Iterative Hierarchical Optimization for Misspecified Problems

Daniel J. Mankowitz ¹
Timothy A. Mann ²
Shie Mannor ¹

Abstract
For complex, high-dimensional Markov Decision Processes (MDPs), it may be necessary to represent the policy with function approximation. A problem is misspecified whenever, the representation cannot express any policy with acceptable performance. We introduce IHOMP: an approach for solving misspecified problems. IHOMP iteratively learns a set of context specialized options and combines these options to solve an otherwise misspecified problem. Our main contribution is proving that IHOMP enjoys theoretical convergence guarantees. In addition, we extend IHOMP to exploit Option Interruption (OI) enabling it to decide where the learned options can be reused. Our experiments demonstrate that IHOMP can find near-optimal solutions to otherwise misspecified problems and that OI can further improve the solutions.

1. Introduction
Reinforcement Learning (RL) algorithms can learn near-optimal solutions to well-defined problems. However, real-world problems rarely come in the form of a concrete problem description. A human has to translate the poorly defined target problem into a concrete problem description. A Misspecified Problem (MP) occurs when an optimal solution to the problem description is inadequate in the target problem. Unfortunately, creating a well-defined problem description is a challenging art. Furthermore, MPs can have serious consequences in many domains ranging from smart-grids (Abiri-Jahromi et al., 2013; Wu et al., 2010) and robotics (Smart and Kaelbling, 2002) to inventory management systems (Mann and Mannor, 2014). In this paper, we introduce a hierarchical approach that mitigates the consequences of problem misspecification. Why are problems misspecified? While a problem description can be misspecified for many reasons, one important case is due to the state representation. It is well established in the machine learning (Levi and Weiss, 2004; Zhou et al., 2009) and RL (Konidaris et al., 2011) literature that “good” features can have a dramatic impact on performance. Finding “good” features to represent the state is a challenging domain specific problem that is generally considered outside of the scope of RL. Unfortunately, domain experts may not supply useful features either because they do not fully understand the target problem or the technicalities of reinforcement learning. We may also prefer a MP with a limited state representation for several reasons: (1) Regularization (Singh et al., 1995; Geramifard et al., 2012). (2) due to Memory and System constraints (Roy and How, 2013; Singh et al., 1995). (3) Learning on Large Data: After learning on large amounts of data, augmenting a feature set with new features to get improved...
performance is non-trivial and often inefficient (Geramifard et al., 2012). **How can we mitigate misspecification?** Learning a hierarchical policy can mitigate the problems associated with a MP. To illustrate how learning a hierarchical policy can repair MPs, consider the S-shaped domain shown in Figure 1a. To solve the task the agent must move from the bottom left corner to the goal region denoted by the letter ‘G’ in the top right. The state representation only permits policies that move in a straight line. So the problem is misspecified, and it is not solvable with a flat policy approach (Figure 1a.i). However, if we break up the state-space, as shown in Figure 1a.ii, and learn one policy for each cell, the problem is solvable. The partial policies shown in Figure 1a.ii are an example of abstract actions, called options (Sutton et al., 1999), macro-actions (Hauskrecht et al., 1998; He et al., 2011), or skills (Konidaris and Barto, 2009). Previous approaches have proposed algorithms for learning options to learn or plan faster (McGovern and Barto, 2001; Moerman, 2009; Konidaris and Barto, 2009). In contrast, our objective is to learn options to repair a MP.

**Contributions:** Our main contributions are: (1) Introducing Iterative Hierarchical Optimization for Misspecified Problems (IHOMP), which learns options to repair and solve MPs. (2) Theorem 1 shows that IHOMP converges to a near-optimal solution relating the quality of the learned policy to the quality of the options learned by the “black box” RL algorithm. (3) Theorem 2 proves that Regularized Option Interruption (ROI) can be safely incorporated into IHOMP. (4) Experiments demonstrating that, given a misspecified problem, IHOMP can learn options to repair and solve the problem. Experiments showing IHOMP-ROI learning partitions and discovering reusable options. This divide-and-conquer approach may also enable us to scale and solve larger MDPs.

