Adaptive Metareasoning for Bounded Rational Agents

Justin Svegliato and **Shlomo Zilberstein**
College of Information and Computer Sciences
University of Massachusetts Amherst
$\{$jsvegliato,shlomo$\}$@cs.umass.edu

Abstract

In computational approaches to bounded rationality, metareasoning enables intelligent agents to optimize their own decision-making process in order to produce effective action in a timely manner. While there have been substantial efforts to develop effective meta-level control for anytime algorithms, existing techniques rely on extensive offline work, imposing several critical assumptions that diminish their effectiveness and limit their practical utility in the real world. In order to eliminate these assumptions, *adaptive metareasoning* enables intelligent agents to *adapt* to each individual instance of the problem at hand without the need for significant offline preprocessing. Building on our recent work, we first introduce a model-free approach to meta-level control based on reinforcement learning. We then present a meta-level control technique that uses temporal difference learning. Finally, we show empirically that our approach is effective on a common benchmark in meta-level control.

1 Introduction

Due to the computational complexity of decision making in the real world, it has long been recognized that building intelligent agents that exhibit *perfect rationality* is beyond our reach [Simon, 1947; 1982]. There have therefore been substantial efforts to develop computational approaches to *bounded rationality* [Horvitz, 1990; Russell and Wefald, 1991b; Zilberstein, 1993]. By taking into account the computational costs of decision making, these models enable intelligent agents to compute satisficing actions—decisions deemed "good enough"—in complex situations with a limited amount of time available for deliberation. One particularly promising approach to bounded rationality has been based on *metareasoning*. Metareasoning gives intelligent agents the capacity to explicitly deliberate about and hence optimize their own decision-making process in order to produce effective action in a timely manner. As the autonomy and complexity of intelligent agents continues to grow, the need for metareasoning has become increasingly more important given the uncertainty about the range of potential situations and exceptions that they might encounter as well as the variability and limitations of their reasoning capabilities [Zilberstein, 2011].

In one of the most promising approaches to metareasoning, an intelligent agent monitors and controls *anytime algorithms*, which have been developed for a wide range of real-time planning and decision-making tasks, such as belief-space planning [Pineau *et al.*, 2003], probabilistic inference [Ramos and Cozman, 2005], heuristic search [Hansen *et al.*, 1997; Richter *et al.*, 2010], motion planning [Luna *et al.*, 2013], and object detection [Richtsfeld *et al.*, 2013]. Simply put, an anytime algorithm is an algorithm that gradually improves the quality of a solution as it runs and returns the current solution if it is interrupted. This behavior enables an anytime algorithm to trade computation time with solution quality. In intelligent agents, this has proven to be useful since they must often approximate solutions to complex problems in order to respond within an acceptable amount of time. However, to exploit the trade-off between solution quality and computation time, the agent must solve a non-trivial meta-level control problem: it must decide when to interrupt the anytime algorithm and act on the current solution.

There has been tremendous progress in the development of effective meta-level control for anytime algorithms. One common approach estimates the stopping point of the algorithm before it begins and lets it run until that stopping point has been reached [Horvitz, 1987; Boddy and Dean, 1994]. Since the stopping point is not revised after the algorithm begins, this approach is called *fixed allocation*. If there is little uncertainty about the performance of the algorithm or the urgency for the solution, fixed allocation interrupts the algorithm near the optimal stopping point. However, in real-time decision-making problems, there is often considerable uncertainty about either or both variables [Paul *et al.*, 1991]. Hence, another more sophisticated approach monitors the performance of the algorithm and estimates the stopping point at run time [Horvitz, 1990; Breese and Horvitz, 1991; Zilberstein and Russell, 1995; Hansen and Zilberstein, 2001]. Given that the stopping point is continually revised based on the performance of the algorithm, this approach is called *monitoring*, which is a form of introspective reasoning [Cox and Ram, 1999]. Monitoring has proven to be a better approach to meta-level control than fixed allocation because it more effectively handles variance and noise in the performance of the algorithm.

Existing meta-level control techniques that use either fixed allocation or monitoring have traditionally relied on significant offline work. In particular, before the activation of the anytime algorithm, a model that describes its performance must be compiled for the given problem [Zilberstein, 1996]. However, relying on such a model, called a *performance profile*, imposes several assumptions that are often not feasible in real world applications, particularly:

- *There is ample time for offline compilation of the performance profile of the anytime algorithm.*
- *The anytime algorithm performs uniformly across each individual problem instance that may be encountered.*
- *The distribution of problem instances that the anytime algorithm must solve in the future is known and fixed.*
- *The anytime algorithm always runs on the same machine under the same CPU and memory conditions.*

In short, existing techniques depend on many unrealistic assumptions that diminish their effectiveness and limit their practical utility in the real world.

