Report Number: wucse-2009-15

2009

Scheduling Design with Unknown Execution Time Distributions or Modes

Robert Glaubius, Terry Tidwell, Christopher Gill, and William D. Smart

Open soft real-time systems, such as mobile robots, experience unpredictable interactions with their environments and yet must respond both adaptively and with reasonable temporal predictability. Because of the uncertainty inherent in such interactions, many of the assumptions of the real-time scheduling techniques traditionally used to ensure predictable timing of system actions do not hold in those environments. In previous work we have developed novel techniques for scheduling policy design where up-front knowledge of execution time distributions can be used to produce both compact representations of resource utilization state spaces and efficient optimal scheduling policies over those state spaces. This... Read complete abstract on page 2.

Scheduling Design with Unknown Execution Time Distributions or Modes

Robert Glaubius, Terry Tidwell, Christopher Gill, and William D. Smart

Complete Abstract:

Open soft real-time systems, such as mobile robots, experience unpredictable interactions with their environments and yet must respond both adaptively and with reasonable temporal predictability. Because of the uncertainty inherent in such interactions, many of the assumptions of the real-time scheduling techniques traditionally used to ensure predictable timing of system actions do not hold in those environments. In previous work we have developed novel techniques for scheduling policy design where up-front knowledge of execution time distributions can be used to produce both compact representations of resource utilization state spaces and efficient optimal scheduling policies over those state spaces. This paper makes two main contributions beyond our previous work, to the state of the art in scheduling open soft real-time systems: (1) it shows how to relax the assumption that the entire distribution of execution times is known up front, to allow online learning of an execution time distribution during system run-time; and (2) it shows how to relax the assumption that the execution time of a system action can be characterized by a single distribution, to accommodate different execution time distributions for an action being taken in one of multiple modes. Each of these contributions allows a wider range of system actions to be scheduled adaptively and with temporal predictability, which increases the applicability of our approach to even more general classes of open soft real-time systems.

2009-15

Scheduling Design with Unknown Execution Time Distributions or Modes

Authors: Robert Glaubius, Terry Tidwell, Christopher Gill, and William D. Smart

Corresponding Author: rlg1@cse.wustl.edu

Abstract: Open soft real-time systems, such as mobile robots, experience unpredictable interactions with their environments and yet must respond both adaptively and with reasonable temporal predictability. Because of the uncertainty inherent in such interactions, many of the assumptions of the real-time scheduling techniques traditionally used to ensure predictable timing of system actions do not hold in those environments. In previous work we have developed novel techniques for scheduling policy design where up-front knowledge of execution time distributions can be used to produce both compact representations of resource utilization state spaces and efficient optimal scheduling policies over those state spaces.

This paper makes two main contributions beyond our previous work, to the state of the art in scheduling open soft real-time systems: (1) it shows how to relax the assumption that the entire distribution of execution times is known up front, to allow online learning of an execution time distribution during system run-time; and (2) it shows how to relax the assumption that the execution time of a system action can be characterized by a single distribution, to accommodate different execution time distributions for an action being taken in one of multiple modes. Each of these contributions allows a wider range of system actions to be scheduled adaptively and with

Notes:
On-line version of paper submitted to RTSS 2009, with full proof in Appendix A.

Type of Report: Other

Scheduling Design with Unknown Execution Time Distributions or Modes

Robert Glaubius, Terry Tidwell, Christopher Gill, and William D. Smart
{rlg1,ttidwell, cdgill, wds}@cse.wustl.edu
Department of Computer Science and Engineering
Washington University, St. Louis

***Abstract*—**

Open soft real-time systems, such as mobile robots, experience unpredictable interactions with their environments and yet must respond both adaptively and with reasonable temporal predictability. Because of the uncertainty inherent in such interactions, many of the assumptions of the real-time scheduling techniques traditionally used to ensure predictable timing of system actions do not hold in those environments. In previous work we have developed novel techniques for scheduling policy design where up-front knowledge of execution time distributions can be used to produce both compact representations of resource utilization state spaces and efficient optimal scheduling policies over those state spaces.

This paper makes two main contributions beyond our previous work, to the state of the art in scheduling open soft real-time systems: (1) it shows how to relax the assumption that the entire distribution of execution times is known up front, to allow online learning of an execution time distribution during system run-time; and (2) it shows how to relax the assumption that the execution time of a system action can be characterized by a single distribution, to accommodate different execution time distributions for an action being taken in one of multiple modes. Each of these contributions allows a wider range of system actions to be scheduled adaptively and with temporal predictability, which increases the applicability of our approach to even more general classes of open soft real-time systems.

