can be more latent in naturalistic tasks and must therefore be inferred from observations. A natural way to address this issue would be to introduce hidden Markov model-like inference to determine context boundaries. Another limitation is that in naturalistic tasks, familiar contexts can change over time, making the initial cluster assignment no longer appropriate after a certain point. In such cases, it may make more sense to reassign the context to a different cluster rather than have cluster assignments be static and irreversible.

We note that there are alternative ways in which the hierarchical agent could have modelled the hierarchical task, each expressing alternative hierarchies and conditional dependencies between task components – our choice above was by no means unique. A more complete agent could potentially infer the best model from task interactions, adding yet another layer of complexity to our approach. We have avoided this extension in our work, however, as our focus was to demonstrate the general benefits that even a single hierarchical model can offer.

Given the model-based setting of our work, as in other such schemes, agents were endowed with knowledge that there are task components such as transition functions and reward functions that can be used for planning. In this sense, the hierarchical structure given to to our agents can be regarded as part of the task model that a model-based agent would have access to. Similarly, in the hierarchical task with sublevels, the agents also knows of the task's hierarchical structure (e.g., that doors in upper levels may lead to sublevels, which in turn have exits). Yet, in both hierarchical and non-hierarchical settings, the agents do not know the parameters of such transition and reward functions, or how they combine them, and have to infer them to plan correctly. Indeed, as we mentioned above, the agents that operate on hierarchical tasks can be regarded as a generalisation of the agents that operate in the non-hierarchical tasks. If these more general agents were to solve the non-hierarchical tasks, they would simply not recruit the parts of their hierarchy used to represent the sublevels or the unused doors in the door sequences. Analogously, human cognitive neuroscience studies have shown that humans can flexibly shift between hierarchical and non-hierarchical tasks by recruiting increasingly anterior regions of the prefrontal cortex-basal ganglia (PFC-BG) circuitry and that they can learn to do so via reinforcement learning [15].

It remains an open question of how our hierarchical model can be implemented in neural circuits. Previous work using neural network models of the PFC-BG circuitry has shown that such models are capable of approximating non-parametric Bayesian inference over task-sets and other hierarchical task structures using a mixture of experts scheme and gating [14, 11], allowing the appropriate networks (and levels of hierarchy) to be deployed for a given task. Several other related neural network schemes have been advanced to model cortical recurrent networks for adaptive learning [32, 24] and even for guiding waves of dopamine across the striatum to support adaptive task learning in mice [33]. In models of category learning, neural networks have successfully learned clusters of substructures somewhat analogous to those studied here [34]. However, to date such networks have not been applied to sequential decision making in the model-based RL setting, which remains a considerable challenge for neural networks.

Our framework offers a normative perspective on what is necessary computationally for an agent to learn hierarchical, compositional task structures. And through our empirical results, we have validated this, showing that only an agent that learned structures hierarchically was capable of supporting compositional generalisation as well as effective transfer of structural knowledge to a wider range of tasks.

Code availability

All code will be made available on github by the time of publication.

Acknowledgements

The authors would like to thank Nicholas Franklin and Cris Buc Calderon for very helpful comments in improving the quality of the manuscript. We also gratefully acknowledge the use of Nicholas Franklin's code [19] (`https://github.com/nicktfranklin/IndependentClusters`), upon which we have built our work.

References

[1] Collins AGE, Frank MJ. Neural signature of hierarchically structured expectations predicts clustering and transfer of rule sets in reinforcement learning. Cognition. 2016;152:160–169.

[2] Baram AB, Muller TH, Nili H, Gavert MM, Behrens TEJ. Entorhinal and ventromedial prefrontal cortices abstract and generalize the structure of reinforcement learning problems. Neuron. 2020;109:1–11.

[3] Harlow H. The formation of learning sets. Psychological Review. 1949;56:51–65.

[4] Mnih V, et al. Human-level control through deep reinforcement learning. Nature. 2015;518:529–533.

[5] Badia AP, et al. Agent57: Outperforming the Atari Human Benchmark. arXiv:200313350. 2020.

[6] Silver D, et al. Mastering the game of Go with deep neural networks and tree search. Nature. 2016;529:484–489.

[7] Silver D, et al. Mastering the game of Go without human knowledge. Nature. 2017;550:354–359.

[8] Taylor ME, Stone P. Transfer Learning for Reinforcement Learning Domains: A Survey. JMLR. 2009;10:1633–1685.

[9] Kansky K, et al. Schema Networks: Zero-shot Transfer with a Generative Causal Model of Intuitive Physics. arXiv:170604317. 2017.

[10] Gershman SJ, Blei DM, Niv Y. Context, learning, and extinction. Psychological Review. 2010;117:197–209.

