Time-Regularized Interrupting Options

Daniel J. Mankowitz DANIELM@TX.TECHNION.AC.IL
Timothy A. Mann MANN@EE.TECHNION.AC.IL
Shie Mannor SHIE@EE.TECHNION.AC.IL
Electrical Engineering Department, The Technion - Israel Institute of Technology, Haifa 32000, Israel

Abstract

High-level skills relieve planning algorithms from low-level details. But when the skills are poorly designed for the domain, the resulting plan may be severely suboptimal. Sutton et al. (1999) made an important step towards resolving this problem by introducing a rule that automatically improves a set of skills called options. This rule terminates an option early whenever switching to another option gives a higher value than continuing with the current option. However, they only analyzed the case where the improvement rule is applied once. We show conditions where this rule converges to the optimal set of options. A new interrupting Bellman operator that simultaneously improves the set of options is at the core of our analysis. One problem with the update rule is that it tends to favor lower-level skills. We introduce a regularization term that favors longer duration skills. Experimental results demonstrate that this approach can derive a good set of high-level skills even when the original set of skills cannot solve the problem.

1. Introduction

Options are control structures that can implement both high-level skills that accomplish a subgoal as well as primitive actions that execute for a single timestep (Sutton et al., 1999). Because of their flexibility options are often better suited for modeling complex problems than primitive actions (Stone et al., 2005; Konidaris et al., 2012). In addition, options have been shown experimentally (Precup & Sutton, 1997; Sutton et al., 1999; Silver & Ciosek, 2012) and theoretically (Mann & Mannor, 2014) to speed up the convergence rates of planning algorithms.

Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

In planning problems, high-level skills are given as a set of options, and a planner determines how these options can be put together to form a solution. One disadvantage of options is that they are opaque and indivisible. If a badly designed set of options are provided to a planner for solving a task, the planner may not be able to compose the options to solve the task. Thus, for a fixed set of options, the best solution may be unsatisfactory.

A Semi-Markov Decision Process (SMDP) model is defined with respect to a fixed set of options. Modifying an option results in a new option and therefore a new SMDP model. The only way to get a better solution is to change the options themselves and solve the new SMDP model instead. This is termed *model improvement*. Performing model improvement multiple times is termed *model iteration*. Each subsequent SMDP model can solve a task with greater efficiency until convergence.

Consider a city transit planner who has to design bus stops such that passengers can reach popular or important destinations in a city. It may be the case that a bus line (modeled as an option) has been previously constructed whose end terminal bypasses newly developed destinations of interest, as shown in Figure 1a. In this case, it is desirable to construct a new bus stop (modify the option) such that the end terminal is in a more desirable location in the city. A passenger may need to take more than one bus line (multiple options) to reach a popular destination as seen in Figure 1b. However, since none of the bus terminals are located at this destination, the city planner needs to modify the location of each of these terminals. Here, at least two iterations of model iteration are required to find the optimal locations of the new terminals.

There are two approaches to find new options: (1) discover new options from scratch, or (2) try to improve on the existing options. Option discovery is the process of learning and solving a set of sub-goals for a domain. A solution to a subgoal produces an option. This approach has been extensively studied (da Silva et al., 2012; Menache et al., 2002;

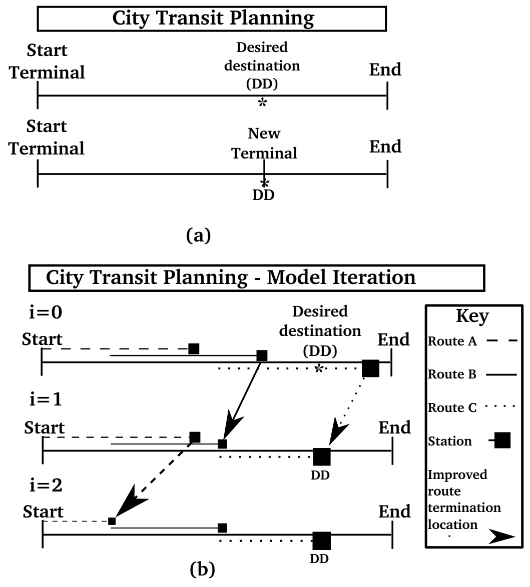


Figure 1. A city transit planner. (a) A bus line whereby the destination of interest is not found at the bus terminal. A new terminal is placed at the desired location. (b) A city transit planner in which option interruption is necessary for more than one iteration to address the model misspecification at iteration $i = 0$.

McGovern & Barto, 2001), but these techniques are often computationally expensive and lack formal guarantees.

Alternatively, we can try to improve an existing set of options. In many real-world problems, options may already be given by a domain expert. It makes sense to improve upon these options rather than discover new options from scratch. Sutton et al. (1999) introduced option interruption as a mechanism for improving a set of options by tuning their termination rules. Option interruption opportunistically looks for situations where switching to another option early produces a better solution (Figure 1b). Sutton et al. (1999) showed that their rule can improve a set of options (model improvement). However, they only consider improving the set of options for a single step of model improvement. Comanici & Precup (2010) developed an algorithm that learns locally-optimal termination conditions for options using a gradient-based algorithm. However, this method converges to a locally optimal solution and requires augmentation of the state space. Our proposed method extends the mechanism introduced by Sutton et al. (1999) by proving conditions where iteratively improving the options converges to the globally optimal set of options.