I. Introduction

Open soft real-time systems, such as mobile robots, must respond adaptively to varying operating conditions. In many situations, such as when a mobile robot approaches a physical obstacle or a branch in its possible navigation path, decisions must be made and enacted within specific timing constraints (e.g., before hitting the obstacle or passing a way point at which a different path should have been taken). In previous work [1], [2], [3], we described a method for automatically synthesizing optimal scheduling policies tailored to the particular operating parameters of such systems. Specifically, given a distribution describing the execution time for each task and a desired resource share, that method automatically generates a scheduling policy that optimally shares that resource under our system model.

That method is suitable for an important category of system actions where a single characteristic distribution of execution times can be determined readily *a priori*, for example the movement of a robot's pan-tilt unit from one position to another. However, for other actions it may be difficult to characterize their execution time distributions before the system is running, and instead decisions must be made dynamically, based on observations made at run-time. Furthermore, the execution time distribution for an action may differ from one mode of system operation to another. For example, as we illustrate in Figure 1 in Section II the dwell time required for image capture by a digital camera may depend significantly on both the scene being captured and whether or not the camera's view of the scene is occluded. To increase the applicability of our approach to such cases, new techniques for scheduling policy design are thus needed.

To address that need, in this paper we refine the system model used in our previous work by allowing either of two key simplifying assumptions to be relaxed (while still retaining the other assumption): (1) that we can obtain a full characterization of system behavior *a priori*; or (2) that a single mode of operation is sufficient to describe the system's behavior. While our long-term goal is to design scheduling policies for systems with both unknown modes and unknown task execution time distributions within those modes, learning optimal scheduling policies for a system with both of these competing sources of uncertainty appears to be quite challenging in general – consider for example the problem of how to distinguish whether an outlying observation belongs to a given mode, particularly initially when few observations have been made.

In this paper we therefore relax each of the two assumptions separately, showing how to generate scheduling policies automatically for either case, as a necessary and important first step towards our more general goal of considering them together which we defer to future work. For brevity we focus our treatment of these extensions in the context of our previously described scheduling policy design techniques [2]. However, the new techniques we present in this paper can be adapted to work within any framework that allows scheduling policy synthesis given a refinable model of tasks' durations and desired scheduling properties, such as their observed worse case execution times and task priorities in the case of Rate Monotonic Scheduling [4].

This research was supported in part by NSF grants CNS-0716764 (Cybertrust) and CCF-0448562 (CAREER).

In Section II we describe our resulting system model with these two alternative refinements. In Section III we describe our solution approaches for online learning of execution time distributions and modes. In Section IV we present analytical and experimental evauations of these solution approaches. In Section V we discuss related work. In Section VI we present our conclusions about the techniques presented in this paper, and describe future directions for this research.

II. System Model

In previous work [1], we proposed a system model in which: (1) multiple threads of execution require mutually exclusive use of a single common resource (e.g., a CPU); (2) whenever a thread is granted the resource, it occupies the resource for a finite and bounded subsequent duration; (3) the duration for which a thread occupies the resource may vary from run to run but overall obeys a known, independent and bounded distribution; and (4) a scheduler repeatedly chooses which thread to run according to a given scheduling policy, dispatches that thread, and waits until the end of the duration during which the thread occupies the resource.

That system model established a basic foundation for scheduling policy design in open soft real-time systems built atop commonly used operating systems such as Linux or VxWorks. For example, within the Linux kernel, hard and soft interrupts may be threaded and placed under scheduler control [5], with different resulting durations of resource occupation for the different kinds of interrupts. However, for cyber-physical systems that basic system model is not sufficient. Since complex interactions among physical, computational, and communication components may result in execution time distributions that (1) are initially unknown or at the very least are sufficiently hard to characterize *a priori*; or (2) are distinctly different in different modes of system operation.

a) Unknown Execution Time Distributions:: In the first case, we may only have access to a poor model of the distribution that governs the execution time of each system action, if we have any knowledge at all. Therefore, we must learn these distributions from observations obtained while the system runs. This extension to our system model provides a foundation for scheduling enforcement in complex cyber-physical systems such as mobile robots, where unforeseen factors such as the exact state of the environment may make accurate *a priori* characterization of execution times infeasible.

In addition, run-time adaptation of a model of execution times may be vital if the model differs significantly from what is observed online. For example, execution time distributions generated through simulations or observations made during previous runs of a system may be useful to establish reasonable initial scheduling behavior, but if distributions of subsequently observed execution times differ, the scheduling policy should be adapted to use those more situational observations.