In order to eliminate these assumptions, we propose an adaptive approach to metareasoning for bounded rational agents. Simply put, *adaptive metareasoning* aims to provide intelligent agents with two capabilities. The first capability enables the agent to adapt its own decision-making process to each individual instance of the problem at hand. More importantly, however, the second capability allows the agent to monitor and control its own decision-making process without the need for extensive offline work. In other words, the key idea is to replace substantial offline preprocessing with efficient online methods for predicting the performance of the anytime algorithm on each individual instance of the problem at hand. Both capabilities therefore provide bounded rational agents with significantly more effective metareasoning that can be far easier to implement and deploy in practice.

In recent work, we introduce one of the first adaptive metareasoning methods [Svegliato *et al.*, 2018]. In particular, we present a meta-level control technique that uses an online performance prediction framework. Similar to earlier work, our technique monitors the performance of the anytime algorithm and estimates the stopping point at run time. However, in place of a performance profile that must be compiled prior to the activation of the algorithm, our technique predicts future performance based on the current performance of the algorithm on each individual instance of the problem at hand. Even without significant offline work, our technique has been shown to be more effective than state-of-the-art techniques across a wide range of common benchmark domains.

While our meta-level control technique takes advantage of a *single problem instance* without any prior knowledge using online performance prediction, it is possible to exploit *multiple problem instances* using reinforcement learning. Building on our recent work in adaptive metareasoning, our primary contributions are: (a) a model-free approach to meta-level control based on reinforcement learning, (2) a meta-level control technique that uses temporal difference (TD) learning, and (3) an empirical analysis that shows that our approach is effective on a common benchmark in meta-level control.

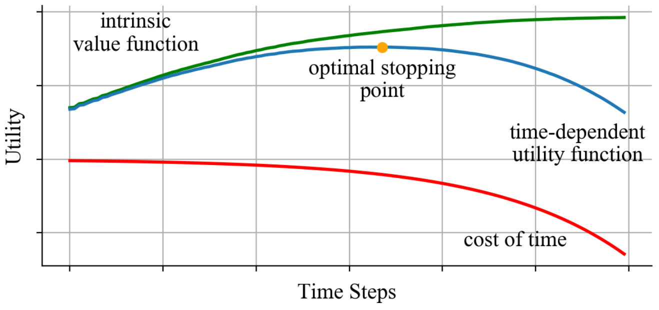

Figure 1: An idealized example of the meta-level control problem.

2 Meta-Level Control Problem

We begin by reviewing the meta-level control problem for anytime algorithms. This problem requires a model that represents the utility of a solution computed by an algorithm. Intuitively, in real-time decision-making tasks, a solution of a particular quality computed in a second has higher utility than a solution of the same quality computed in an hour. This suggests that the utility of a solution is a function of both quality and computation time [Horvitz and Rutledge, 1991; Boddy and Dean, 1994]. Accordingly, we define the utility of a solution as follows.

Definition 1. *A **time-dependent utility function**, $U(q, t)$, represents the utility of a solution of quality q at time step t.*

It is often possible to simplify a time-dependent utility function by expressing it as the difference between two functions called *object-level utility* and *inference-level utility* [Horvitz, 1988]. First, object-level utility represents the utility of a solution if we consider only the quality of that solution, ignoring the cost of computation time. Second, inference-level utility represents the utility of a solution if we take into account only the time needed to compute that solution, disregarding the value of solution quality. Adopting standard terminology [Russell and Wefald, 1991a], we define this property below [Hansen and Zilberstein, 2001].

Definition 2. *A time-dependent utility function, $U(q, t)$, is **time-separable** if the utility of a solution of quality q at time step t can be expressed as the difference between two functions, $U(q, t) = U_I(q) - U_C(t)$, where $U_I(q)$ is the **intrinsic value function** and $U_C(t)$ is the **cost of time**.*

Given a time-dependent utility function, the meta-level control problem is the problem of deciding when to interrupt an anytime algorithm and act on the current solution. Figure 1 illustrates a typical instance of the meta-level control problem [Zilberstein, 1996]. In this example, the algorithm should be interrupted at the optimal stopping point. This is the point at which the time-dependent utility function is the highest. However, in practice, the optimal stopping point often cannot be determined due to considerable uncertainty about the performance of the algorithm or the urgency for the solution. The optimal stopping point must therefore be estimated using a model of either or both variables. Similar to earlier work [Hansen and Zilberstein, 2001], we assume that there is only uncertainty about the performance of the algorithm.