**Background:** Let $M = \langle S, A, P, R, \gamma \rangle$ be an MDP, where $S$ is a (possibly infinite) set of states, $A$ is a finite set of actions, $P$ is a mapping from state-action pairs to probability distributions over next states, $R$ maps each state-action pair to a reward in $[0, 1]$, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi(a|s)$ gives the probability of executing action $a \in A$ from state $s \in S$. Let $M$ be an MDP. The value function of a policy $\pi$ with respect to a state $s \in S$ is $V_M^\pi(s) = \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R(s_t, a_t) | s_0 = s \right]$ where the expectation is taken with respect to the trajectory produced by following policy $\pi$. The partial policies shown in Figure 1a.ii are an example of abstract actions, called options (Sutton et al., 1999), macro-actions (Hauskrecht et al., 1998; He et al., 2011), or skills (Konidaris and Barto, 2009). Previous approaches have proposed algorithms for learning options to learn or plan faster (McGovern and Barto, 2001; Moerman, 2009; Konidaris and Barto, 2009). In contrast, our objective is to learn options to repair a MP.

![Figure 1: (a) An episodic MDP with S-shaped state-space and goal region G. (i) Flat approach: A misspecified single policy failing to solve the task (ii) Hierarchical approach: Using and combining simple policy representations to solve a task. (b) Learning options (see Learning Options Section)](image-url)
action $a \in A$, and the optimal action-value function is denoted by $Q^*_M(s,a)$. Throughout this paper, we will drop the dependence on $M$ when it is clear from the context.

**Learning Options:** We define an option $\sigma$ by a tuple $\sigma = \langle \pi_\theta, \beta \rangle$, where $\pi_\theta$ is a parametric policy with parameter vector $\theta$ and $\beta : S \rightarrow \{0,1\}$ indicates whether the option has finished ($\beta(s) = 1$) or not ($\beta(s) = 0$) given the current state $s \in S$. Given a set of options $\Sigma$ with size $m \geq 1$, the inter-option policy is defined by $\mu : S \rightarrow [m]$ where $S$ is the state-space and $[m]$ is the index set over the options in $\Sigma$. An inter-option policy selects which options to execute from the current state by returning the index of one of the options. By defining inter-option policies to select an index (rather than the options), we can use the same policy even as the set of options is adapting. Figure 1b shows an arbitrary partitioning $\mathcal{P}$, consisting of 5 sub-partitions $\{\mathcal{P}_i | i = 1 \cdots 5\}$, defined over the original MDP’s state space. Each $\mathcal{P}_i$ is initialized with an arbitrary option and its corresponding Local-MDP $M'_i$. Local-MDP $M'_i$ is an episodic MDP that terminates once the agent escapes from $\mathcal{P}_i$ and upon terminating receives a reward equal to the value of the state the agent would have transitioned to in the original MDP. Therefore, we construct a modified MDP called a Local-MDP and apply a planning or RL algorithm to solve it. The resulting solution (policy) is a specialized option. In the next section, we introduce an algorithm for dynamically learning and improving options using iterative hierarchical optimization.

**IHOMP:** IHOMP (Algorithm 1) takes the original MDP $M$, a partition $\mathcal{P}$ over the state-space and a number of iterations $K \geq 1$ and returns a pair $\langle \mu, \Sigma \rangle$ containing an inter-option policy $\mu$ and a set of options $\Sigma$. The number of options $m = |\mathcal{P}|$ is equal to the number of classes (sub-partitions) in the partition $\mathcal{P}$ (line 1). The inter-option policy $\mu$ returned by IHOMP is defined (line 1) by $\mu(s) = \arg \max_{i \in [m]} \mathbb{I}\{s \in P_i\}$, where $\mathbb{I}\{\cdot\}$ is the indicator function returning 1 if its argument is true and 0 otherwise and $P_i$ denotes the $i^{th}$ class in the partition $\mathcal{P}$. Thus $\mu$ simply returns the index of the option associated with the partition class containing the current state. On line 1, IHOMP initializes $\Sigma$ with arbitrary options (IHOMP can also be initialized with options that we believe might be useful to speed up learning). Next (lines 1–1), IHOMP performs $K$ iterations. In each iteration, IHOMP updates the options in $\Sigma$ (lines 1–1). Note that the value of a option depends on how it is combined with other options. If we allowed all options to change simultaneously, the options could not reliably propagate value off of each other. Therefore, IHOMP updates each option individually. Multiple iterations are needed so that the option set can converge (Figure 1b). The process of updating an option (lines 1–1) starts by evaluating $\mu$ with the current option-set $\Sigma$ (line 1). Any number of policy evaluation algorithms could be used here, such as TD($\lambda$) with function approximation (Sutton and Barto, 1998) or LSTD (Boyan, 2002), modified to be used with options. Then we use the original MDP $M$ to construct a Local-MDP $M'$ (line 1). Next, IHOMP uses a planning or RL algorithm to approximately solve the Local-MDP $M'$ returning a parametrized policy $\pi_\theta$ (line 1). Any planning or RL algorithm for regular MDPs could fill this role provided that it produces a parametrized policy. However, in our experiments, we used a simple actor-critic PG algorithm, unless otherwise stated. Then a new option $\sigma' = (\pi_\theta, \beta'_i)$ is created (line 1) where $\pi_\theta$ is the policy derived on line 1 and $\beta'_i(s) = \begin{cases} 0 & \text{if } s \in P_i \\ 1 & \text{otherwise} \end{cases}$. The definition of $\beta'_i$ means that the option will terminate only if it leaves the $i^{th}$ partition. Finally, we update the option set $\Sigma$ by
replacing the $i^{th}$ option with $\sigma_i$ (line 1). It is important to note that in IHOMP, updating an option is equivalent to solving a Local-MDP.

