Mixed-Initiative Bayesian Sub-Goal Optimization in Hierarchical Reinforcement Learning

Haozhe Ma
National University of Singapore
Singapore
haozhe.ma@comp.nus.edu.sg

Thanh Vinh Vo
National University of Singapore
Singapore
votv@comp.nus.edu.sg

Tze-Yun Leong
National University of Singapore
Singapore
leongty@comp.nus.edu.sg

ABSTRACT

In hierarchical reinforcement learning, human expertise is often involved in defining sub-goals that decompose the final objective into relevant sub-tasks. However, existing approaches with human-defined sub-goals often lack crucial information about their correlations, limiting their applicability in environments with multiple parallel tasks or mutually conflicting solutions. To address this issue, we propose a mixed-initiative Bayesian sub-goal optimization algorithm that combines human expertise with AI automated reasoning to identify reasonable sub-goals. Our algorithm employs a probabilistic graphical model to capture the correlations among the candidate sub-goals and refine the encoded knowledge to reduce the introduced biases. We conduct experiments in high-dimensional environments with both discrete and continuous controls. In comparison with relevant baselines, our algorithm can achieve better performance in effectively solving problems with multiple selectable solutions. We have empirically demonstrated that our approach is robust against varying levels of human knowledge and expertise, consistently converging to optimal hierarchical policies even amidst misleading or conflicting human guidance.

KEYWORDS

Hierarchical Reinforcement Learning, Human-AI Collaboration, Sub-Goal Optimization, Markov Random Field

ACM Reference Format:
Haozhe Ma, Thanh Vinh Vo, and Tze-Yun Leong. 2024. Mixed-Initiative Bayesian Sub-Goal Optimization in Hierarchical Reinforcement Learning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Hierarchical reinforcement learning (HRL) solves problems at different levels of abstraction, leading to promising breakthroughs in many complex domains. The hierarchical approach is especially effective in long-duration tasks with delayed and sparse rewards, where naive exploration strategies such as ϵ-greedy, action-space noise injection [8], Boltzmann exploration [4], etc., do not work well. HRL can also alleviate the heavy computational burden in high-dimensional problems. Moreover, some general policies can be easily transferred to similar tasks, enabling the reuse of learning results. Goal-conditioned frameworks, which are based on the common divide-and-conquer strategy that humans usually use to complete real-life tasks, are one major research branch in HRL. By setting associate sub-goals and pursuing them sequentially, the overall task can be accomplished with high-level indicated targets.

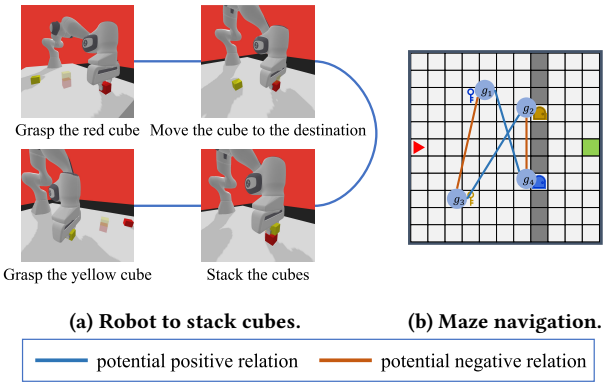

Figure 1: Examples to show the correlations among sub-goals.

Many state-of-the-art efforts have focused on algorithms employing a two-level hierarchical model [19, 22, 34, 35, 46, 48]: the higher level optimizes the policy to select a sub-goal representing a short-term task; the lower level learns the sub-policies to achieve the targeted sub-goals. The sub-goal space plays a vital role in controlling the sparsity of task decomposition and learning performance. However, defining an appropriate set of sub-goals often requires extensive expert knowledge, which makes it hard to scale to complex domains, particularly where human expertise tends to be limited. Moreover, the sub-goal space introduces biases, and in severe cases, may lead to sub-optimal policies. In such scenarios, intelligent agents should assist in mitigating biases and excluding knowledge that may divert the agent into "risky" or "dangerous" regions of the solution space. Furthermore, many existing methods define fixed sub-goal space with unalterable and independent sub-goals, lacking crucial information about the correlations among these sub-goals [27, 28]. However, in many tasks, these correlations are highly significant. For example, consider a scenario where a robot aims to stack a yellow cube on top of a red cube (see Figure 1a) [11]. The successful movement of the red cube to the desired location is highly dependent on the robot's ability to grasp the corresponding cube accurately. Additionally, the successful stacking of the yellow cube is highly dependent on the prior placement of the red cube. In another example, an agent navigates a maze, collecting the key matching the door color to progress (see Figure 1b) [6].

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

In this scenario, "collecting the yellow key" (g_1) and "opening the yellow door" (g_2) are positively correlated (similarly for g_3 and g_4). However, the agent only needs to collect one of the two keys to open one of the two doors, so there exist mutual exclusive relations between "collecting the yellow key" (g_1) and "collecting the blue key" (g_3) (similarly for g_2 and g_4), indicating the potential negative correlations between the sub-goals. Therefore, encoding both positive and negative correlations among sub-goals will substantially improve decision making efficiency in these contexts.

Mixed-initiative interaction refers to a flexible strategy where humans and AI collaborate and contribute what it is best suited at the most appropriate time [1]. Building on a mixed-initiative approach, we propose a **HU**man knowledge based **B**ayesian **S**ub-**G**oal **O**ptimization (**HUBS-GO**) algorithm that can be embedded into HRL frameworks. In contrast to the approaches that rely entirely on a fixed sub-goal space [13, 19, 22, 34, 48], our HUBS-GO maintains a dynamic sub-goal set along the agent training to timely provide filtered sub-goals which leads faster to optimal policies. Unlike the algorithms that pursue fully automatic search [16, 25, 26, 31, 42], our algorithm takes advantage of general and domain-specific knowledge, which is encoded in an initial model. Given a set of candidate sub-goals and their potential correlations based on general human knowledge, HUBS-GO learns a critic function to evaluate their utilities of being selected. Our algorithm is flexible to be embedded into the goal-conditioned HRL frameworks without modifying the original architecture. The critic function can be updated synchronously while training the agent. The timely learned function examines each candidate sub-goal to discriminate its relevance and significance in the exploration process and progressively guide the agent through the dynamically filtered sub-goal space. With HUBS-GO embedded, the agent can determine a more reasonable sub-goal space and converge to the optimal hierarchical policies based on the selected sub-goals.