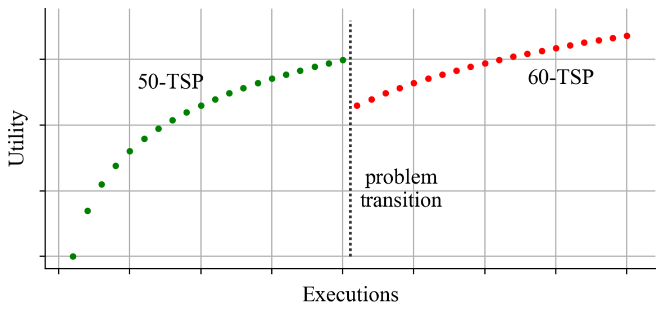

Figure 2: A typical illustration of a model-free approach to meta-level control based on reinforcement learning.

3 Model-Free Meta-Level Control

We now introduce a model-free approach to meta-level control for anytime algorithms based on reinforcement learning. Reinforcement learning has led to many mature and sophisticated methods [Sutton and Barto, 1998] that have proven to be effective across a wide range of applications from game playing [Tesauro, 1995] to helicopter control [Kim *et al.*, 2004]. The benefits of reinforcement learning methods is that they incrementally learn a policy online and readily adapt to changing circumstances. In adaptive metareasoning, this is especially important because the parameters of the meta-level control problem, particularly the characteristics of the problem, the availability of computational resources, and the settings of the anytime algorithm, often change in many real world applications. There also may not be enough time to build a meta-level control strategy—known as a *monitoring policy*—prior to the activation of the anytime algorithm.

Figure 2 depicts how reinforcement learning methods can quickly learn an effective monitoring policy for a particular problem (i.e., the *green* 50-TSP section). While the monitoring policy may temporarily degrade when the characteristics of the problem change (e.g., a *shift* from 50-TSP to 60-TSP), reinforcement learning methods can rapidly adapt the monitoring policy (i.e., the *red* 60-TSP section). Generally, when the meta-level control problem shifts gradually, particularly the hyperparameters of the problem, the system, or the anytime algorithm evolve over time, the monitoring policy can be learned and adapted by observing its performance on the meta-level control problem. Note that the monitoring policy can also be learned on the fly when there is not enough time before the activation of the anytime algorithm.

Although we are not aware of any prior use of reinforcement learning methods for meta-level control, such a framework is an especially good fit for several reasons. First, the decision to interrupt the anytime algorithm is often non-stationary because the hyperparameters of the problem, the system, and the anytime algorithm often change over time. Second, the meta-level control problem typically contains a large region of the state space that is unlikely to be reached and can be ignored in practice, which reduces the overhead required to construct an effective monitoring policy. In particular, reinforcement learning methods use experience that is limited to only reachable regions of the state space while existing dynamic programming methods must develop a *universal* policy that covers the entire state space. Third, the trade-off between exploration and exploitation is essential for learning an effective monitoring policy, especially when the parameters of the meta-level control problem change gradually. In short, the meta-level control problem shares many properties with problems for which reinforcement learning methods have been shown to be effective.

In order to develop a model-free approach to meta-level control that builds a monitoring policy using reinforcement learning methods, it is necessary to represent the meta-level control problem as a reinforcement learning problem. Reinforcement learning methods are typically applied to problems described as a *Markov decision process* (MDP). An MDP is defined by a tuple $\langle S, A, T, R, s_0 \rangle$, where S is a finite set of states, A is a finite set of actions, $T : S \times A \times S \rightarrow [0,1]$ represents the probability of reaching state $s' \in S$ after performing $a \in A$ in state $s \in S$, $R : S \times A \times S \rightarrow \mathbb{R}$ represents the expected immediate reward of reaching state $s' \in S$ after performing action $a \in A$ in state $s \in S$, and s_0 is the initial state. A solution to an MDP is a mapping $\pi : S \rightarrow A$, which could be partial, that indicates that action $\pi(s) \in A$ should be taken in state $s \in S$. A policy π induces the value function $V^\pi : S \rightarrow \mathbb{R}$ that represents the expected cumulative reward of each state. An optimal policy π^* maximizes this expected cumulative reward. Note that since the meta-level control problem has an indefinite horizon, we do not need a discount factor γ. It is always beneficial to interrupt the anytime algorithm after a finite amount of time [Hansen, 2007].