[11] Collins AGE, Frank MJ. Cognitive Control Over Learning: Creating, Clustering, and Generalizing Task-Set Structure. Psychological Review. 2013;120:190–229.

[12] Madarász TJ, Behrens TEJ. Better Transfer Learning with Inferred Successor Maps. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc.; 2019. Available from: https://proceedings.neurips.cc/paper/2019/file/274a10ffa06e434f2a94df765cac6bf4-Paper.pdf.

[13] Lehnert L, Littman ML, Frank MJ. Reward-predictive representations generalize across tasks in reinforcement learning. PLoS Comput Biol. 2020;16:e1008317.

[14] Frank MJ, Badre D. Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis. Cereb Cortex. 2012;22:509–526.

[15] Badre D, Frank MJ. Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 2: Evidence from fMRI. Cereb Cortex. 2012;22:527–536.

[16] Collins AGE, Cavanagh JF, Frank MJ. Human EEG Uncovers Latent Generalizable Rule Structure during Learning. J Neurosci. 2014;34:4677–4685.

[17] Tomov MS, Dorfman HF, Gershman SJ. Neural Computations Underlying Causal Structure Learning. J Neurosci. 2018;38:7143–7157.

[18] Luyckx F, Nili H, Spitzer B, Summerfield C. Neural structure mapping in human probabilistic reward learning. eLife. 2019;8:e42816.

[19] Franklin NT, Frank MJ. Compositional clustering in task structure learning. PLoS Comput Biol. 2018;14:e1006116.

[20] Franklin NT, Frank MJ. Generalizing to generalize: Humans flexibly switch between compositional and conjunctive structures during reinforcement learning. PLoS Comput Biol. 2020;16:e1007720.

[21] Teh YW, Jordan MI, Beal MJ, Blei DM. Hierarchical Dirichlet Processes. Journal of the American Statistical Association. 2006;101:1566–1581.

[22] Wilson A, Fern A, Ray S, Tadepalli P. Multi-task reinforcement learning: A hierarchical bayesian approach. In: Ghahramani Z, editor. Proceedings of the 24th International Conference on Machine Learning. Omni Press; 2007. p. 1015–1022. Available from: https://icml.cc/imls/conferences/2007/proceedings/papers/463.pdf.

[23] Wang JX, Kurth-Nelson Z, Tirumala D, Soyer H, Leibo JZ, Munos R, et al. Learning to reinforcement learn. arXiv:161105763. 2016.

[24] Wang JX, Kurth-Nelson Z, Kumaran D, Tirumala D, Soyer H, Leibo JZ, et al. Prefrontal cortex as a meta-reinforcement learning system. Nat Neurosci. 2018;21:860–868.

[25] Duan Y, Schulman J, Chen X, Bartlett PL, Sutskever I, Abbeel P. RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. arXiv:161102779. 2016.

[26] Finn C, Abbeel P, Levine S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In: Precup D, Teh YW, editors. Proceedings of the 34th International Conference on Machine Learning. vol. 70 of Proceedings of Machine Learning Research. PMLR; 2017. p. 1126–1135. Available from: http://proceedings.mlr.press/v70/finn17a.html.

[27] Finn C, Xu K, Levine S. Probabilistic Model-Agnostic Meta-Learning. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R, editors. Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc.; 2018. Available from: https://papers.nips.cc/paper/2018/file/8e2c381d4dd04f1c55093f22c59c3a08-Paper.pdf.

[28] Sutton RS, Precup D, Singh S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. Artificial Intelligence. 1999;112:181–211.

[29] Botvinick MM, Niv Y, Barto AC. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. Cognition. 2009;113:262–280.

[30] Konidaris GD, Barto AG. Building portable options: Skill transfer in reinforcement learning. In: Sangal R, Mehta H, Bagga RK, editors. Proceedings of the Twentieth International Joint Conference on Artificial Intelligence. vol. 20. Morgan Kaufmann Publishers Inc.; 2007. p. 895–900. Available from: `https://www.ijcai.org/Proceedings/07/Papers/144.pdf`.

[31] Lehnert L, Littman ML. Successor Features Combine Elements of Model-Free and Model-based Reinforcement Learning. JMLR. 2020;21:1–53.

[32] Tsuda B, Tye KM, Siegelmann HT, Sejnowski TJ. A modeling framework for adaptive lifelong learning with transfer and savings through gating in the prefrontal cortex. PNAS. 2020;117:29872–29882.

[33] Hamid AA, Frank MJ, Moore CI. Wave-like dopamine dynamics as a mechanism for spatiotemporal credit assignment. Cell. 2021;184:2733–2749.e16.

[34] Love BC, Medin DL, Gureckis TM. SUSTAIN: A Network Model of Category Learning. Psychological Review. 2004;111:309–332.