We conduct experiments in challenging high-dimensional environments with both discrete and continuous controls to assess our approach from three perspectives. First, we show that HUBS-GO can successfully identify the reasonable (or human-intuitively-understandable) sub-goals with detailed distributions. Second, we integrate HUBS-GO into HRL frameworks and compare its performance against popular hierarchical RL methods with both predefined sub-goal space [22, 34] and automatically detected sub-goal space [25], as well as flat RL algorithms [10, 38, 45]. We empirically show that our algorithm works well by effectively leveraging and refining human knowledge and outperforms the baselines. Lastly, we demonstrate that HUBS-GO is robust with respect to different levels of human knowledge in the sub-goal specification, even in the face of potentially misleading or confusing guidance.

2 BACKGROUND AND RELATED WORK

A reinforcement learning (RL) agent plans and acts in an unknown environment modeled as a Markov decision process (MDP). An MDP is defined as a tuple $\langle S, A, T, R, \gamma \rangle$ where S is the state space, A is the action space, T is the transition function, and R is the reward function. A discounted factor γ is introduced for problems with infinite horizons to weigh the importance of future rewards. An RL agent aims to derive an optimal policy $\pi : S \rightarrow A$ that maximizes the expected discounted rewards over the decision horizon [43]. Model-free RL algorithms can be roughly divided into two categories: value-based and policy-based. Value-based methods learn the Q-functions $Q(s, a)$ for each state-action pair. The optimal policy is derived by selecting the action to receive maximal Q-values. Policy-based methods represent the policy explicitly as $\pi_\theta(a|s)$ and optimize the parameters θ by gradient ascent over a performance objective. There is also a range of hybrid algorithms that combine the two approaches, called actor-critic methods. For high-dimensional domains, deep reinforcement learning (DRL) methods adopt deep neural networks as function approximators for the large state or action spaces [10, 14, 15, 24, 32, 33, 38, 45].

The mathematical model underlying all the hierarchical reinforcement learning methods is the semi-Markov decision process (SMDP). An SMDP extends an MDP, such that each macro-action may take multiple time steps to complete, i.e., the transition function $P(s', \tau|s, a)$ is the probability of transiting to a new state s' in τ steps, after executing action a in state s [44]. An HRL model contains multiple levels of temporal abstraction or granularity on problem descriptions [35]. In the single-agent, single-task domain, the policies at various levels can be learned synchronously in an end-to-end manner [2, 22, 23, 34] or asynchronously in a level-by-level manner [7, 9, 29]. Unlike the *options* framework, which defines multiple levels of macro-actions [18, 20, 30, 37], our work is rooted in the goal-conditioned approach, which adopts a divide-and-conquer strategy. In this strategy, the higher level specifies sub-goals to decompose the long-horizon trajectory into sub-tasks, while the lower level learns the corresponding sub-policies to accomplish these sub-goals. Recent research has placed significant emphasis on the definition, discovery, learning, and optimization of sub-goals, as they directly control the task decomposition and significantly influence the learning performance.

Many existing studies operate under the assumption that sub-goals are handcrafted manually based on human prior knowledge. For instance, the *h-DQN* algorithm employs a Q-learning-based approach to learn high-level sub-goals [22]. The *FuNs* algorithm extends the feudal RL to hierarchical scenarios [46]. Additionally, the *HRAC* optimizes this framework by constraining the next sub-goal within an adjacency range [48], while *HAC* constructs hierarchies with more than two levels by independently training each level to overcome the instability issues [23]. However, these algorithms typically require extensive domain expertise and involve labor-intensive processes for defining the sub-goal space. To address these challenges, alternative approaches have focused on automatically discovering sub-goals. There are two major classes: to learn hierarchical policy in *UNIfication with sub-goals discovery* (*UNI*) or *Independent of Sub-goals Discovery (ISD)* [35].

Some UNI approaches learn the sub-goals derived from different kinds of option-critics [2, 18, 20, 37]. However, learning is very sensitive to the defined critics which may lead to unstable performance. Some algorithms discover the low-dimensional sub-goal space, like the *HADS* learns the sub-goals based on some visible feature extraction [25], or the algorithm learns an abstract mapping function to reduce the high-dimensional observations to low-dimensional information. The ISD approaches learn task-agnostic sub-goals independently from hierarchical learning. Sub-goal discovery is

usually performed in a pre-training process and the detected sub-goals are transferable to HRL agents across various similar tasks. Some straightforward methods aim to find the bottlenecks by frequency [29, 39] or graphs of transition histories [40]. Many novel approaches attempt to scale up in high-dimensional domains. Both the *SDRL* [26] and *Oracle-SAGE* [5] discover sub-goals with symbolic representation, and the latter additionally incorporates graph neural networks to represent the observations. The *HIRO* algorithm introduces the goal-conditioned framework to continuous controls [34]. The *HASSLE* algorithm searches for the centroids of the state space clusters [3]. The *HSP* algorithm introduces the asymmetric self-play to learn the sub-goals using a continuous latent embedding [41]. The *QRM* algorithm specifies high-level tasks from learning a reward machine to decompose the trajectory [16].

We aim to examine the power and advantages of both humans' prior knowledge and machines' automatic learning ability to develop a sub-goal optimization method. Our proposed algorithm is a novel idea in the UNI branch, but also shares the characteristics of ISD methods, e.g., the sub-goals can be pre-learned before we optimize the agent policy and can be easily transferred to similar tasks. The *HAI-GO* algorithm proposed a human-AI collaborative framework to automatically optimize the pre-defined sub-goals [27], however it only considered the sub-goals as independent of each other. Our approach introduces crucial information regarding the correlations among sub-goals and relaxes the assumption that human knowledge must be precisely defined. Our HUBS-GO enables the refinement of prior knowledge, making it more "forgiving" by allowing human knowledge to be more arbitrary and general during the model encoding process, without compromising performance.

3 METHODOLOGY

The main idea of our HUBS-GO is to combine the advantages of human expertise and the automatic learning ability of AI to optimize sub-goals. Human experts specify some candidate sub-goals and their correlations based on general knowledge, AI agent then extracts an optimal sub-goal space. In HUBS-GO, a critic function is employed to define a probabilistic graphical model over the candidate sub-goals, capturing their potential weights. The critic function will gradually provide the agent with a dynamically filtered sub-goal set to support more concise and accurate knowledge updates, until convergence to the optimal policy. HUBS-GO can optimize the prior knowledge, eliminate the introduced bias, and finally obtain a reasonable sub-goal set based on the learned distribution so that the winning sub-goals can be used for future learning or transferring to similar tasks.

3.1 Problem Formulation

We consider a fully observable environment $\mathcal{E}$. Let S be the state space and A be the action space. Suppose $G = \{g_1, g_2, \ldots, g_N\}$ contains human-defined sub-goals and $E = \{(g_u, g_v)_i\}$ contains the connected edges between two sub-goals to denote their correlations. We define a Markov random field (MRF) to model the relations among the candidate sub-goals. We define the random variable $\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$ as indicators to represent whether to select the corresponding sub-goals or not. The critic function, denoted as $q(\mathbf{w}; \lambda)$, represents the distribution of $\mathbf{w}$.