However, modifying the termination conditions of options tend to make options' durations shorter and shorter. This amounts to breaking the original high-level skills down into lower and lower-level skills. Technically, this is the optimal strategy because low-level skills can exactly represent the optimal solution. However, planning with temporally extended options can lead to significantly faster convergence (Mann & Mannor, 2014; Precup & Sutton, 1997; Sutton et al., 1999; Silver & Ciosek, 2012).

The main technical contributions of this paper are three-fold. First, we develop the Interrupting Option Value Iteration (IOVI) algorithm that simultaneously plans and improves the original option set. IOVI is the first known algorithm that incorporates model iteration with Value Iteration. Second, we prove that by applying option interruption iteratively to an initial set of options, using a new interrupting Bellman operator, IOVI algorithm converges to a global optimum ((Sutton et al., 1999) only analyzes a single iteration of option interruption). Furthermore, we add a time-based regularization to IOVI to form the Time Regularized IOVI (TRIOVI) algorithm. Time-based regularization has been incorporated to help prevent breaking the original high-level skills down to options with short durations. We show that the option set produced by TRIOVI converges to a local optimum with respect to the selected regularization function. Adding a regularization term has not been considered by (Sutton et al., 1999) or (Comanici & Precup, 2010).

A lack of domain knowledge can result in incorrectly modeling a problem domain. This is termed model misspecification (Joseph et al., 2013). In complex problems, it is not always clear whether good options have been provided by an expert to solve a task (Stolle & Precup, 2002). Planning with poorly designed options may produce a sub-optimal solution or no solution at all. As we demonstrate in our experiments, our proposed algorithm can derive a high-quality solution to a problem even when the original set of options cannot solve the task.

2. Background

An option is a temporally extended control structure defined by a triple $\langle I, \pi, \beta \rangle$ where I is the set of states where the option can be initiated, π is the option's policy, which determines how the option behaves in encountered states, and β is the set of termination probabilities determining when an option will stop executing. β is typically either a function of state s or time t. In this paper, we treat β as a function of both state and time; that is $\beta(s, t)$. Throughout this paper, we assume that we are given an initial set of $m \geq 1$ options $\mathcal{O}_0$ where each option $o_j = \langle I_j, \pi_j, \beta_j \rangle$ for $j = 1, 2, \ldots, m$. For convenience we will denote $\{1, 2, \ldots, m\}$ by $[m]$. Model iteration corresponds to modifying the termination probabilities β_j for $j \in [m]$, but leaves I_j and π_j unchanged. Since the options are indexed, it will often be convenient to abuse notation and use options interchangeably with their indices.

A Semi-Markov Decision Process (SMDP) can be defined by a five-tuple $\langle S, \mathcal{O}, P, R, \gamma \rangle$ where S is a set of states, $\mathcal{O}$ is a set of options, and P is the transition probability kernel. We assume rewards received at each timestep are in $[0, R_{\text{MAX}}]$ so that R is a mapping from $S \times \mathcal{O}$ to $[0, \frac{R_{\text{MAX}}}{1-\gamma}]$ representing the expected discounted sum of rewards received during the execution of an option o initialized from a state s, and $\gamma \in [0, 1)$ is the discount factor that causes an agent to place lower value on reward that takes more time to acquire. Model iteration constructs new SMDPs each time the option set is updated. To keep the notation consistent, we define $P^{\pi_j}(s'|s,t)$ to be the probability of reaching state s' t-timesteps after initializing the option o_j from state s and $R^{\pi_j}_{s,s',t}$ to be the expected discounted reward for encountering state s' exactly t timesteps after initializing option o_j in state s. Both of these definitions remain the same regardless of how we change the termination probabilities. Therefore, they are not affected by model iteration.

An option policy $\mu : S \to [m]$ is a mapping from states to indices over the space of options $\mathcal{O}$. The action-value function $Q^{\mu} : S \times [m] \to \mathbb{R}$ represents the long-term value of taking an option $o_j \in \mathcal{O}$ from a state $s \in S$ and thereafter always selecting options according to policy μ and is defined by $Q^{\mu}(s, o) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t | (s, o), \mu\right]$ where R_t is a random variable with support $[0, R_{max}]$ representing reward received at timestep t. It is well known that Q^{μ} can be written recursively as

$$Q^{\mu}(s, o_j) = \sum_{t=1}^{\infty} \sum_{s' \in S} \Phi^{o_j}_{s,s',t}(Q^{\mu}(s', \mu(s')), Q^{\mu}(s', o_j)) \ , \quad (1)$$

where

$$\Phi^{o_j}_{s,s',t}(v_{\text{term}}, v_{\text{cont}}) = P^{\pi_j}(s'|s,t)\Bigg(R^{\pi_j}_{s,s',t} + \gamma^t \left[\beta_{o_j}(s',t) v_{\text{term}} + (1 - \beta_{o_j}(s',t)) v_{\text{cont}}\right]\Bigg) \ . \quad (2)$$

The first argument v_{term} represents the expected cumulative reward received if o_j terminates in s' exactly t timesteps after being initiated in s, while the second argument v_{cont} represents the expected cumulative reward received if the agent continues following o_j from s'. Let $\Pi_{\mathcal{O}}$ be the set of all option policies defined over the option set $\mathcal{O}$. The optimal action-value function $Q^* = \max_{\mu \in \Pi_{\mathcal{O}}} Q^{\mu}$. Likewise the state-value function (or simply value function) $V^{\mu}(s) = Q^{\mu}(s, \mu(s))$ and the optimal value function $V^*(s) = \max_{o \in \mathcal{O}} Q^*(s, o)$.