The meta-level control problem can easily be represented as an MDP. The state of computation is expressed by a tuple (q, t) that indicates the availability of a solution of quality q at time t (which is denoted by q_t). Because a solution is not available at the start of the anytime algorithm, the start state is typically $s_0 = (0, 0)$. The possible actions are either to STOP the computation or CONTINUE the computation for another time step with duration Δt. We assume that the transition function $T(s'|s, a)$ is unknown and not necessarily stationary because the underlying parameters of the meta-level control problem may change gradually. Finally, we need to define a suitable reward function. On the one hand, the reward for continuing the anytime algorithm for another time step can be expressed as the difference between two utilities: the utility of the current solution and the utility of the previous solution generated by the anytime algorithm. On the other hand, however, stopping the anytime algorithm generates no reward. We provide a formal description of the reward function below.

Definition 3. *The **immediate reward of anytime computation** (RAC) can be expressed as the difference between the utility of the current solution and the utility of the previous solution, that is:*

$$RAC(q_t,\ \textsc{Continue},\ q_{t+\Delta t}) = U(q_{t+\Delta t},\ t{+}\Delta t) - U(q_t,\ t).$$

The immediate reward for stopping the computation is always 0*, that is:*

$$RAC(q_t,\ \textsc{Stop},\ q_t) = 0.$$

It is easy to verify that such a reward function is consistent with the objective of maximizing time-dependent utility. Running an anytime algorithm until some solution of quality q_t at time t generates a cumulative reward equal to $U(q_t, t)$.

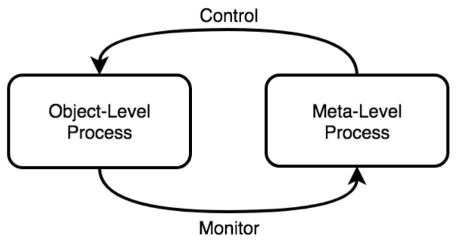

Figure 3: The metareasoning architecture of an intelligent agent.

4 Meta-Level Control Technique

In this section, we present a meta-level control technique for anytime algorithms that uses TD learning. Such a technique incrementally improves the monitoring policy after every intermediate solution generated by the anytime algorithm. In particular, when the anytime algorithm generates a new solution to the instance of the problem at hand, the intermediate reward of anytime computation can be used to update the action-value function and the monitoring policy. Note that our technique can easily be modified to use different TD learning methods. While there are many existing on-policy and off-policy TD learning methods, such as TD(λ) [Sutton, 1995; Tesauro, 1995] and Sarsa(λ) [Sutton and Barto, 1998; Chen and Wei, 2008], we outline our approach using Q-learning [Watkins and Dayan, 1992] because it has been analyzed extensively [Szepesvári, 1998; Maei *et al.*, 2010] and has seen success across a wide range of applications [Srivihok and Sukonmanee, 2005; Tan *et al.*, 2009].

Algorithm 1 describes our meta-level control technique. First, the current time step and solution quality are both initialized, the current action is selected from the ϵ-greedy policy induced by the initial action-value function, and the anytime algorithm is activated. Next, the performance of the anytime algorithm is monitored at fixed intervals. At each monitoring step, the immediate reward of anytime computation is first calculated based on the utility of the new solution and the utility of the current solution. The action-value function is then updated using the Q-learning update rule discussed below. A new action is in turn selected from the ϵ-greedy policy induced by the updated action-value function. Note that the new action may be randomly selected as a result of ϵ-greedy exploration. Finally, if the new action indicates to stop the algorithm, the anytime algorithm is interrupted and the current solution is returned. Otherwise, the anytime algorithm continues to run. The anytime algorithm is monitored at fixed intervals until interrupted or terminated naturally.

In general, when the anytime algorithm is resumed given the $d = \text{CONTINUE}$ action, the action-value function is updated using the following update rule:

$$Q(q_t,\, d) \overset{+}{\leftarrow} \alpha[RAC(q_t,\, d,\, q_{t'}) + \max_{d'} Q(q_{t'},\, d') - Q(q_t,\, d)].$$

However, when the anytime algorithm is stopped given the $d = \text{STOP}$ action, there is no change in the state of computation. It therefore offers no learning: that is, the action-value function is not updated when the anytime algorithm is interrupted. Note that it is possible to define update rules for a wide range of on-policy and off-policy TD learning methods.