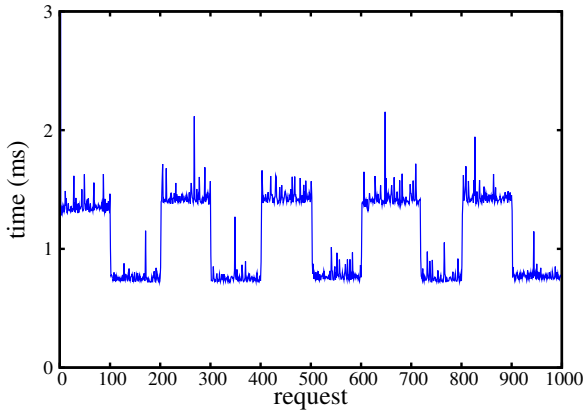


Fig. 1. Digital camera image capture times: the lower capture times when the camera's view is occluded and the higher capture times when it isn't, are observably distinguished by those modes.

b) Unknown Modes of Operation:: In the second case, the execution time distributions are known *a priori* as before, but the system may operate in one of a number of possible known modes. Each mode has its own independent distributions of the execution times for each system action in that mode. While all distributions for all actions in all modes are known before run-time, the particular mode the system is in at any given time must be inferred by observation.

c) Illustrative Example:: A common action in mobile robotics is to capture a camera image to obtain information about a robot's environment. Because many cameras deployed on common robotics platforms do image compression as a normal part of their data processing, the response time of this system action depends on properties of the scene being imaged. For the first case, a camera with an unobstructed view of a sufficiently rich scene will experience different image capture times simply by pointing it at different parts of the scene. For the second case, if the lighting of the scene changes or if the camera's view of the scene is occluded, distinctly different image capture times may result from those lighting or occlusion modes for a given camera orientation. To illustrate these effects, we alternately occluded and uncovered a camera and measured its image capture times during those occlusion modes. Figure 1 shows the results of those trials, with the observed image capture times being noticeably smaller when the camera's view was obscured than when it had an un-obscured view of the scene.

As we noted in Section I, we consider unknown distributions vs. unknown modes separately in the techniques presented in this paper, though this illustrative example

motivates the utility of both our separate treatment of these here and our plans for considering them jointly in future work. Some thoughts on this synthesis are described in Section VI. In Section III we next describe the intuition behind our previous solution approach, and how we have extended that approach to handle each of these cases.

III. Solution Approach

In this section we first describe our system representation and how it is used to automate optimal scheduling policy design. This technique serves as the foundation for each of the extensions described in this paper. We then describe these extensions in detail, starting with our approach for systems where (1) the distribution of execution times is not known up front and must be learned online. We then examine the case where (2) the system operates in any of a number of different modes.

We model each mode of the task scheduling problem as a Markov Decision Process [6], [7], or MDP. An MDP consists of a set of states $\mathcal{X}$, actions $\mathcal{A}$, a real-valued cost function that indicates the immediate benefit or detriment of entering a state, and a transition system that describes how states evolve. We also have a set of decision epochs $d = 0, 1, 2, \ldots$ associated with an MDP such that at each decision epoch, a controller observes its current state x_d and chooses an action a_d, and then the system transitions to a new state x_{d+1} and emits cost c_{d+1}.

A task scheduling problem *mode* is defined as an MDP with actions corresponding to the decision to dispatch a particular task; the system state is the cumulative utilization of each task, so that $x = (x_1, x_2, \ldots, x_n)$ is a vector with non-negative integer components. In state x the system transitions to $y = x + t\Delta_a$ with probability $P(t|a)$, where Δ_a is a vector with the a^{th} component equal to one and all other components zero; in other words, running task a increases its cumulative utilization, x_a, by the duration t with probability $P(t|a)$. The cost of entering a state x, $c(x)$, is defined in terms of the distance between x and the ideal utilization at time $\tau(x)$, $\tau(x)u$:

$$c(x) = \sum_{a=1}^{n} |x_a - \tau(x)u_a|, \tag{1}$$

where the elapsed time at x, $\tau(x)$, is the sum of each task's cumulative utilization,

$$\tau(x) = \sum_{a=1}^{n} x_a.$$

A *policy* π maps states to actions. The value V^π of a policy is the expected, discounted sum of long-term costs observed while following that policy

$$V^\pi(x) = \mathbf{E}\left\{\sum_{d=0}^{\infty} \gamma^d c(x_d) | x_0 = x\right\}, \tag{2}$$

where the expectation is taken with respect to the probability of observing x_{d+1} after dispatching $\pi(x_d)$ in x_d. The discount factor $\gamma \in [0, 1)$ weights future costs against more immediate ones. The optimal value function V is the value of any policy that minimizes Equation 2. It is well-established that the optimal value function satisfies the Bellman equation

$$V(x) = \max_{a \in \mathcal{A}} \{Q(x, a)\}, \tag{3}$$

where

$$Q(x, a) = \sum_{t=1}^{\infty} P(t|a)[\gamma V(x + t\Delta_a) - c(x + t\Delta_a)] \tag{4}$$

is the value of choosing action a in x then behaving optimally afterwards.