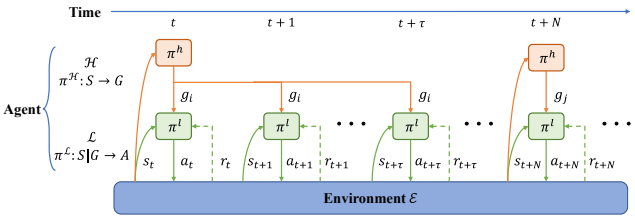

Figure 2: The two-level goal-conditioned hierarchical reinforcement learning framework prototype.

We initialize a two-level HRL agent: the high level $\mathcal{H}$ optimizes an approximator $Q(s, g; \theta)$ to estimate the Q-values for each state-sub-goal pair $\{s, g\}$, where θ are the parameters; the low level $\mathcal{L}$ optimizes a policy to select elementary actions given the state and the targeted sub-goal $a = \pi(s|g)$. Our algorithm synchronously learns both the hierarchical policies and the critic function. HUBS-GO finally derives an optimal sub-goal space G^* and optimal hierarchical policies after training.

3.2 Preliminaries: Goal-Conditioned HRL Framework

HUBS-GO can be embedded into the general goal-conditioned HRL frameworks. Figure 2 demonstrates a prototype with a common two-level structure. At the time step t, given the state s_t of the environment, the high level $\mathcal{H}$ selects a sub-goal g_i from sub-goal space G based on its policy $\pi^{\mathcal{H}}$. The sub-goal represents a short-term task that the agent is expected to complete in the next few steps. Given the sub-goal g_i as well as the state s_t, the low level $\mathcal{L}$ selects an elementary action to interact with the environment based on its policy $\pi^{\mathcal{L}}$. The environment will transit to a new state s_{t+1} while returning a reward signal r_{t+1}. Usually, the two levels operate at different scales of temporal abstraction. In this case, after each step of the high level, the low level will act for N steps, where $N > 1$, represents the expected number of steps that the low level can complete a specific sub-goal. The high-level policy revises a new sub-goal g_j after N steps or when the low level successfully achieves the current target.

The high-level interaction can be modeled by an MDP denoted as $\langle S, G, T^{\mathcal{H}}, R^{\mathcal{H}}, \gamma^{\mathcal{H}} \rangle$. The main goal is to learn an optimal high-level policy $\pi^{h^*} : S \rightarrow G$ to maximize the discounted high-level cumulative rewards $G_t^{\mathcal{H}} = \sum_{\tau=t}^{\infty} \gamma^{h^{\tau-t}} r_t^{\mathcal{H}}$. The low-level interaction can be modeled by an MDP denoted as $\langle S, A, T^{\mathcal{L}}, R^{\mathcal{L}}, \gamma^{\mathcal{L}} \rangle$. The low-level policy is conditioned on the targeted sub-goal that is instructed by the high level. Given the state s_t as well as the sub-goal g_i, the low-level policy aims to learn a policy to select the elementary action $a_t = \pi^{\mathcal{L}}(s_t|g_i)$. The policies can be derived by any applicable flat RL algorithms, allowing for flexibility in implementation across discrete and continuous control domains.

The hierarchical framework with the high level learning sub-goal policy and the low level learning action policy provides the foundation for many HRL algorithms. We will show how HUBS-GO can be embedded into this general framework in Section 3.3.

3.3 Bayesian Sub-Goal Optimization

A well-trained sub-goal policy can decompose the final objective into some sub-tasks and lead the low level to achieve the target more efficiently. However, it is hard to introduce the correlation information by only defining sub-goals. More importantly, human knowledge will inevitably introduce bias, since the pre-defined sub-goals are not necessarily optimal. To address these challenges, our proposed algorithm allows human experts to define a set of candidates and model their correlations with very general knowledge and leaves the AI to automatically optimize the sub-goal space and learn an optimal policy. The HUBS-GO learns an additional critic function to model the distribution over the candidates to evaluate their utilities, based on which, the high level will gradually select one sub-goal from a dynamic filtered sub-goal set.

Given the human-specified sub-goal space $G = \{g_1, g_2, \ldots, g_N\}$, based on their potential correlations from a human perspective, experts indicate a set of correlation connections as $E = \{(g_u, g_v)_i\}$. The critic function can be defined as $q(\mathbf{w}; \lambda)$ over the indicator variable $\mathbf{w} = [w_1, w_2, \ldots, w_N]^T$, to represent the MRF model. Each entry $w_i \in \{0, 1\}$ corresponds to each candidate sub-goal g_i and indicates whether to select it by $w_i = 1$ or not to select it by $w_i = 0$. The connecting edge $(g_u, g_v)_i$ indicates either a potential positive or negative correlation between the sub-goal g_u and g_v. For example, "cleaning the kitchen" may positively correlate with "making dinner", while "making dinner" may negatively correlate with "moving to the bathroom". In cases where there is no relevant prior knowledge available, it is often advantageous to consider a fully connected graphical model that takes into account all possible connections. The critic function can be modeled by defining a pairwise MRF distribution [21]:

$$q(\mathbf{w}; \lambda) = \frac{1}{Z(\lambda)} \prod_{(g_u, g_v) \in E} e^{\phi_i(w_u, w_v; \lambda_i)} \prod_{g_j \in G} e^{\psi_j(w_j; \lambda_j)}, \tag{1}$$

where λ is the parameters to learn, $Z(\lambda)$ is the normalization factor, ϕ_i and ψ_j are the potential functions of the connected sub-goals pair and the independent indicator respectively.

To integrate the critic function back into the agent learning, we draw a sample $\mathbf{w}_t$ from the learned distribution for each high-level iteration and generate a subset $\hat{G}_t$ that only contains the sub-goals whose corresponding entry $w_{ti} = 1$. The high level will only select one sub-goal from the filtered $\hat{G}_t$. To implement a Q-learning-based approach, the high-level module learns an approximator $Q^{\mathcal{H}}(s, g; \theta)$ to estimate the Q-values for each state-sub-goal pair $\{s, g\}$. Given the dynamic sub-goal set $\hat{G}_t$, we define the temporal difference target (TD-target) that is conditioned on $\hat{G}_t$ as:

$$y_{\hat{G}_t} = r_t^{\mathcal{H}} + \gamma^{\mathcal{H}} \max_{g' \in \hat{G}_t} Q^{\mathcal{H}}(s_{t+N}, g'; \theta).$$