The Bellman optimality operator $\mathcal{T}_{\mathcal{O}}$ for an option set $\mathcal{O}$, and operating on an arbitrary $Q \in \mathbb{R}^{|S \times [m]|}$ is defined as

$$(\mathcal{T}_{\mathcal{O}} Q)(s, o_j) = \sum_{t=1}^{\infty} \sum_{s' \in S} \Phi^{o_j}_{s,s',t}(V(s), Q(s, o_j)) \ , \quad (3)$$

where $V(s) = \max_{k \in [m]} Q(s, o_k)$. The operator $\mathcal{T}_{\mathcal{O}}$ is monotone, a max-norm contraction with coefficient γ, and has a unique fixed point Q^* (Szepesvári, 2010). $\mathcal{T}_{\mathcal{O}}$ also defines the Value Iteration (VI) algorithm for planning in SMDPs.

3. The Algorithm: IOVI

Interruption Options Value Iteration (IOVI, Algorithm 1) interlaces model iteration with VI. In other words, it plans and improves the option set simultaneously. This algorithm takes an arbitrary (nonempty) initial set of options $\mathcal{O}_0$ as well as θ, a threshold determining when the algorithm has converged. This algorithm iteratively improves the options during Value Iteration until the value function has converged. This improvement step will be referred to as the option update step. The option update parameter l denotes the number of iterations of VI to be performed before performing an option update step. The option update step is always performed on the original set of options $\mathcal{O}_0$ where $\beta_{0,j}(s,t)$ denotes the termination probability of the j^{th} option in $\mathcal{O}_0$. This step produces a new set of options $\mathcal{O}_i$ where i refers to the iteration. The options are improved by adjusting the respective termination probabilities $\beta_{0,j}(s,t)$ of the original set of options $\mathcal{O}_0$ at each iteration i. The termination probabilities of the options $\mathcal{O}_i$, where i is the iteration, are calculated based on the rule

$$\beta_{i,j}(s,t) = \max\Bigg(\beta_{0,j}(s,t), \mathbb{I}\left\{Q_i(s,o_j) < \max_{k \in [m]} Q_i(s, o_k)\right\}\Bigg) \ , \quad (4)$$

where $\mathbb{I}\{\cdot\}$ is the indicator function assigning 1 when its argument evaluates to true and 0 otherwise. Here, $\beta_{j,i+1}(s,t)$ is the termination probability of the j^{th} option at iteration $i+1$, $\beta_{j,0}(s,t)$ is the termination probability of the j^{th} option in the original option set and $Q_i(s,o)$ is the action-value estimate at the i^{th} iteration. Thus, given the original set of options $\mathcal{O}_0$ and $Q_i \in \mathbb{R}^{|S \times [m]|}$, an updated set of options is given by

$$U(\mathcal{O}_0, Q_i) = \{o_j \in \mathcal{O}_0 \mid \langle I_j, \pi_j, \beta_{i,j} \rangle\} \quad (5)$$

where $\beta_{i,j}$ is defined by (4). We update based on the original options $\mathcal{O}_0$ instead of the previous option set $\mathcal{O}_{i-1}$, which prevents model iteration from getting stuck in local optimum.

It is important to note that Algorithm 1 turns out to be equivalent to iteratively operating the Interrupting Bellman (IB) operator $\mathcal{G}$ on the action-value function Q. The IB operator is introduced in Section 4.2. The main additional computational cost over traditional VI is in line 3 which involves simulating the options whilst executing the interruption rule.

In the sections to follow, we present convergence guarantees for this algorithm in both the IO framework (Section 4) and the TRI framework (Section 5).

Algorithm 1 Interrupting Option Value Iteration

```
Require:
    • $\mathcal{O}_0$ : an initial set of options
    • $\theta > 0$ : error threshold
    • $l$ : frequency of option updates
1: $Q_0 \leftarrow 0; i \leftarrow 0$
2: repeat
3:     $Q_{i+1} \leftarrow (\mathcal{T}_{\mathcal{O}_i})^l Q_i$ {$l$ iterations of VI}
4:     $\mathcal{O}_{i+1} \leftarrow U(\mathcal{O}_0, Q_i)$ according to (5)
5:     $i \leftarrow i+1$
6: until $\theta \geq \|Q_i - Q_{i-1}\|_\infty$
7: return: $(Q_i, \mathcal{O}_i)$ { Action-values & options}
```

4. Interrupting Options (IO) Framework

Given an initial set of options $\mathcal{O}_0$ it is often possible to improve upon $\mathcal{O}_0$ by terminating some or all of the options prematurely, as mentioned in Section 1.

4.1. The IO Rule

Sutton et al. (1999) introduced a simple but effective method for option interruption that terminates an option o prematurely, if during the course of o's execution it encounters a state such that

$$Q(s,o) < V(s) \ . \tag{6}$$

where $V(s) = \max_{o \in \mathcal{O}} Q(s,o)$. An *interrupting option* $\widehat{o} = \langle I, \pi_o, \beta' \rangle$ (Sutton et al., 1999), is the same as the standard option except that its termination condition $\beta'(s) = 1$ whenever $Q(s,o) < V(s)$. The state s where the option terminated is referred to as the *interrupting state*. Interrupting options create a new set of options and define a new SMDP. The new option set still has $[m]$ options. For a given policy over options $\mu : S \to [m]$, the interruption rule can be written as $Q^\mu(s,o) < V^\mu(s)$. Since the number of options do not change after applying the IO rule, μ can still refer to options by their indices. However, the behavior of μ can change due to calling options from the new option set.