Algorithm 1: A meta-level control technique that uses Q-learning with ϵ-greedy exploration.

```
Input: An anytime algorithm $A$, an exploration rate $\epsilon$, a
       learning rate $\alpha$, a duration $\Delta t$, and an action-value
       function $Q$
Output: A solution $\sigma$
1  $t \leftarrow 0$
2  $q_t \leftarrow 0$
3  $d \leftarrow \pi^\epsilon(q_t)$
4  $A.Start()$
5  $Sleep(\Delta t)$
6  while $A.Running()$ do
7      $\sigma \leftarrow A.CurrentSolution()$
8      $q_{t+\Delta t} \leftarrow \sigma.Quality()$
9      $r = U(q_{t+\Delta t}, t + \Delta t) - U(q_t, t)$
10     $Q(q_t, d) \overset{+}{\leftarrow} \alpha[r + \max_{d'} Q(q_{t+\Delta t}, d') - Q(q_t, d)]$
11     $d \leftarrow \pi^\epsilon(q_t)$
12     if $d = \text{STOP}$ then
13         $A.Stop()$
14         return $\sigma$
15     $t \leftarrow t + \Delta t$
16     $Sleep(\Delta t)$
17 return $\sigma$
```

Once the action-value function has been updated, the greedy monitoring policy can efficiently be built by performing a one-step lookahead over all actions available at the current state. Because the anytime algorithm can either be resumed or interrupted at each time step, there are only two actions that must be examined in the one-step lookahead. Constructing a greedy monitoring policy therefore requires negligible overhead after each update of the action-value function. The greedy monitoring policy is calculated as follows:

$$\pi(q_t) \leftarrow \arg\max_d Q(q_t, d).$$

This calculation can easily be modified to yield the ϵ-greedy monitoring policy $\pi^\epsilon(q_t)$ in Algorithm 1. Softmax action selection in addition to other exploration strategies can be used with little modification and negligible overhead as well.

TD learning exhibits many desirable properties that makes it particularly suitable for meta-level control. First, TD learning is guaranteed to converge to the optimal monitoring policy if none of the states are "starved" and the action-value function is a complete lookup table. Next, due to incremental updating and bootstrapping, TD learning has been observed to result in faster convergence than other reinforcement learning methods in practice. Finally, TD learning eliminates the significant offline work required by existing meta-level control techniques because it is suited for online deployment.

The state of computation has traditionally been represented using the computation time and the quality of the solution. While our technique follows such a representation, we recognize that these features may not be adequate because the transition dynamics over the computation time and the quality of the solution is likely not Markovian. Any decision-theoretic

	$\epsilon = 0.3$	$\epsilon = 0.2$	$\epsilon = 0.1$	$\epsilon = 0.01$
$\alpha = 0.1$	**15.523 ± 2.31**	**14.764 ± 2.09**	**22.510 ± 2.02**	57.082 ± 1.38
$\alpha = 0.01$	**20.301 ± 2.16**	**19.514 ± 2.59**	**17.222 ± 2.04**	47.373 ± 2.57
$\alpha = 0.001$	**17.623 ± 2.07**	**23.581 ± 1.59**	34.932 ± 2.06	47.655 ± 1.82
$\alpha = 0.0001$	39.819 ± 2.77	49.723 ± 2.07	45.593 ± 2.89	60.101 ± 0.98

Table 1: The average time-dependent utility loss (%) of the monitoring policies learned by our meta-level control technique.

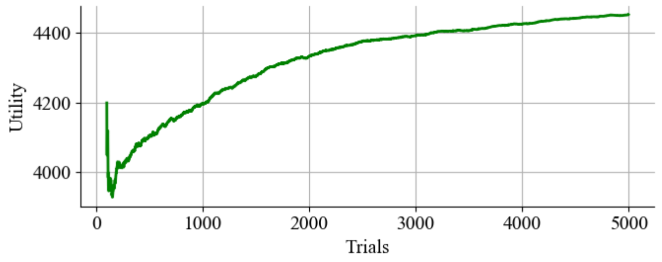

Figure 4: The rate of improvement in the monitoring policy for the hyperparameters $\alpha = 0.1$ and $\epsilon = 0.2$ during training.