Since methods for computing the optimal policy are not the focus of this work, we omit discussion of these methods here. In previous work [2], [3] we described methods that allow us to represent the value function despite the infinite state space under consideration. This is the basis of our automatic scheduling design technique. In order to extend this approach past the limitations of our previous work we now look at two extensions.

d) Online Learning:: We take an *indirect approach* to online reinforcement learning in this work [8]. We use observations of the system to estimate a system model; we then use the optimal solution for this estimate as an approximation to the optimal solution of the original system. This is in contrast to *direct approaches* such as Q-learning [9], which use the observations to tune the parameters of an approximation to Q directly. Since the transition function is defined with respect to the task durations, which are independent of the system state, constructing a transition model reduces to estimating the task durations. Combined with the fact that the cost function is determined completely by the user-specified utilization target, our indirect, model-based approach appears most appropriate in this domain.

Under our system model, task durations are described by discrete probability histograms. We approximate the task distribution by simply maintaining histograms corresponding to historical observations of the system that we want to control. Suppose we have a collection of past observations (a_d, t_d), where task a_d was run for t_d time quanta at decision epoch d, for $d = 1, \ldots, m$. Then let $O_m(a)$ be the number of observations with $a_d = a$, and let $O_m(a, t)$ be the number of observations with both $a_d = a$ and $t_d = t$. Then our approximate system model consists of the histogram estimators for each task a, defined as

$$P_m(t|a) = O_m(a, t)/O_m(a).$$

In Section IV, we present analytical results for the sample complexity of learning an optimal scheduling policy given that we initially have no knowledge of the

task durations. We also present empirical results considering a number of exploration strategies that address the exploration-exploitation trade off: *how often should we make decisions that are apparently suboptimal in order to reduce uncertainty, and so improve our chances for quickly discovering the true optimal behavior?*

We decompose our analytical results into two parts by examining the sample complexity of guaranteeing an accurate estimate of the duration density, then looking at the accuracy of our approximate policies as a function of the density estimator's residual. We expect that this will allow for straightforward generalization of our results to richer descriptions of stochastic tasks.

e) Multi-Mode Systems:: In order to schedule tasks in systems with a set of modes, $\mathcal{M}$, we must first derive the set of scheduling policies $\{\pi_M | M \in \mathcal{M}\}$ where π_M is the optimal scheduling policy for mode M. Here we assume that distributions for each task are known in each mode, and that tasks are common across all modes; this allows us to use our scheduling synthesis techniques from previous work directly. Given the current system mode, the scheduler just follows the policy synthesized for that mode.

This leaves the problem of estimating the current system mode, which in this extension to our system model can only be learned via observation. The probability of a task a running for duration t, in mode $M \in \mathcal{M}$ is given by $P(t|a, M)$ and is defined by the duration distributions for each mode. Given a collection of past observations (a_d, t_d), where task a_d was run for t_d time quanta at decision epoch d, for $d = 1, \ldots, m$ the maximally likely current mode is

$$\underset{M \in \mathcal{M}}{\operatorname{argmax}} \left\{ \prod_{i=1}^{w} P(t_i | a_i, M) \right\}.$$

Using this approach, w, the window of past observations to use in estimating the current mode, becomes a design parameter which affects two different quantities in the system: (1) the expected lag time between an actual mode change and when that mode change is observed; and (2) the expected error rate for mislabeling the current mode. Intuitively we expect that larger values of w should result in higher expected lag times as more evidence is required to decide that a mode change has occurred, while resulting in lower error rates for labeling the current mode, as this mode prediction is made with more information. Alternately, we expect smaller values of w to result in lower lag times, but higher error rates for current mode prediction.

This straightforward approach to mode identification is a direct outgrowth of our system model. However, this approach makes several two key assumptions about the nature of modes in the system: that (1) the probability of switching away from the current system mode is low and (2) the probability of switching between distinct modes is uniform for all pairs of modes.