Sutton et al. (1999) showed that by interrupting an initial set of options $\mathcal{O}_0$, following an option policy μ, according to (6), $V_1^\mu \geq V_0^\mu$ where V_0^μ is the value function corresponding to the initial option set $\mathcal{O}_0$, and V_1^μ is the value function corresponding to the interrupted option set $\mathcal{O}_1$. In addition, strict improvement, $V_1^\mu(s) > V_0^\mu(s)$, is possible if there is a non-zero probability of encountering an interrupting state s' from state s, after initiating an option according to a policy μ. However, this is limiting as it has only been shown to improve options for a single update. We have extended this to model iteration by modifying the option set iteratively in IOVI, using (4), and guaranteeing convergence of the algorithm.

4.2. The Interrupting Bellman Operator

To prove properties such as convergence for IOVI, we introduce a new operator called the Interrupting Bellman (IB) operator. The IB operator $\mathcal{G}$ is given in Definition 1.

Definition 1. *For any estimate of the value function $Q \in \mathbb{R}^{|S \times [m]|}$, the IB operator $\mathcal{G}$ is defined by*

$$(\mathcal{G}Q)(s,o_j) = \sum_{s' \in S} \sum_{t=1}^{\infty} \Phi^{o_j}_{s,s',t}(V(s), z(s',o,Q)) \ , \tag{7}$$

where

$$z(s',o_j,Q) = \begin{cases} V(s') & \text{if } Q(s',o_j) < V(s') \\ Q(s',o_j) & \text{otherwise} \end{cases} , \tag{8}$$

and $V(s) = \max_{j \in [m]} Q(s,o_j)$.

It turns out that $\mathcal{G}$ has a unique fixed point Q^*. The IB operator for the value function V is defined similarly. Starting with an arbitrary $Q \in \mathbb{R}^{|S \times [m]|}$ and option set $\mathcal{O}_0$, applying $\mathcal{G}$ to Q and its iterates $i \geq 1$ times explicitly produces new action-value function estimates $Q_1, Q_2, \ldots, Q_i$ and implicitly produces a sequence of option sets $\mathcal{O}_1 = U(\mathcal{O}_0, Q_1), \mathcal{O}_2 = U(\mathcal{O}_0, Q_2), \ldots, \mathcal{O}_i = U(\mathcal{O}_0, Q_i)$ according to (5). The importance of the operator $\mathcal{G}$ is that it always operates explicitly on the original option set $\mathcal{O}_0$, but $\mathcal{G}Q_i$ is equivalent to $\mathcal{T}_{\mathcal{O}_i} Q_i$ for the i^{th} iteration. Because of this $\mathcal{G}$ can be seen as simultaneously performing planning and model iteration.

4.3. Convergence of IOVI using the IO rule

By performing model iteration in an algorithm such as IOVI, we dynamically and iteratively interrupt and modify the current set of options to efficiently solve the task at hand. This technique improves the overall solution and IOVI converges to a unique fixed point Q^* as is stated in the theorem to follow. This fixed point corresponds to the optimal value function given the best possible set of options that can be derived by modifying the original option set's termination conditions according to (4).

Theorem 1. *Let $\mathcal{O}_0$ be an initial set of options. The IB operator $\mathcal{G}$ has a unique fixed point Q^* and the following relationship is satisfied,*

$$\|Q^* - \mathcal{G}Q\|_\infty \leq \gamma \|Q^* - Q\|_\infty \ . \tag{9}$$

Due to space limitations, we provide only a sketch of the proof here (the complete proof is in the supplementary material). Expanding the expression $\|Q^* - \mathcal{G}Q\|_\infty$ introduces the term $\mathcal{Z}_q \triangleq [z(s',o,Q^*) - z(s',o,Q)]$ which contains the interruption function from (8). The proof hinges on showing that $\mathcal{Z}_q \leq ||Q^* - Q||_\infty$. The definition of the $z(.)$ function naturally breaks down into two cases (see (8)). Since the expression $\mathcal{Z}_q$ contains two instances of the $z(.)$ function, we end up with four distinct cases. $\mathcal{Z}_q \leq \|Q^* - Q\|_\infty$ in each of the four possible cases which proves that the algorithm converges. Two of the four distinct cases are not possible and therefore only two cases are considered. The cases include (i) $z(s', o, Q^*) = Q^*(s, o)$ and $z(s', o, Q) = Q(s, o)$, (ii) $z(s', o, Q^*) = Q^*(s, o)$ and $z(s', o, Q) = V(s)$. Case (ii) has two distinct subcases, namely: $(ii.1)$ $Q^*(s, o) - V(s) > 0$ and $(ii.2)$ $V(s) - Q^*(s, o) > 0$.

Here γ, the discount factor, is the contraction coefficient determining the convergence rate. The above theorem states that for an arbitrary initial action-value function Q, convergence to the optimal interrupting action-value function Q^*, the unique fixed point of $\mathcal{G}$, is guaranteed. Its convergence, in the worst-case is approximately at the same rate as VI. Convergence of IOVI is also ensured.