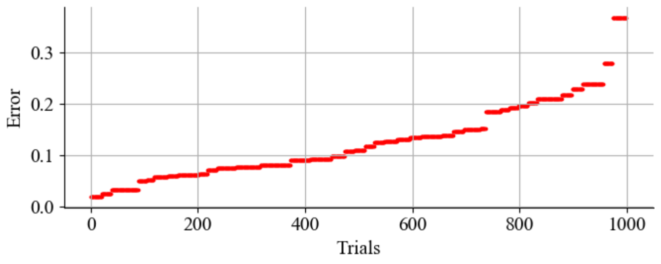

Figure 5: The time-dependent utility losses of the monitoring policy for the hyperparameters $\alpha = 0.1$ and $\epsilon = 0.2$ during testing.

approach to meta-level control, including methods that use either planning or reinforcement learning, could therefore benefit from a richer representation of the state of computation. While it is encouraging that we observe near optimal monitoring policies given only the computation time and the quality of the solution on a common benchmark in meta-level control, we emphasize that our technique can readily be modified to exploit problem-specific instances features.

It is easy to see how alternative reinforcement learning methods could be used by a meta-level control technique as well. For instance, a technique that uses Monte Carlo (MC) learning [Wang *et al.*, 2012] would update the monitoring policy only after the final solution of the anytime algorithm. In particular, once the anytime algorithm has terminated and generated the final solution, the cumulative rewards (i.e., the reward-to-go from each state along the trial sequence) can be used to update the action-value function and the monitoring policy accordingly. In contrast, because our technique uses TD learning, it updates the action-value function and the monitoring policy after every intermediate solution of the anytime algorithm. While TD learning is likely a better approach to meta-level control given incremental updating and bootstrapping, MC learning can be more tolerant of violations of the Markov property. It is also possible to use TD(λ), a more general approach that blends MC and TD learning.

5 Experiments

We show that our meta-level control technique learns a near optimal monitoring policy on a common benchmark in meta-level control of anytime algorithms [Hansen and Zilberstein, 2001]. In particular, the Lin-Kernighan heuristic is a tour improvement algorithm that solves the traveling salesman problem (TSP) approximately [Lin and Kernighan, 1973]. The algorithm starts with an initial tour and gradually improves that tour by swapping specific subtours until convergence. In practice, the algorithm generally achieves a solution within 90% of the optimal solution. Note that the algorithm represents the behavior of a wide range of anytime algorithms that have been deployed across a number of domains.

All experiments use our meta-level control technique and the Lin-Kernighan heuristic to solve different instances of a 50-TSP. In general, each experiment includes a training step and a testing step. First, we train our technique on 5,000 random instances of the 50-TSP. During the training step, our technique learns the monitoring policy using Q-learning with ϵ-greedy exploration as described in Algorithm 1. Finally, after the training step, we test our technique on 1,000 random instances of the 50-TSP. During the testing step, our technique uses the monitoring policy learned during the training step. Note that the monitoring policy remains fixed because improvement and exploration have been disabled.

Figure 3 shows the architecture of an intelligent agent with metareasoning capabilities. Each experiment represents a typical instance of the meta-level control problem where an intelligent agent must decide when to interrupt an anytime algorithm and act on the current solution. To do this, we run two processes in parallel. The *object-level process* uses an anytime algorithm to solve an instance of a particular problem. At the same time, the *meta-level process* uses our meta-level control technique to monitor and control the anytime algorithm at fixed intervals. The experiment concludes when the anytime algorithm is either interrupted or terminated naturally. Note that our technique monitors the anytime algorithm approximately every tenth of a second.

As discussed earlier, meta-level control requires a time-dependent utility function. Similar to earlier work, we define the time-dependent utility of a solution of quality q at time step t as the function, $U(q, t) = \alpha q - e^{\beta t}$, where the rates α and β are selected in practice based on the value of a solution and the urgency for a solution [Hansen and Zilberstein, 2001]. In all experiments, we deliberately select rates to avoid trivializing the problem (e.g., by making the urgency for a solution so high that the algorithm is interrupted immediately or so low that it runs to completion). Note that the first and second terms of the time-dependent utility function are the intrinsic value function and the cost of time respectively.