**Algorithm 1 IHOMP**

**Require:** $M \{\text{MDP}\}$, $P \{\text{Partitioning of } S\}$, $K \{\text{Iterations}\}$

1. $m \leftarrow |P| \{\# \text{ of partitions.}\}$
2. $\mu(s) = \text{arg}\max_{i \in [m]} \{s \in P_i\}$
3. Initialize $\Sigma$ with $m$ options. \{1 option per partition.\}
4. for $k = 1, 2, \ldots, K$ do \{Do $K$ iterations.\}
5. for $i = 1, 2, \ldots, m$ do \{One update per option.\}
6. **Policy Evaluation:**
7. Evaluate $\mu$ with $\Sigma$ to obtain $V_M^{(\mu, \Sigma)}$
8. **Option Update:**
9. Construct Local-MDP $M'_i$ from $M$ & $V_M^{(\mu, \Sigma)}$;
10. Solve $M'_i$ obtaining policy $\pi_\theta$
11. $\sigma_i \leftarrow \langle \pi_\theta, \beta_i \rangle$
12. Replace $\sigma_i$ in $\Sigma$ by $\sigma'_i$
13. end for
14. end for
15. return $\langle \mu, \Sigma \rangle$

**Analysis of IHOMP:** We provide the first convergence guarantee for combining hierarchically and iteratively learning options in a continuous state MDP using IHOMP (Lemma 1 and Lemma 2, proven in the supplementary material). We use this guarantee and Lemma 2 to prove Theorem 1. This theorem enables us to analyze the quality of the inter-option policy returned by IHOMP. An important parameter for determining the quality of a policy returned by IHOMP is the misspecification error defined below.

**Definition 1** Let $P$ be a partition over the target MDP’s state-space. The **mis-specification error** is $\eta_P = \max_{i \in [m]} \eta_i$ where $\eta_i$ is the smallest $\eta_i \geq 0$, such that $V_M^*(s) - V_M^{\pi_\theta}(s) \leq \eta_i$, for all $s \in P_i$ and $\pi_\theta$ is the policy returned by the option learning algorithm executed on $M'_i$.

The misspecification error quantifies the quality of the Local-MDP solutions returned by our option learning algorithm. If we used an exact solver to learn options, then $\eta_P = 0$. However, if we use an approximate solver, then $\eta_P$ will be non-zero and the quality will depend on the partition $P$. Generally, using finer grain partitions will decrease $\eta_P$. However, Theorem 1 reveals that adding too many options can also negatively impact the returned policy’s quality.

**Theorem 1** Let $\varepsilon > 0$. If we run IHOMP with partition $P$ for $K \geq \log_\gamma (\varepsilon(1 - \gamma))$ iterations, then the algorithm returns stitching policy $\varphi = \langle \mu, \Sigma \rangle$ such that $\|V_M^* - V_M^\varphi\|_\infty \leq \frac{m \eta_P}{(1 - \gamma)^2} + \varepsilon$ where $m$ is the number of partition classes in $P$.