The parameters θ will be updated by minimizing the temporal difference error (TD-error), i.e., the distance between $y_{\hat{G}_t}$ and the predicted Q-value $Q^{\mathcal{H}}(s_t, g_t; \theta)$, so the loss function conditioned on the sub-goal space $\hat{G}_t$ can be written as:

$$L_{\hat{G}_t} = \frac{1}{2}\big(y_{\hat{G}_t} - Q^{\mathcal{H}}(s, g; \theta)\big)^2. \tag{2}$$

To make the critic function optimally evaluate the utilities of each candidate, the main objective is to update the $q(\mathbf{w}; \lambda)$ to be the best approximation to the real posterior $p(\mathbf{w}|y_{\hat{G}_t})$. The posterior gives the distribution of indicator $\mathbf{w}$ conditioned on the corresponding TD-target $y_{\hat{G}_t}$. To proceed, a black-box variational inference (BBVI) approach [36, 47] is adopted to optimize the parameters λ, that is, to minimize the KL-divergence of $q(\mathbf{w}; \lambda)$ and $p(\mathbf{w}|y_{\hat{G}_t})$:

$$\begin{aligned} D_{KL}\big(q(\mathbf{w}; \lambda)||p(\mathbf{w}|y_{\hat{G}_t})\big) &= D_{KL}\big(q(\mathbf{w}; \lambda)||p(\mathbf{w})\big) \\ &\quad - \mathbb{E}_{\mathbf{w} \sim q(\mathbf{w}; \lambda)}\big[\log p(y_{\hat{G}_t}|\mathbf{w})\big] + constant, \end{aligned} \tag{3}$$

where $p(\mathbf{w})$ is a prior distribution that can be initialized as a categorical distribution with uniform probabilities for all entries. Furthermore, based on the least squares method, we can regard $y_{\hat{G}_t}$ as observed value and $Q(s, g; \theta)$ as predicted value. The squared error in Equation 2 implies a Gaussian error around the target value, which can be rephrased as the negative-log-likelihood of the model:

$$y_{\hat{G}_t} = Q^{\mathcal{H}}(s, g; \theta) + \epsilon_{\hat{G}_t}, \quad \epsilon_{\hat{G}_t} \sim \mathcal{N}(0, \sigma^2),$$

which will result in the log-likelihood as:

$$\begin{aligned} \log p(y_{\hat{G}_t}|\mathbf{w}) &= -\frac{1}{2}\big(y_{\hat{G}_t} - Q^{\mathcal{H}}(s, g; \theta)\big)^2 + constant \\ &= -L_{\hat{G}_t} + constant. \end{aligned} \tag{4}$$

Thus, substituting Equation 4 into Equation 3 yields the final loss function (detailed derivation is shown in Appendix):

$$L(\lambda) = \mathbb{E}_{\mathbf{w} \sim q(\mathbf{w}; \lambda)}[L_{\hat{G}_t}] + D_{KL}\big(q(\mathbf{w}; \lambda)||p(\mathbf{w})\big).$$

The loss function requires the computation of expectation and KL-divergence over the MRF distribution, which is computationally challenging. To overcome this challenge, we resort to estimating the two terms by drawing a batch of independent samples from the distribution and computing their empirical values. For sampling from the pairwise MRF distribution, a Metropolis-Hasting-based Markov Chain Monte Carlo (MCMC) algorithm is adopted [12]. In each iteration, following a burn-in period of 32 steps, we draw 64 samples using the current learned parameters λ_t. The samples will be used to estimate the gradient of the loss function in Equation 5. The parameters will be updated to λ_{t+1} by a gradient descent approach with a fine-tuned learning rate. We adopt gradient descent to minimize the loss function. For each update, we draw S samples from $q(\mathbf{w}; \lambda)$ (the detailed method is shown in Appendix) based on the current parameters and estimate the gradient as:

$$\nabla_\lambda L(\lambda) \approx \frac{1}{S} \sum_{s=1}^{S} \nabla_\lambda \log q(\mathbf{w}_s; \lambda_t)\big(L_{\hat{G}_t} + \log q(\mathbf{w}_s; \lambda_t) - \log p(\mathbf{w}_s)\big). \tag{5}$$

Lastly, we present one special case of our MRF model, that is, all sub-goals are independent of each other, in other words, the correlated connections set $E = \emptyset$. The critic function can be simplified as a set of Bernoulli distributions: $\mathbf{q} = \{q_i(w_i; \lambda_i)\}$, where $i \in \{1, 2, \ldots, N\}$. In this scenario, the loss function for each λ_i can be separately computed and summed up as:

$$L(\lambda) = \sum_{i=1}^{N} \mathbb{E}_{w_i \sim q_i(w_i; \lambda_i)}[L_{\hat{G}_t}] + D_{KL}\big(q_i(w_i; \lambda_i)||p_i(w_i)\big).$$

To draw samples from the Bernoulli distributions, the Gumbel-Softmax is implemented to reparameterize, which effectively reduces the computation [17]. A detailed derivation is shown in the Appendix.

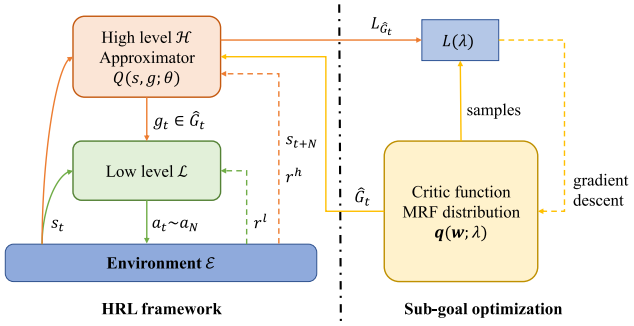

Figure 3: An overview of the HRL framework with the integration of our proposed HUBS-GO algorithm.

Algorithm 1 HRL frameworks with HUBS-GO

Require: Environment $\mathcal{E}$
Require: Agent with high-level $\mathcal{H}$ and low-level $\mathcal{L}$
Require: Human defined candidate sub-goal space $G = \{g_i\}$
Require: Human defined correlation connections $E = \{(g_u, g_v)_i\}$
Require: MRF distribution critic function $q(\mathbf{w};\lambda)$
for each high-level iteration **do**
  $g_t = \mathcal{H}(s_t)$ with ϵ-greedy policy filtered $\hat{G}_t$
  for each low-level iteration **do**
    $a_\tau = \mathcal{L}(s_\tau | g_t)$ based on some exploration strategy
    Act a_τ in $\mathcal{E}$, receive $s_{\tau+1}$, $r_{\tau+1}$
    Update $\mathcal{L}$ with some RL algorithm
    Break if agent achieves g_t
  end for
  Draw a batch of samples from $q(\mathbf{w};\lambda)$
  Compute the gradient of the loss as Equation 5
  Update parameters λ by gradient descent
end for

3.4 Dynamically Filtered Sub-Goals

Our HUBS-GO algorithm not only learns the critic function along with the agent learning but also provides a dynamic sub-goal space to help the agent gradually converge to optimal policies. In this section, we present how HUBS-GO is embedded into the HRL framework by an ϵ-greedy strategy to trade-off between exploration and exploitation. An overview of the interaction of the two modules is shown in Figure 3.