Ideally, the quality of a solution can be defined as the approximation ratio, $q = c^*/c$, where c^* is the cost of the optimal solution and c is the cost of the current solution. However, because the cost of the optimal solution cannot quickly be computed for the benchmark problems, we estimate the quality of a solution as the approximation ratio, $q = \ell/c$, where ℓ is a problem-dependent lower bound on the optimal solution. For the TSP, we estimate solution quality, specifically the tour quality, using the lower bound, ℓ_{tsp}, defined as the length of the minimum spanning tree of the TSP, which is computed in polynomial time using Prim's algorithm.

Table 1 shows the effectiveness of the monitoring policies learned by our meta-level control technique for a range of learning rates α and exploration rates ϵ. We evaluate our technique using the *average time-dependent utility loss*. This is expressed as the average difference between the time-dependent utility of the learned monitoring policy and the optimal time-dependent utility for all trials in the testing step. Observe that the hyperparameters $\alpha = 0.1$ and $\epsilon = 0.2$ result in the best monitoring policy. The entries with an average time-dependent utility loss under 25% have been highlighted. Note that *lower* values indicate *better* performance.

Figure 4 illustrates the rate of improvement in the monitoring policy when our meta-level control technique uses the learning rate $\alpha = 0.1$ and the exploration rate $\epsilon = 0.2$ during the training step. Observe that our technique nearly converges to the optimal time-dependent utility. The early trials are noisy because our technique starts with a random monitoring policy that must stabilize rapidly before learning.

Figure 5 describes the performance of the monitoring policy learned by our meta-level control technique for the learning rate $\alpha = 0.1$ and the exploration rate $\epsilon = 0.2$ during the testing step. Each point indicates the time-dependent utility loss for a particular trial. Observe that our technique only performs poorly on a handful of trials. This is likely because our technique did not encounter similar trials during the training step. While we use a lookup table to represent the action-value function, more sophisticated function approximations may result in faster convergence and a better monitoring policy [Konidaris *et al.*, 2011]. Note that the trials have been sorted in the order of increasing time-dependent utility loss.

6 Conclusion

Building on our recent work in adaptive metareasoning, we propose a model-free approach to meta-level control based on reinforcement learning. In particular, we offer a meta-level control technique that uses TD learning. It eliminates many assumptions of current techniques: not only does it obviate extensive offline work, but it also adapts to each individual instance of the problem at hand. Finally, we show empirically that our approach is effective on a common benchmark in meta-level control. Future work will explore richer feature-based representations of the state of computation as well as other reinforcement learning methods with more sophisticated function approximations.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. This work was supported in part by the National Science Foundation grants IIS-1405550 and IIS-1724101.

References

[Boddy and Dean, 1994] Mark Boddy and Thomas L. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67(2):245–285, 1994.

[Breese and Horvitz, 1991] John S. Breese and Eric J. Horvitz. Ideal reformulation of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 129–143, 1991.

[Chen and Wei, 2008] Sheng-Lei Chen and Yan-Mei Wei. Least-squares SARSA(Lambda) algorithms for reinforcement learning. In *Proceedings of the Fourth International Conference on Natural Computation*, pages 632–636, 2008.

[Cox and Ram, 1999] Michael T. Cox and Ashwin Ram. Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence*, 112(1):1–55, 1999.

[Hansen and Zilberstein, 2001] Eric A. Hansen and Shlomo Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1-2):139–157, 2001.

[Hansen *et al.*, 1997] Eric A. Hansen, Shlomo Zilberstein, and Victor A. Danilchenko. Anytime heuristic search: First results. Technical Report 97-50, Computer Science Department, University of Massachussetts Amherst, 1997.

[Hansen, 2007] Eric A. Hansen. Indefinite-horizon POMDPs with action-based termination. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence*, pages 1237–1242, 2007.

[Horvitz and Rutledge, 1991] Eric Horvitz and Geoffrey Rutledge. Time-dependent utility and action under uncertainty. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 151–158, 1991.

[Horvitz, 1987] Eric J. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of the Third Workshop on Uncertainty in Artificial Intelligence*, 1987.

[Horvitz, 1988] Eric Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceedings of the Seventh AAAI Conference on Artificial Intelligence*, pages 111–116, 1988.

[Horvitz, 1990] Eric J. Horvitz. *Computation and action under bounded resources*. PhD thesis, Stanford University, CA, 1990.

[Kim *et al.*, 2004] H.J. Kim, Michael I. Jordan, Shankar Sastry, and Andrew Y. Ng. Autonomous helicopter flight via reinforcement learning. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 799–806, 2004.