The proof of Theorem 1 is divided into three parts. The main challenge is that updating one option can impact the value of other options. Our analysis starts by bounding the impact of updating one option. Note that $\Sigma$ represents a option set and $\Sigma_i$ represents a option set where we have updated the $i^{th}$ option (corresponding to the $i^{th}$ partition class $P_i$) in the set. In the first part, we show that the error between $V_M^\varphi$, the globally optimal value function, and $V_M^{(\mu, \Sigma_i)}$, is a contraction when $s \in P_i$ and is bound by $\|V_M^* - V_M^{(\mu, \Sigma)}\|_\infty + \frac{\eta_P}{1 - \gamma}$ otherwise (Lemma 1). In the second part, we apply an inductive argument to show that updating all $m$ options results in a $\gamma$ contraction over the entire state space (Lemma 2). In the third part, we apply this contraction recursively, which proves Theorem 1. Theorem 1 tells us that when the misspecification error is small, IHOMP returns a near-optimal inter-option policy.

**Learning Partitions via Regularized Option Interruption:** To relax the a-priori partitioning assumption, we incorporate Regularized Option Interruption (ROI) (Mankowitz
et al., 2014) into this work to enable IHOMP to automatically learn a near-optimal partition from an initially misspecified problem. IHOMP keeps track of the action value function \( Q^{\mu,\Sigma}(s, j) \) which represents the expected value of being in state \( s \in S \) and executing option \( j \), given the inter-option policy \( \mu \) and option set \( \Sigma \). ROI uses this estimate of the action-value function to enable the agent to choose when to switch options according to the following termination rule: 

\[
\beta_j(s, t) = \begin{cases} 
1 & \text{if } Q^{\mu,\Sigma}(s, j) < V^{\mu,\Sigma}(s) - \rho, \\
0 & \text{otherwise}
\end{cases}
\]

Here \( \beta_j(s, t) \) corresponds to the termination probability of the \( j \)th option partition and \( V^{\mu,\Sigma}(s) = \max_{i \in [m]} Q^{\mu,\Sigma}(s, i) \). This rule is illustrated in Figure 3(a). A user has designed a partition resulting in a MP (Figure 3a(i)) compared to the optimal partition for this domain (Figure 3a(ii)). IHOMP applies ROI to ‘modify’ the initial partition into the optimal one. By learning the optimal action-value function \( Q^{s,\mu,\Sigma}(s, j) \), IHOMP builds a near-optimal partition (Figure 3a(iii)) that is implicitly stored within this action-value function. That is, if the agent is executing an option in partition class 1, and the value of continuing with option 1, \( Q^{\mu,\Sigma}(s, 1) \), is less than \( V^{\mu,\Sigma}(s) - \rho \) for some regularization function \( \rho \) (see the \( x \) location in Figure 3a(iii)), then switch to the new option partition \( (\beta_j(s, t) = 1) \). Otherwise, continue executing the current option (see the \( y \) location in Figure 3a(iii)). IHOMP-ROI automatically learns an improved partition between iterations. We show that ROI can be safely incorporated into IHOMP in Theorem 2.

**Theorem 2 (IHOMP-ROI Approximate Convergence)** 

\[
\|V^*_M - V^\phi_M\|_\infty \leq \frac{\log p}{(1-\gamma)^2} + \varepsilon
\]

holds for IHOMP-ROI.

**Experiments and Results:** We performed experiments on two well-known RL benchmarks: Puddle World (PW) (Sutton, 1996) and the Pinball domain (Konidaris and Barto, 2009). We also created a domain which we call the Two Rooms domain to demonstrate how IHOMP-ROI can improve partitions. In each experiment, we defined a MP, where no flat policy is adequate, and in some of the tasks, cannot solve the task at all. These experiments simulate situations where the policy representation is constrained to avoid overfitting, manage system constraints, or coping with poorly designed features. In each case, IHOMP learns a significantly better policy compared to the non-hierarchical approach. In the Two Rooms domain, IHOMP-ROI improves the initial partition.