Corollary 1. *Let $l \geq 1$ and $\mathcal{O}_0$ be an initial set of options. If IOVI is executed with parameters $\mathcal{O}_0$, $\theta = 0$ and l, then it converges (in the limit) to Q^*.*

This result confirms convergence of IOVI which iteratively modifies options using (6). The algorithm converges to a globally optimal solution Q^*. This framework does not however, preserve the duration of the options. As mentioned by (Mann & Mannor, 2014), preserving the duration of options is a desirable property as it increases the rate of convergence of approximate dynamic programming algorithms. Therefore, we extended the IO rule to include a time-based regularization term, which encourages solutions that preserve the duration of options. Adding a regularization term adds significant complexity to preserving the theoretical convergence guarantees presented in this paper. We will show that convergence to a locally optimal solution for a regularization term can be guaranteed.

5. Time-Regularized Interruption (TRI) Framework

The Time-Regularized Interruption (TRI) framework is an extension of the IO framework that introduces time-based regularization functions to preserve option duration while performing model iteration. The addition of regularization creates a variation of the IOVI algorithm, which will be referred to as Time Regularized IOVI or TRIOVI.

5.1. The TRI Rule

Unlike in the IO framework, in the TRI framework, we assume that each time TRIOVI constructs a new set of options, it solves for the fixed point of the optimum Bellman operator $\mathcal{T}_{\mathcal{O}_i}$ given the new set of options $\mathcal{O}_i$. The TRI rule behaves differently depending on the selected regularization function, and we define a flexible set of regularization functions that penalize models containing options with short durations.

Definition 2. *For all $t, t' \in \mathbb{N}$ such that $t < t'$ an admissible regularization function $\rho : \mathbb{N} \to [0, \infty)$ satisfies $\rho(t) \geq \rho(t')$, and we denote the set of all admissible regularization functions by Θ.*

Although many time dependent regularization functions are possible, one interesting example is the following:

$$\rho(t, \lambda) = \lambda^t \left(\frac{R_{\text{MAX}}}{1 - \gamma} \right) \ , \tag{10}$$

where $\lambda \in [0, 1]$ is the parameter controlling regularization and $t \geq 1$ denotes the number of timesteps that the current option has been executing. As λ is set closer to 0, less regularization occurs meaning that options may terminate earlier. On the other hand, as λ is set closer to 1, more regularization occurs preventing options from terminating quickly. It should be noted that time t is only incorporated into each option's termination condition and therefore does not increase the size of the state space.

Given a regularization function $\rho \in \Theta$, we can describe TRIOVI as a process $\mathcal{P}_\rho$ starting from an initial set of options $\mathcal{O}_0$ and an arbitrary initial action-value function $Q_0 \in \mathbb{R}^{|S \times [m]|}$. On the first round, TRIOVI uses VI to acquire the fixed point of $\mathcal{T}_{\mathcal{O}_0}$, denoted by $Q_0^* = Q_1$, and updates the option set by $\mathcal{O}_1 = U_\rho(\mathcal{O}_0, \mathcal{O}_0, Q_1)$ where

$$U_\rho(\mathcal{O}_0, \mathcal{O}_i, Q_i) = \{ j \in [m] \mid \langle I_j, \pi_j, \beta_{i,j} \rangle \} \tag{11}$$

and $\beta_{i,j}(s, t)$

$$= \max \Big(\beta_{0,j}(s, t), \mathbb{I}\left\{ Q_i(s, o_j) < V_i(s) - \alpha_i \rho(t) \right\} \Big) \tag{12}$$

for $\alpha_i = \begin{cases} 1 & \text{if } i = 1 \text{ or } \beta_{i-1,j}(s, t) \neq 1 \\ 0 & \text{otherwise} \end{cases}$. Then a new round begins. At the i^{th} round, TRIOVI uses VI to acquire the fixed point of $\mathcal{T}_{\mathcal{O}_{i-1}}$, denoted by $Q_{i-1}^* = Q_i$, and the option set $\mathcal{O}_i = U_\rho(\mathcal{O}_0, \mathcal{O}_{i-1}, Q_i)$ is obtained.

Notice that with $\rho(t) = 0$ for $t \geq 0$ the option update rule (11) reduces to the rule used in the IO framework. However, in this new rule, the termination probabilities (12) for $\mathcal{O}_i$ (and therefore the option update rule (11)) are defined based on termination probabilities of the previous option set $\mathcal{O}_{i-1}$. The intuition behind α_i is that it turns the penalty on and off based on the previous option set $\mathcal{O}_{i-1}$. Option interruption can be thought of as making an option's duration shorter. If the previous $o_j \in \mathcal{O}_{i-1}$ does not interrupt in state s' at time t ($\alpha_i = 1$), then interrupting at (s', t) is penalized because we only want options to become shorter if the gain in value is significant. On the other hand, if $o_j \in \mathcal{O}_{i-1}$ interrupts at (s', t) $(\alpha_i = 0)$, deciding not to interrupt at (s', t) corresponds to lengthening the option's duration. In this case, the regularization would encourage switching to longer options even though they are suboptimal. Without the α_i function, model iteration with a regularization function can chatter back and forth between two different option models and never converge. Thus, the dependence on the previous option set's termination probabilities seems necessary.

5.2. Convergence of TRIOVI using the TRI rule

We show that the TRIOVI algorithm converges to a locally optimal value function Q^*_ρ for a regularization function ρ. That is, in the limit, Q_ρ ceases to change.

Lemma 1. *Let $\rho \in \Theta$ and $\mathcal{P}_\rho$ be the process induced by TRIOVI with initial option set $\mathcal{O}_0$ and $Q_0 \in \mathbb{R}^{|S \times [m]|}$, then for all $i \geq 1$, $Q_i \leq Q_{i+1}$.*

Lemma 1 says that the next model produced by TRIOVI is at least as good as the previous model. We use the previous lemma to prove that TRIOVI with regularization converges to a local optimum.