On the one hand, we update the critic function based on the high-level module by drawing samples from $q(\mathbf{w};\lambda)$, computing the conditional loss functions $L_{\hat{G}_t}$ in Equation 2 and estimating the gradient in Equation 5, as stated in Section 3.3. On the other hand, HUBS-GO synchronously provides the filtered sub-goal set $\hat{G}_t$ to the high-level module. To generate $\hat{G}_t$, we re-utilize the samples that are already drawn for critic function optimization previously, to estimate the probabilities for each configuration of the indicator variable $\mathbf{w}_t$. Consequently, the $\hat{G}_t$ contains only the sub-goals indicated by the greedily selected $\mathbf{w}_t$ with the highest empirical probability. With an increasing probability of ϵ, the high level will only select sub-goals from $\hat{G}_t$; while with a probability of $1-\epsilon$, the high level returns the sub-goal normally from the initial candidate space G_0. Whether from $\hat{G}_t$ or G_0 for the next decision step, the high-level module makes its selection based on their respective Q-values: $g_t = \arg\max_{g' \in \hat{G}_t | G_0} Q^{\mathcal{H}}(s_t, g'; \theta_t)$. The gradually increasing ϵ controls the frequency for HUBS-GO to interact with the HRL framework. The main process of the goal-conditioned HRL algorithm embedded with the HUBS-GO is shown in Algorithm 1.

Upon completing the learning process, we can analyze the final MRF distribution and derive an optimal sub-goal space G^* based on the critic function. In our implementation, we draw 10,000 independent and identically distributed (i.i.d.) samples and calculate the empirical probabilities for each configuration of the indicator variable. The configuration yielding the highest probability will determine the final G^*. For instance, when considering three candidates, if the probability of the configuration $\mathbf{w} = \{1, 0, 1\}$ surpasses all other configurations, it suggests selecting the first and third candidates as the final sub-goals. It is worth noting that there may exist multiple configurations that share a similar level of high probabilities, indicating different sub-goal spaces that can all lead to optimal hierarchical policies. Additionally, the learned distribution provides valuable insights into the black-box environment, enhancing our understanding of its underlying dynamics. The complete model enables the achievement of an optimal hierarchical policy by fully exploiting the derived sub-goal space G^*. Moreover, both the derived sub-goals and the policies can be easily transferred to similar tasks, facilitating the adaptation and utilization of learned knowledge in new scenarios.

4 EXPERIMENTS

We conduct experiments in two different types of environments: grid-maze environments [6] with discrete control and arm robot environments [11] with continuous control. The environments provide sparse rewards, where a reward value of 1 is only given upon successful completion of the overall target, and a reward value of 0 is assigned to all other states. In the mini-grid environments, we define two tasks: (a) the *FourRoom* task, the agent navigates to the target location in a maze with interconnected rooms; (b) the *KeyDoor* task, the agent collects one key from two keys to unlock the door in the same color to reach the target location. In the arm robot environments, we define three tasks: (c) the *StackCubes* task, the robot stacks cubes at a designated location; (d) the *PushAnyOne* task, the robot selects one cube and pushes it to the target location; (e) the *CollectThreeBalls* task, the robot collects three out of six balls on the table, with a wall separating the balls into two groups. One main challenge is that there are multiple mutually parallel solution routes for the agent to complete the final task, which requires the agent to select and focus on one possible solution.

The evaluation focuses on three main aspects: in Section 4.1, we examine the ability of HUBS-GO to identify reasonable sub-goal spaces after optimization. In Section 4.2, we compare HUBS-GO embedded HRL algorithm with state-of-the-art baselines and analyze the convergence speed and learning performance. In Section 4.3, we investigate the robustness of HUBS-GO in refining different degrees of encoded human knowledge, including different correlation connections and deliberately introduced misleading sub-goals.

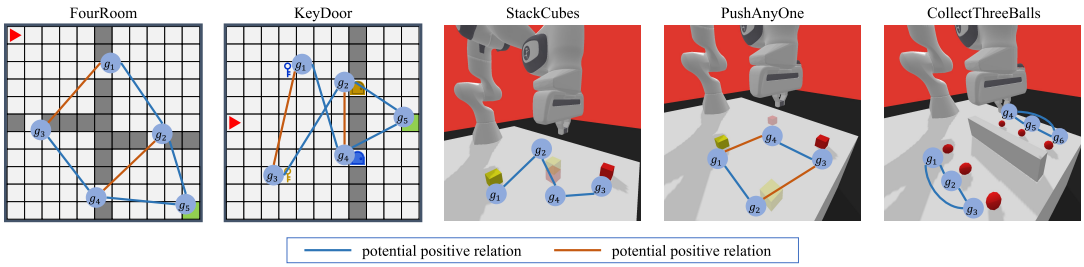

Figure 4: The initial settings of the sub-goals and their connected edges.

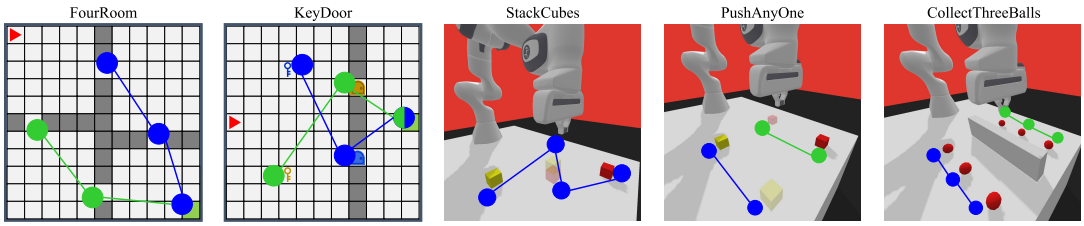

Figure 5: The identified optimal sub-goal sets by HUBS-GO.

4.1 Sub-Goal Optimization

In this section, we assess the optimized sub-goal distributions by HUBS-GO. Prior to the learning process, we define a set of initial sub-goals from a human perspective. Additionally, we give the potential correlations among these sub-goals based on our basic understanding of the environments. The initial settings are illustrated in Figure 4, where the potential positive correlations are represented by blue arcs, and the potential negative correlations are represented by orange arcs.