[Konidaris *et al.*, 2011] George Konidaris, Sarah Osentoski, and Philip S Thomas. Value function approximation in reinforcement learning using the Fourier basis. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, volume 6, page 7, 2011.

[Lin and Kernighan, 1973] Shen Lin and Brian W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21(2):498–516, 1973.

[Luna *et al.*, 2013] R. Luna, I. A. Şucan, M. Moll, and L. E. Kavraki. Anytime solution optimization for sampling-based motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5068–5074, 2013.

[Maei *et al.*, 2010] Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S. Sutton. Toward off-policy learning control with function approximation. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, pages 719–726, 2010.

[Paul *et al.*, 1991] C. J. Paul, Anurag Acharya, Bryan Black, and Jay K. Strosnider. Reducing problem-solving variance to improve predictability. *Communications of the ACM*, 34(8):80–93, 1991.

[Pineau *et al.*, 2003] Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pages 1025–1032, 2003.

[Ramos and Cozman, 2005] Fabio Tozeto Ramos and Fabio Gagliardi Cozman. Anytime anyspace probabilistic inference. *International Journal of Approximate Reasoning*, 38(1):53 – 80, 2005.

[Richter *et al.*, 2010] Silvia Richter, Jordan Tyler Thayer, and Wheeler Ruml. The joy of forgetting: Faster anytime search via restarting. In *Proceedings of the Conference on Automated Planning and Scheduling*, pages 137–144, 2010.

[Richtsfeld *et al.*, 2013] Andreas Richtsfeld, Michael Zillich, and Markus Vincze. Anytime perceptual grouping of 2D features into 3D basic shapes. In *Proceedings of the Ninth International Conference on Computer Vision Systems*, pages 73–82, 2013.

[Russell and Wefald, 1991a] Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 49:361–395, 1991.

[Russell and Wefald, 1991b] Stuart J. Russell and Eric H. Wefald. *Do the Right thing: Studies in Limited Rationality*. MIT Press, Cambridge, MA, USA, 1991.

[Simon, 1947] Herbert A. Simon. *Administrative Behavior*. Macmillan, New York, NY, USA, 1947.

[Simon, 1982] Herbert A. Simon. *Models of Bounded Rationality*. MIT Press, Cambridge, MA, USA, 1982.

[Srivihok and Sukonmanee, 2005] Anongnart Srivihok and Pisit Sukonmanee. E-commerce intelligent agent: Personalization travel support agent using Q-learning. In *Proceedings of the Seventh International Conference on Electronic Commerce*, pages 287–292, 2005.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.

[Sutton, 1995] Richard S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3(1):9–44, 1995.

[Svegliato *et al.*, 2018] Justin Svegliato, Kyle Hollins Wray, and Shlomo Zilberstein. Meta-level control of anytime algorithms with online performance prediction. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, 2018.

[Szepesvári, 1998] Csaba Szepesvári. The asymptotic convergence-rate of Q-learning. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 1064–1070, 1998.

[Tan *et al.*, 2009] Ying Tan, Wei Liu, and Qinru Qiu. Adaptive power management using reinforcement learning. In *Proceedings of the International Conference on Computer-Aided Design*, pages 461–467, 2009.

[Tesauro, 1995] Gerald Tesauro. Temporal difference learning and TD-gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[Wang *et al.*, 2012] Yi Wang, Kok S. Won, Wee S. Lee, and Daniel J. Hsu. Monte Carlo Bayesian reinforcement learning. In *Proceedings of the Twenty-Ninth International Conference on Machine Learning*, pages 1135–1142, 2012.

[Watkins and Dayan, 1992] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[Zilberstein and Russell, 1995] Shlomo Zilberstein and Stuart J. Russell. Approximate reasoning using anytime algorithms. In S. Natarajan, editor, *Imprecise and Approximate Computation*, pages 43–62. Springer, 1995.

[Zilberstein, 1993] Shlomo Zilberstein. *Operational Rationality through Compilation of Anytime Algorithms*. PhD thesis, Computer Science Division, University of California Berkeley, 1993.

[Zilberstein, 1996] Shlomo Zilberstein. Using anytime algorithms in intelligent systems. *AI Magazine*, 17(3):73, 1996.

[Zilberstein, 2011] Shlomo Zilberstein. Metareasoning and bounded rationality. In M. Cox and A. Raja, editors, *Metareasoning: Thinking about Thinking*, pages 27–40. MIT Press, Cambridge, MA, USA, 2011.