Theorem 2. *Let $\rho \in \Theta$. The sequence of value functions $Q_0, Q_1, \ldots, Q_i$ produced by the process $\mathcal{P}_\rho$ (the TRIOVI algorithm) with initial option set $\mathcal{O}_0$ and $Q_0 \in \mathbb{R}^{|S \times [m]|}$ converges to a local optimum with probability 1.*

Theorem 2 says that TRIOVI converges to a local optimum with any admissible regularization function $\rho \in \Theta$ provided that it solves for the optimal action-value function between option updates. The fact that TRIOVI does not necessarily converge to the global optimum is due to using a non-zero regularization function and cannot be avoided. However, as we will see in our experiments, TRIOVI often converges to solutions that are close to optimal.

6. Experiments and Results

The following experiments in a transit planning domain and an inventory management domain demonstrate IOVI's and TRIOVI's ability to iteratively improve on an initial set of options. IOVI and TRIOVI are able to derive a solution even when the initial options cannot solve the task. For TRIOVI, we used (10) as the regularization function in all of our experiments. The resulting algorithm has two tunable parameters l and λ, where l controls the frequency at which the options are updated and $\lambda \in [0, 1]$ controls the time-based regularization. We experimented with $l = \{1, 10, 20, 30, 40\}$ and $\lambda = \{0, 0.1, 0.3, 0.5\}$, unless noted otherwise. Notice that when $\lambda = 0$, this is equivalent to $\rho(t) = 0$ for all $t \geq 1$, which corresponds to IOVI.

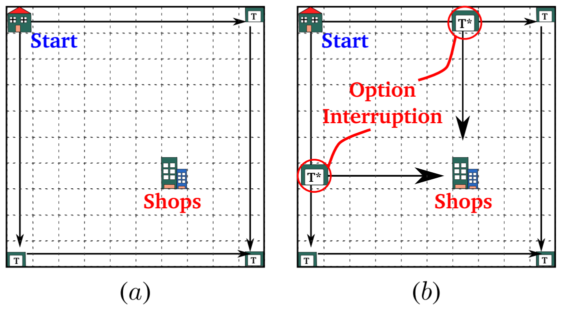


Figure 2. A transit planning system with (a) misspecified options so that there is no way to reach the goal state (shopping mall) from the start state (house) and (b) interrupted options that enable the agent(s) to successfully transition to the goal state from the start state. The interruption states are denoted T^* and are analogous to new bus terminals.

6.1. Misspecified Options in a Transit Planning System

We implemented a transit planning task as a gridworld (Sutton & Barto, 1998), where each cell represents a city block. The original option set contained four options transitioning (with zero probability of terminating) in the four directions: north, south, east, and west. Here options represent bus routes. The objective is to determine appropriate locations for bus stops (represented by option termination) so that residents can efficiently travel to popular destinations in the city. For example, suppose that residents want to take the bus to a new shopping center (Figure 2a). The existing bus routes may need to be modified. Both IOVI and TRIOVI can discover where to place new bus stops that allow residents to efficiently reach the new shopping center (Figure 2b). IOVI's solution may involve frequently interrupting options which results in adding a large number of bus stops. Real transit planning problems have budget constraints that prohibit adding an excessive number of bus stops, even if doing so is optimal. This highlights the need for TRIOVI as adding regularization causes TRIOVI to converge to options that terminate less frequently implying solutions with fewer bus stops. By tuning the regularization parameter λ, we can find a solution that provides efficient transportation within budget constraints.

Figure 3a shows that IOVI converges regardless of the frequency that we perform option updates (i.e., l). IOVI converges with the fewest iterations when $l = 1$, implying that the options are updated at every iteration. This is expected as the options are modified more frequently, resulting in a faster convergence rate to the optimal model.

For TRIOVI we fix $l = 40$ so that VI has time to converge between model improvements. We can analyze the convergence rate of TRIOVI for different values of the regularization parameter λ as seen in Figure 3b. Here, we can see that the algorithm converges regardless of the value of

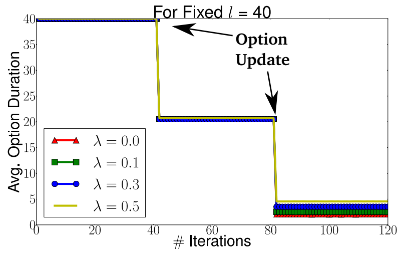
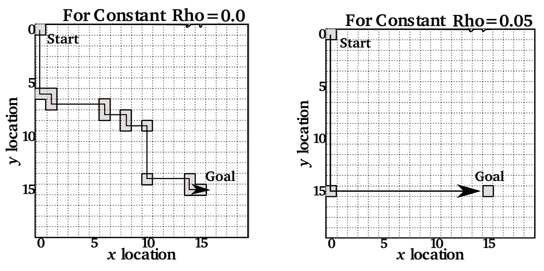


Figure 5. The transit planning system whereby a single option update is insufficient to converge to the optimal option durations



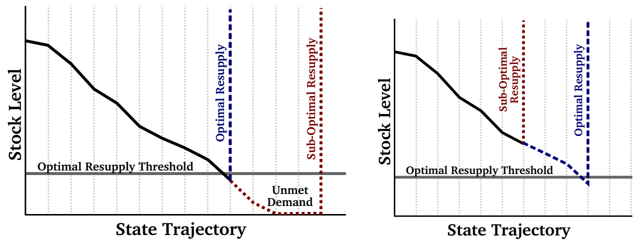
(*a*) (*b*)

Figure 6. A transit planning system with (*a*) Option interruption using no regularization. This does not preserve option duration and therefore results in premature terminations. (*b*) Option interruption using regularization which preserves option duration and results in a more direct and efficient planning solution.