The suggested sub-goal spaces by HUBS-GO after learning the critic function are illustrated in Figure 5. Notably, the results present an important capability, that is, it may finally indicate multiple sub-goal sets with the similar highest probabilities, each one representing a selectable breakdown of the trajectory (shown with blue and green circles, respectively). In these experimental tasks, a primary challenge is that there exist multiple possible routes for the agent to achieve the final goal. Thus, selecting all keys, doors, or cubes indiscriminately is not an optimal solution, as it may result in the agent repeatedly switching between these options or unnecessarily increasing the workload. For instance, in the KeyDoor task, separately indicating the yellow and blue pairs of key-doors enables the agent to focus on a single solution, which aligns more closely with human intuition and reasoning.

4.2 HUBS-GO Embedded Hierarchical Reinforcement Learning

HUBS-GO dynamically filters the initial sub-goal space and provides a more accurate set of sub-goals for the agent to select from, which is a key factor in improving the performance of the HRL algorithm. In this section, we compare our HUBS-GO with four state-of-the-art HRL baselines: *h-DQN* [22], *HIRO* [34], *HADS* [25] and *HIGL* [19], as well as popular flat RL baselines: *DDQN* [45], *PPO* [38] and *TD3* [10]. The h-DQN and DDQN are only implemented in the grid-maze environments with discrete action spaces, while the TD3 is only implemented in the arm robot environments with continuous action spaces. For each experiment, we run multiple times with different random seeds to show the average performance with standard error.

In terms of sub-goal initializations, the h-DQN algorithm shares the same initial goal space as HUBS-GO. The HADS, HIRO and HIGL algorithms autonomously discover sub-goals. In the practical implementations, we utilize the interfaces provided by the respective environments to define the sub-goals. In the "MiniGrid" environment, sub-goals are identified as targeted coordinates. In the arm robot environment, sub-goals are defined as some abstract descriptions, such as "grasping the cube", the interface interprets the verbal expressions to a set of targeted regions for the robot's arms end to reach.

Figure 6 illustrates the episodic returns throughout training, and Table 1 presents average returns and standard errors. As the results suggest, HUBS-GO shows a slight initial delay at the beginning, followed by rapid convergence to higher and more stable return values. The short lag can be attributed to the process of learning the critic function, while additional exploration is required in the early stages to establish a more accurate distribution of sub-goals. One of the main advantages of HUBS-GO is its ability to capture the correlations among the sub-goals. The critic function learns the distribution of candidate sub-goals, the configurations with high probabilities will separately indicate different possible solutions to complete the task, which further reduces the space of sub-goals to select from. This allows the agent to focus on a single route and avoid hesitation. In the FourRoom task, HUBS-GO efficiently discriminates between two separate routes to the target location by learning the corresponding configuration probabilities, while the baselines struggle by treating all doors equally, leading to repeated

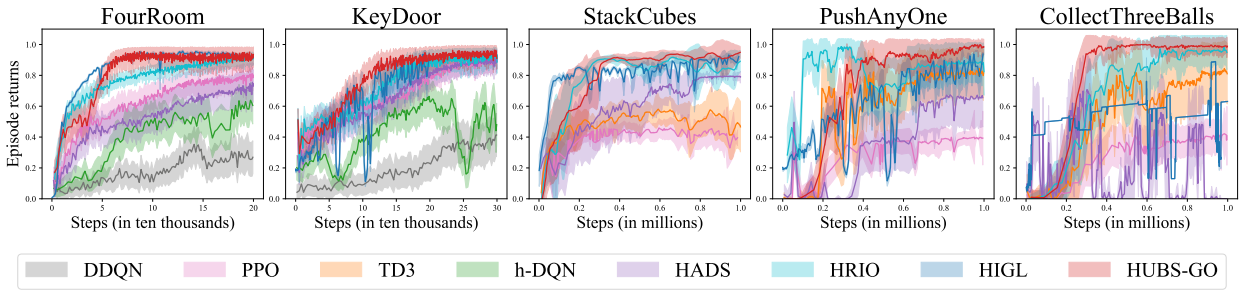

Figure 6: Performance comparison of HUBS-GO with baselines.

Table 1: Average episodic returns of the agent training

Algorithm	Environments				
	FourRoom	KeyDoor	StackCubes	PushAnyOne	CollectThreeBalls
HUBS-GO (specific correlations)	**0.8904 ± 0.0044**	**0.8763 ± 0.0053**	**0.8891 ± 0.0152**	**0.9173 ± 0.0123**	**0.9627 ± 0.0099**
HUBS-GO (fully connected)	0.8416 ± 0.0879	0.8580 ± 0.0999	0.8563 ± 0.1026	0.9095 ± 0.0125	0.9370 ± 0.0190
HUBS-GO (fully independent)	0.7114 ± 0.1196	0.7425 ± 0.0723	0.8209 ± 0.1072	0.8744 ± 0.1154	0.8467 ± 0.1380
DDQN	0.1899 ± 0.0077	0.2107 ± 0.0109	-	-	-
PPO	0.6664 ± 0.0083	0.7567 ± 0.0084	0.3934 ± 0.0068	0.3337 ± 0.0114	0.3122 ± 0.0116
TD3	-	-	0.4890 ± 0.0045	0.6903 ± 0.0144	0.6407 ± 0.0159
hDQN	0.4287 ± 0.0166	0.4584 ± 0.0126	-	-	-
HADS	0.5610 ± 0.0095	0.7603 ± 0.0093	0.7087 ± 0.0085	0.4306 ± 0.0239	0.2694 ± 0.0156
HIRO	0.8250 ± 0.0054	0.8332 ± 0.0064	0.7820 ± 0.0141	0.8730 ± 0.0094	0.8514 ± 0.0193
HIGL	0.8447 ± 0.0172	0.5439 ± 0.0210	0.8105 ± 0.0121	0.8553 ± 0.0209	0.6274 ± 0.0041

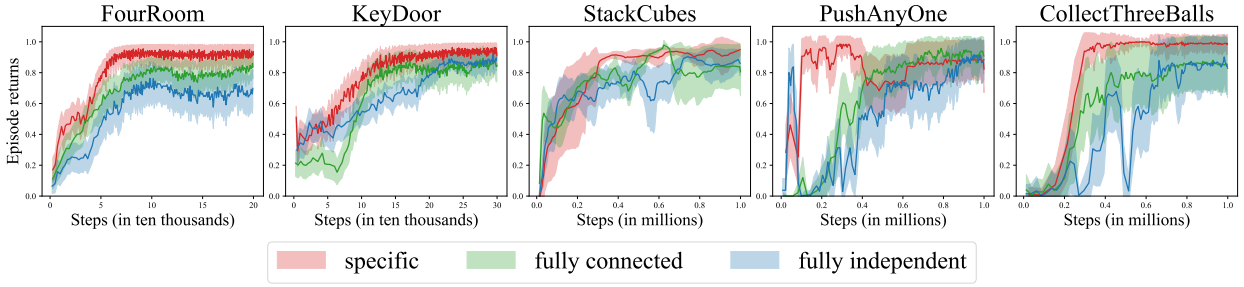

Figure 7: Comparison of HUBS-GO with different pre-identified correlations.

switching between routes. Similarly, in the KeyDoor task, HUBS-GO focuses on picking up one key and opening one door to reach the target, whereas the baselines attempt to pick up both keys and open both doors, resulting in redundant steps. In the StackCubes task, HUBS-GO quickly learns a practical order of sub-goals given the possible dependency relations, while other algorithms take longer to explore these implicit relations. In the PushAnyOne task, HUBS-GO efficiently utilizes provided correlations to match cubes with correct-colored destinations, whereas other algorithms often incorrectly push cubes to different-colored destinations. Lastly, in the CollectThreeBalls task, as the wall divides the balls into two groups, HUBS-GO saves time by collecting from one group, while the baselines treat all balls equally, which spend more time striding over the wall. In summary, the critic function enables HUBS-GO to efficiently solve problems especially when there exist multiple parallel selectable solutions or mutually conflicting tasks.