IHOMP is a meta-algorithm. We provide an algorithm for Policy Evaluation (PE) and Policy Learning (PL). For the PW domain, we used SMDP-LSTD (Sorg and Singh, 2010) for PE and a modified version of Regular-Gradient Actor-Critic (RG-AC) (Bhatnagar et al., 2009) for PL. In the Pinball domain, we used Nearest-Neighbor Function Approximation (NN-FA) for PE and UCB Random Policy Search (UCB-RPS) for PL. In the two rooms domain, we use a variation of LSTDQ with Option Interruption for PE and RG-AC for PL. For the PW and Two Rooms domains, each intra-option policy is represented as a probability distribution over actions (independent of the state). We compare their performance to the original misspecified problem using a flat policy with the same representation.

**Puddle World:** Puddle World is a continuous 2-dimensional world containing two puddles as shown in Figure 2a(i). A successful agent (red ball) should navigate to the goal location (blue square), avoiding the puddles. The state space is the \( (x, y) \) location of the agent. Initially, the agent is provided with a misspecified problem. That is, a flat policy that can only move in a single direction (thus it cannot avoid the puddles). IHOMP (with a \( 2 \times 2 \) grid partition) overcomes this misspecification and attains near optimal average reward
compared to the flat policy (Figure 2a(ii)). Figure 2a(iii) compares the performance of different partitions where a 1 × 1 grid represents the flat policy of the initially misspecified problem. The option learning error $\eta_P$ is significantly smaller for all the partitions greater than 1 × 1, resulting in lower cost. On the other hand, according to Theorem 1, adding more options $m$ increases the cost. A trade off therefore exists between $\eta_P$ and $m$. In practice, $\eta_P$ tends to dominate $m$.

**Pinball:** We tested IHOMP on the challenging pinball-world task (Figure 2b(i)) Konidaris and Barto (2009). The agent is initially provided with a 5-feature flat policy $\langle 1, x, y, \dot{x}, \dot{y} \rangle$. This results in a misspecified problem as the agent is unable to solve the task using this limited representation as shown by the average reward in Figure 2b(ii). Using IHOMP with a 4 × 3 × 1 × 1 grid, 12 options were learned. IHOMP clearly outperforms the flat policy as shown in Figure 2b(ii). It is less than optimal but still manages to sufficiently perform the task (see value function, Figure 2b(iii)). The drop in performance is due to a complicated obstacle setup, non-linear dynamics and partition design. Nevertheless, this shows that IHOMP can produce a reasonable solution with a limited representation.

**Improving Partitions:** We tested IHOMP-ROI on the two rooms domain shown in Figure 3b(i). The agent (red ball) needs to navigate to the goal region (blue square). The policy parameterization is limited to a distribution over actions (moving in a single direction). This limited representation results in a MP as the agent is unable to traverse between the two rooms. If we use IHOMP with a sub-optimal partitioning containing two options as shown by the red and green cells in Figure 3b(ii), the problem is still misspecified. Here, the agent leaves the red cell and immediately gets trapped behind the wall whilst in the green cell. Using IHOMP-ROI, as shown in Figure 3b(iii), the agent learns both the options and a partition such that the agent can navigate to the goal. The green region in the bottom left corner comes about as a function approximation error but does not prevent the agent from reaching the goal. The agent also learns to reuse the red option’s dominant right action (Figure 3b(iii), Region B) to reach the goal.

2. Discussion

We introduced IHOMP, a RL planning algorithm for iteratively learning options and an inter-option policy (Sutton et al., 1999) to repair a MP. We provide theoretical results for IHOMP that directly relate the quality of the final inter-option policy to the misspecification error. IHOMP is the first algorithm that provides theoretical convergence guarantees while iteratively learning a set of options in a continuous state space. In addition, we have
developed IHOMP-ROI which makes use of regularized option interruption (Sutton et al., 1999; Mankowitz et al., 2014) to learn an improved partition to solve an initially misspecified problem. IHOMP-ROI is also able to discover regions in the state space where the options should be reused. In high-dimensional domains, partitions can be learned from expert demonstrations (Abbeel and Ng, 2005) and intra-option policies can be represented as Deep Q-Networks (Mnih, 2015). Option reuse can be especially useful for transfer learning (Tessler et al., 2016) and multi-agent settings (Garant et al., 2015).

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Union’s Seventh Framework Program (FP/2007-2013) / ERC Grant Agreement n. 306638.

References


Emma Brunskill and Lihong Li. PAC-inspired option discovery in lifelong reinforcement learning. JMLR, 2014.