λ. The larger λ values result in convergence in fewer iterations, because more regularization (i.e., larger values of λ) preserves option duration as seen in Figure 3*c*. This results in faster convergence to the optimal solution (Mann & Mannor, 2014; Precup & Sutton, 1997; Sutton et al., 1999; Silver & Ciosek, 2012).

Another interesting finding is displayed in Figure 5. Here, a sub-optimal option duration would have resulted if the options were only interrupted on a single occasion. This is indicated by the plateaus shown in the figure between $0-80$ iterations. This motivates the need to iteratively interrupt options by performing model iteration, see Section 1, such that the optimal option duration can be generated resulting in an optimal policy.

To emphasize the importance of regularization, we compared the bus stop locations derived by IOVI (Figure 6a) to the bus stop locations derived by TRIOVI with a constant penalty function $\rho(t) = 0.05$ for $t \geq 0$ (Figure 6b). Although both algorithms derive optimal solutions to travel from the initial state to the goal state, TRIOVI generates fewer bus stops (fewer option interruptions). This suggests that time-based regularization can play an important role in deriving simpler policies.



(*a*) (*b*)

Figure 7. Learning to optimally resupply inventory is a matter of discovering the optimal times to resupply. If we resupply after the stock level is too low (*a*), we may suffer high costs for unmet demand. On the other hand, if we resupply when the stock level is too high (*b*), then we will pay a high price per unit ordered.

6.2. Discovering when to Restock

An interesting application of IOVI and TRIOVI is determining when to restock inventory (Scarf, 1959). In this task, the agent manages stock for a single commodity in a finite warehouse. The options are simple: (a) resupply (fill the warehouse's remaining space) or (b) order nothing until another resupply is needed. The problem arises from the fact that it is often not clear how long to wait between resupply actions. When the agent makes an order it pays a cost for each ordered unit (-0.2 in our experiments) and a base ordering cost (-20 in our experiments). Thus large orders are effectively discounted, and the agent should only resupply when it can place a large order. On the other hand, not having enough stock to meet stochastic demands results in a high penalty (-100 base cost and -10 unmet demand cost per unit). Resupplying too early results in paying more for each unit, but waiting too long to resupply can result in high penalties for unmet demands (Figure 7). Given a resupply option and an option that orders nothing, both IOVI and TRIOVI can learn the optimal resupply times. In the initial option set $\mathcal{O}_0$, both options never terminate. These options are intentionally designed such that deriving a satisfactory policy in this domain is impossible.

Initially, we fixed $\lambda = 0$ and varied l. Figure 4*a* shows that IOVI converges regardless of our choice for l. As in the transit planning system, decreasing l, which improves the options more frequently, results in faster convergence. We then fixed $l = 40$ and varied the regularization parameter λ. As can be seen in Figure 4*b*, IOVI ($\lambda = 0$) converges to the optimal solution. TRIOVI converges for all λ values but increasing the regularization results in less optimal solutions. This is the price paid for using regularization to keep high-level skills from being broken down. Larger penalty terms reduce the optimality of the solution to preserve option duration. Figure 4*c* shows that IOVI converges to options with short durations as is expected whereas increasing λ causes TRIOVI to converge to option sets with

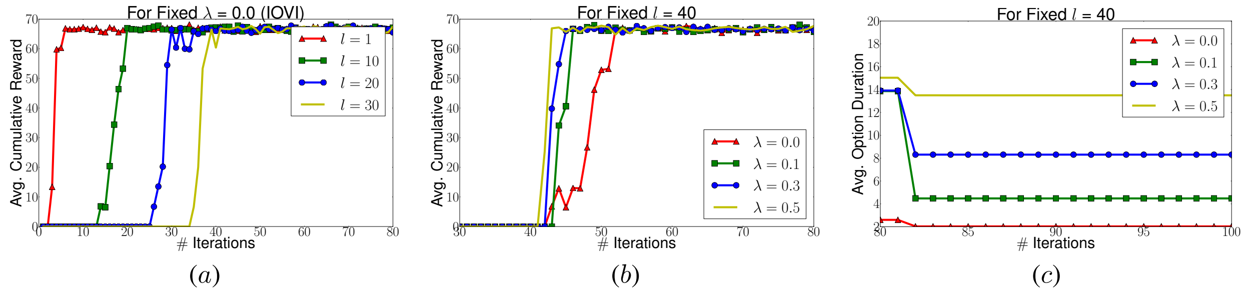


Figure 3. The transit planning system: (a) The cumulative reward for $l = \{1, 10, 20, 30\}$ when the regularization term $\lambda = 0$ which corresponds to IOVI. (b) The cumulative reward for a fixed option update $l = 40$ iterations. (c) The average option duration for $\lambda = \{0.0, 0.1, 0.3, 0.5\}$ when the option update step is performed every $l = 40$ iterations.