4.3 Human Knowledge Refinement

Our proposed algorithm incorporates human expert knowledge, but in contrast to most HRL algorithms, a significant advantage of our model is that it does not rely exclusively on this expertise, but instead optimizes the introduced knowledge to provide dynamically filtered sub-goal spaces. In this section, we examine the robustness of HUBS-GO in the face of varying degrees of human knowledge. In addition to setting up the **specific** correlated edges that best match human understanding, we compare with the **fully connected** MRF model to represent introducing general human knowledge, and a

fully independent candidate space (without any correlated edges) to represent the lack of human knowledge.

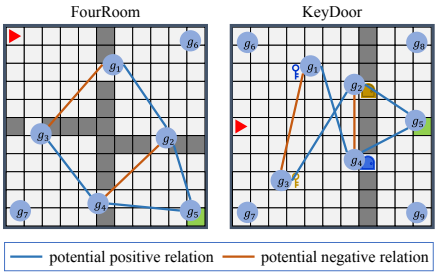

Figure 8: Experimental configurations of introducing incorrect sub-goals into the grid-maze environments.

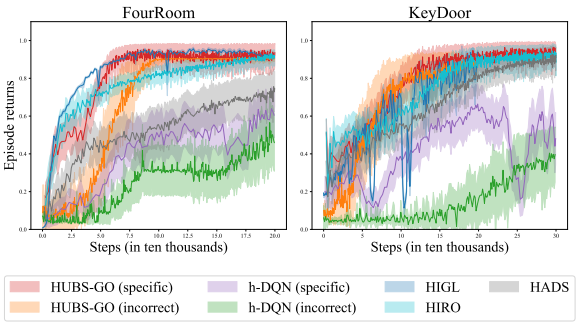

Figure 9: Performance comparison of HUBS-GO with baselines with incorrect sub-goals introduced.

Table 2: Average episodic returns of the agent training with incorrect sub-goals introduced.

Algorithm	Environments	
	FourRoom	KeyDoor
HUBS-GO (specific sub-goals)	**0.8904 ± 0.0044**	**0.8763 ± 0.0053**
HUBS-GO (incorrect sub-goals)	0.7629 ± 0.0163	0.7274 ± 0.0123
h-DQN (specific sub-goals)	0.4287 ± 0.0166	0.4584 ± 0.0126
h-DQN (incorrect sub-goals)	0.2438 ± 0.0068	0.1430 ± 0.0072
HIGL	0.8447 ± 0.0172	0.5439 ± 0.0210
HADS	0.5610 ± 0.0095	0.7603 ± 0.0093
HIRO	0.8250 ± 0.0054	0.8332 ± 0.0064

Figure 7 illustrates the comparison of HUBS-GO with different pre-identified correlations. Table 1 presents the quantitative results and a comparison with the baselines. The results clearly demonstrate that the settings with specific human knowledge achieve the fastest convergence, while the settings with general and poor knowledge perform slower convergence, especially in the fully independent scenario, the agent initially struggles due to the absence of correlations, but eventually converges to the optimal policies. More importantly, the fully connected configuration, despite not involving any additional expertise beyond the sub-goal space, is still able to outperform the baselines.

On the other hand, we also investigate the robustness of HUBS-GO in scenarios where deliberately confusing sub-goals are presented, representing situations where humans may inadvertently introduce incorrect knowledge. In the FourRoom and KeyDoor tasks, we defined additional sub-goals in the corners of the room that could mislead the agent to get far away from the correct solutions, as shown in Figure 8. By comparing HUBS-GO with h-DQN (using the same initial sub-goal space), HIRO, HADS and HIGL, we evaluate the learning performance in Figure 9 and Table 2. The results demonstrate that the incorrect sub-goals will temporarily confuse the high-level model's exploration during the initial stages, but the agent will achieve fast convergence once overcame the misleading sub-goals. Notably, introducing incorrect sub-goals will significantly impact the learning efficiency of h-DQN.

The above results highlight how our algorithm can automatically optimize the human-specified candidate space to refine the introduced knowledge. The introduced correlation plays an indispensable role, that allows the HRL algorithm with integration of HUBS-GO to outperform other baselines. Additionally, the dynamic sub-goal space, which timely reflects the filtered sub-goals, holds great promise for applications in environments where there is insufficient human knowledge.

5 CONCLUSION

We propose HUBS-GO, a mixed-initiative Bayesian sub-goal optimization algorithm. Incorporating the correlations among the sub-goals reduces unnecessary exploration and enables the generation of dynamic sub-goal sets, leading to faster, more stable convergence. Tested in high-dimensional, diverse control environments, HUBS-GO has displayed great prowess in deriving rational sub-goal spaces and learning optimal policies. Its key strength lies in its MRF-based distribution, capable of capturing sub-goal correlations and addressing issues with multiple conflicting solutions and parallel tasks. With the capability to refine the integrated human knowledge, it broadens the scope for human-AI collaboration and heightens robustness by enabling self-optimization of the sub-goal space, an important trait more so in areas where specific prior knowledge is scant. HUBS-GO helps in providing a more comprehensive understanding of black-box environments and can be deployed into the goal-conditioned HRL frameworks without any structural modifications, underlining its high expansibility.

Our initial work invites several avenues for future exploration. First, as the HUBS-GO is built on the assumption that the sub-goals and correlations are responsibly defined and cover at least a subset that provides a positive decomposition of the final task, it still heavily relies on human efforts. Second, as we model the correlations among sub-goals with MRF models, it limits the sub-goal space to discrete values. Extending the MRF models to continuous sub-goal space is also a promising direction. Finally, we need to define better performance measures that take into account human preferences and environmental uncertainties in meeting the objectives in different contexts.

ACKNOWLEDGMENTS

This research is supported by an Academic Research Grant No. MOE-T2EP20121-0015 from the Ministry of Education in Singapore.

REFERENCES

[1] James E Allen, Curry I Guinn, and Eric Horvtz. 1999. Mixed-initiative interaction. *IEEE Intelligent Systems and their Applications* 14, 5 (1999), 14–23.

[2] Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[3] Bram Bakker, Jürgen Schmidhuber, et al. 2004. Hierarchical reinforcement learning based on subgoal discovery and subpolicy specialization. In *Proc. of the 8-th Conf. on Intelligent Autonomous Systems*. Citeseer, 438–445.

[4] Nicolò Cesa-Bianchi, Claudio Gentile, Gábor Lugosi, and Gergely Neu. 2017. Boltzmann exploration done right. *Advances in Neural Information Processing Systems* 30 (2017).

[5] Andrew Chester, Michael Dann, Fabio Zambetta, and John Thangarajah. 2023. Oracle-SAGE: Planning Ahead in Graph-Based Deep Reinforcement Learning. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2022, Grenoble, France, September 19–23, 2022, Proceedings, Part IV*. Springer, 52–67.

[6] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. 2018. Minimalistic Gridworld Environment for Gymnasium. https://github.com/Farama-Foundation/Minigrid

[7] Carlos Florensa, Yan Duan, and Pieter Abbeel. 2016. Stochastic Neural Networks for Hierarchical Reinforcement Learning. In *International Conference on Learning Representations*.

[8] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. 2018. Noisy networks for exploration. (2018).

[9] Roy Fox, Sanjay Krishnan, Ion Stoica, and Ken Goldberg. 2017. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294* (2017).

[10] Scott Fujimoto, Herke Hoof, and David Meger. 2018. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*. PMLR, 1587–1596.

[11] Quentin Gallouédec, Nicolas Cazin, Emmanuel Dellandréa, and Liming Chen. 2021. panda-gym: Open-Source Goal-Conditioned Environments for Robotic Learning. *4th Robot Learning Workshop: Self-Supervised and Lifelong Learning at NeurIPS* (2021).

[12] Will Grathwohl, Kevin Swersky, Milad Hashemi, David Duvenaud, and Chris Maddison. 2021. Oops i took a gradient: Scalable sampling for discrete distributions. In *International Conference on Machine Learning*. PMLR, 3831–3841.

[13] Nico Gürtler, Dieter Büchler, and Georg Martius. 2021. Hierarchical reinforcement learning with timed subgoals. *Advances in Neural Information Processing Systems* 34 (2021), 21732–21743.

[14] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, 1861–1870.

[15] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI Conference on Artificial Intelligence*.

[16] Rodrigo Toro Icarte, Toryn Klassen, Richard Valenzano, and Sheila McIlraith. 2018. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2107–2116.

[17] Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical Reparameterization with Gumbel-Softmax. In *International Conference on Learning Representations*.

[18] Khimya Khetarpal and Doina Precup. 2019. Learning options with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 9955–9956.

[19] Junsu Kim, Younggyo Seo, and Jinwoo Shin. 2021. Landmark-guided subgoal generation in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems* 34 (2021), 28336–28349.

[20] Martin Klissarov, Pierre-Luc Bacon, Jean Harb, and Doina Precup. 2017. Learnings options end-to-end for continuous action tasks. *arXiv preprint arXiv:1712.00004* (2017).

[21] Daphne Koller and Nir Friedman. 2009. *Probabilistic graphical models: principles and techniques*. MIT press.

[22] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. 2016. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in Neural Information Processing Systems* 29 (2016).

[23] Andrew Levy, George Konidaris, Robert Platt, and Kate Saenko. 2018. Learning Multi-Level Hierarchies with Hindsight. In *International Conference on Learning Representations*.

[24] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

[25] Chenghao Liu, Fei Zhu, Quan Liu, and Yuchen Fu. 2021. Hierarchical reinforcement learning with automatic sub-goal identification. *IEEE/CAA Journal of Automatica Sinica* 8, 10 (2021), 1686–1696.

[26] Daoming Lyu, Fangkai Yang, Bo Liu, and Steven Gustafson. 2019. SDRL: interpretable and data-efficient deep reinforcement learning leveraging symbolic planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 2970–2977.

[27] Haozhe Ma, Thanh Vinh Vo, and Tze-Yun Leong. 2023. Hierarchical Reinforcement Learning with Human-AI Collaborative Sub-Goals Optimization. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*. 2310–2312.

[28] Haozhe Ma, Thanh Vinh Vo, and Tze-Yun Leong. 2023. Human-AI Collaborative Sub-Goal Optimization in Hierarchical Reinforcement Learning. In *Proceedings of the AAAI Symposium Series*, Vol. 1. 86–89.

[29] Marlos C Machado, Marc G Bellemare, and Michael Bowling. 2017. A laplacian framework for option discovery in reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2295–2304.

[30] Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. 2018. Eigenoption Discovery through the Deep Successor Representation. In *International Conference on Learning Representations*.

[31] Sridhar Mahadevan and Mauro Maggioni. 2007. Proto-value Functions: A Laplacian Framework for Learning Representation and Control in Markov Decision Processes. *Journal of Machine Learning Research* 8, 10 (2007).

[32] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 1928–1937.

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[34] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-efficient hierarchical reinforcement learning. *Advances in Neural Information Processing Systems* 31 (2018).

[35] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. 2021. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–35.

[36] Rajesh Ranganath, Sean Gerrish, and David Blei. 2014. Black box variational inference. In *Artificial Intelligence and Statistics*. PMLR, 814–822.

[37] Matthew Riemer, Miao Liu, and Gerald Tesauro. 2018. Learning abstract options. *Advances in Neural Information Processing Systems* 31 (2018).

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[39] Özgür Şimşek and Andrew Barto. 2008. Skill characterization based on betweenness. *Advances in neural information processing systems* 21 (2008).

[40] Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. 2005. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proceedings of the 22nd international conference on Machine learning*. 816–823.

[41] Sainbayar Sukhbaatar, Emily Denton, Arthur Szlam, and Rob Fergus. 2018. Learning goal embeddings via self-play for hierarchical reinforcement learning. *arXiv preprint arXiv:1811.09083* (2018).

[42] Sainbayar Sukhbaatar, Zeming Lin, Ilya Kostrikov, Gabriel Synnaeve, Arthur Szlam, and Rob Fergus. 2018. Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play. In *International Conference on Learning Representations*.

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

[44] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

[45] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

[46] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. 2017. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*. PMLR, 3540–3549.

[47] Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. 2018. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 8 (2018), 2008–2026.

[48] Tianren Zhang, Shangqi Guo, Tian Tan, Xiaolin Hu, and Feng Chen. 2020. Generating adjacency-constrained subgoals in hierarchical reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 21579–21590.