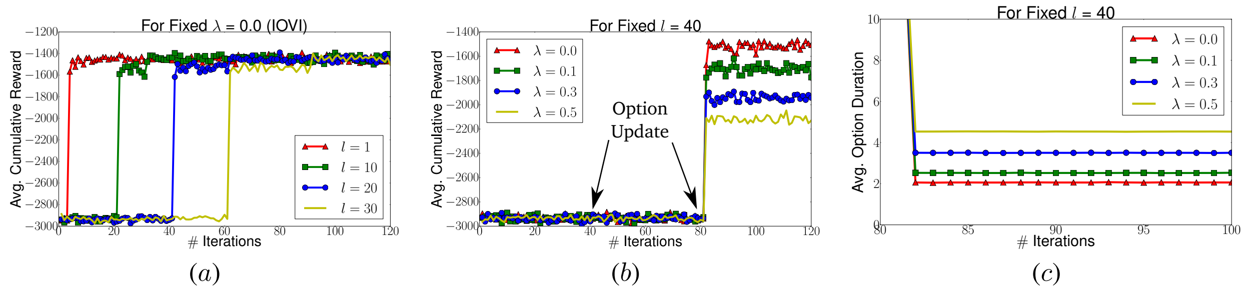


Figure 4. The inventory domain: (a) The cumulative reward for $l = \{1, 10, 20, 30\}$ when the regularization term $\lambda = 0$. (b) The cumulative reward for a fixed option update $l = 40$ iterations. (c) The average option duration for $\lambda = \{0.0, 0.1, 0.3, 0.5\}$ when the option update step is performed every $l = 40$ iterations.

longer durations. A trade-off is evident whereby IOVI converges to an optimal solution at the expense of shorter duration options and slower convergence rates. TRIOVI on the other hand converges to longer duration options and therefore obtains faster convergence rates, but at the expense of a less optimal solution.

7. Discussion

We have defined a dynamic Interrupting Bellman (IB) operator $\mathcal{G}$ which iteratively and implicitly modifies an option's termination conditions using the IO rule and explicitly updates the value function. It is equivalent to the Bellman operator $\mathcal{T}_{\mathcal{O}_i}$ for a single iteration i and ensures convergence to a globally optimum solution. We have demonstrated the interlacing of model iteration and VI in the IOVI algorithm for two different tasks. As our theoretical results predicted, this algorithm does indeed converge for the IO framework. When incorporating the time-based regularization, TRIOVI converges to a locally optimal fixed point Q^*_ρ for the optimal set of options derived from $\mathcal{O}_0$ with respect to ρ. This results in improved convergence rates (Mann & Mannor, 2014; Precup & Sutton, 1997; Sutton et al., 1999; Silver & Ciosek, 2012).

Based on the experimental results, it may be possible to prove that TRIOVI converges to a globally optimal fixed point Q^*_ρ. This fixed point would be for the optimal set of options derived from $\mathcal{O}_0$ with respect to ρ. An additional natural extension to this theory is extending it to function approximation. An addition to this work may include modifying an option's intra-option policy (Sutton et al., 1999). This may provide a more flexible and efficient solution to modifying and planning with misspecified options. This would result in simultaneously modifying the termination conditions as well as the intra-option policies whilst performing planning. Due to the dynamic and flexible nature of IOVI and TRIOVI, it may also be possible to extend this work to *transfer planning*. That is, utilizing the same set of options in a different domain that has not been previously seen to perform planning. Finally, it would be useful to develop a theoretical analysis for the convergence rate of the TRIOVI framework.

8. Acknowledgement

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

Comanici, Gheorghe and Precup, Doina. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the* 9^{th} *International Conference on Autonomous Agents and Multiagent Systems*, pp. 709–714, 2010.

da Silva, B.C., Konidaris, G.D., and Barto, A.G. Learning parameterized skills. In *Proceedings of the Twenty Ninth International Conference on Machine Learning*, June 2012.

Joseph, Joshua, Geramifard, Alborz, Roberts, John W, How, Jonathan P, and Roy, Nicholas. Reinforcement learning with misspecified model classes. In *Proc. of the IEEE International Conference on Robotics and Automation*, 2013.

Konidaris, George, Scheidwasser, Ilya, and Barto, Andrew. Transfer in reinforcement learning via shared features. *J. Mach. Learn. Res.*, 98888:1333–1371, June 2012.

Mann, Timothy A and Mannor, Shie. Scaling up approximate value iteration with options: Better policies with fewer iterations. In *Proceedings of the* 31^{st} *International Conference on Machine Learning*, 2014.

McGovern, Amy and Barto, Andrew G. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *Proceedings of the 18th International Conference on Machine Learning*, pp. 361 – 368, San Fransisco, USA, 2001.

Menache, Ishai, Mannor, Shie, and Shimkin, Nahum. Q-cut: dynamic discovery of sub-goals in reinforcement learning. In *Machine Learning: ECML 2002*, pp. 295–306. Springer, 2002.

Precup, Doina and Sutton, Richard S. Multi-time models for temporally abstract planning. In *Advances in Neural Information Processing Systems 10 (Proceedings of NIPS'97)*, 1997.

Scarf, Herbert. The optimality of (s,s) policies in the dynamic inventory problem. Technical Report NR-047-019, Office of Naval Research, April 1959.

Silver, David and Ciosek, Kamil. Compositional Planning Using Optimal Option Models. In *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, 2012.

Stolle, Martin and Precup, Doina. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation*, pp. 212–223. Springer, 2002.

Stone, Peter, Sutton, Richard S, and Kuhlmann, Gregory. Reinforcement learning for robocup soccer keepaway. *Adaptive Behavior*, 13(3):165–188, 2005.

Sutton, Richard and Barto, Andrew. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

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

Szepesvári, Csaba. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.