The first assumption, that mode changes are relatively rare events, is necessary because otherwise it may not be optimal to follow the policy prescribed by a particular mode. To calculate the true optimal policy for multi-mode systems would require evaluating the cost of future actions across mode boundaries. If the probability of moving into other modes is sufficiently low, the cost within the current mode dominates the overall expected cost, and the optimal mode is exactly the policy synthesized for the system regardless of other modes.

The second assumption, that the probability of switching between any distinct pair of modes is uniform simplifies mode estimation. Altering this assumption would require maintaining additional information about the mode structure; in general, the relationship between modes may best be described as a probabilistic graph-structured model. We defer investigating appropriate strategies for maintaining this information as a subject for future work. We now present experimental results that demonstrate the trade offs between lag and overall error rate induced by different window sizes.

IV. Evaluation

A. Learning Task Distributions Online

In the presence of uncertainty about the distribution of task durations, we must consider the relative merit of choosing exploratory actions over exploiting our current model of the environment. It is well-known in the reinforcement learning literature [10], [11] that there is a trade off between exploring and exploiting; an agent that chooses apparently suboptimal actions may obtain better long-term performance due to its access to more accurate information about the system, while an agent that acts greedily with respect to its current model may lock itself into a local minimum and so never achieve optimal performance.

In this section we examine the question of behaving optimally online while learning the task durations. We approach this question both analytically and empirically. We derive a PAC-style bound [12], [13] on the number of mistakes made by an exploration policy. We then perform an empirical comparison of a number of exploration strategies to evaluate their practical effects on the behavior of the system.

f) Analytical Results:: We characterize the accuracy of an approximate policy in terms of the likelihood of making a mistake. A mistake may occur whenever the approximate policy and the optimal policy disagree. However, it turns out that we can not guarantee every potential mistake is avoided with high probability without requiring arbitrarily many observations, since any magnitude of error

$|P_m(t|a) - P(t|a)|$ may result in a mistake when two tasks have sufficiently similar costs.

The optimal policy π chooses actions greedily with respect to Q; that is,

$$\pi(x) = \operatorname*{argmax}_{a \in \mathcal{A}} \{Q(x, a)\}.$$

Similarly, our approximation to the optimal policy, π_m, selects actions greedily with respect to Q_m, the approximate state-action value function. This approximation is obtained by solving the Bellman equation (see Equations 3 and 4 in Section III) with P_m substituted for P.

The approximate optimal policy π_m does not necessarily make a mistake if it differs from π at x, since it is possible that $Q(x, \pi_m(x)) = Q(x, \pi(x))$; i.e., there may be more than one optimal policy. A mistake occurs when $Q(x, \pi(x)) > Q(x, \pi_m(x))$ and $Q_m(x, \pi_m(x)) > Q_m(x, \pi(x))$; that is, a mistake occurs when the value of these actions is inverted between the approximate and exact state-action value functions. Considering the ordering of these particular actions given an approximation P_m is complicated, since we can not easily separate Q and Q_m from the optimal policies for the respective MDPs. Instead, we will consider the more general problem of getting the ordering of actions under Q_m to match up with the ordering under Q. Additionally, we mentioned above that we can make a mistake any time P_m is not exact. With this in mind, we introduce a tolerance parameter $\epsilon > 0$ and a certainty parameter $\delta > 0$. Our objective is to choose a sample that is sufficiently large to guarantee that for *any* pair of tasks a and b, $Q_m(x, a) > Q_m(x, b)$ with probability at least $(1-\delta)$ whenever $Q(x, a) - Q(x, b) > \epsilon$.

The intuition behind this is that we want to consider the risk of making a mistake. When $Q(x, a) - Q(x, b)$ is near zero, there is not a substantial difference in long-term cost under either action, so we are more lenient about mistakes in these cases. When this difference grows large, it is more important to get the ordering right to minimize costs (notice that in order to get the optimal policy right, it is sufficient that the optimal action maximizes $Q_m(x, \cdot)$ even if the approximation is not particularly accurate).

We address the sample complexity of approximating the optimal policy in two parts. First, we consider the sample complexity of guaranteeing with high probability that the distribution estimation error is bounded. Then we consider the probability of making an expensive mistake given that error bound. This simplifies the analysis and also would allow us to vary our strategy for estimating the task models without substantially impacting the rest of the analysis.

Suppose that we want to choose enough samples m to guarantee with high probability that $|P_m(t|a) - P(t|a)| \leq \lambda$ for some target error $\lambda > 0$. Specifically, we want m large enough to guarantee that

$$\mathbf{P}\{|P_m(t|a) - P(t|a)| \geq \lambda\} \leq \delta. \tag{5}$$

We make the simplifying assumption that we have an equal number of observations of each task; that is, $O_m(a) = \nu = m/n$ for every task. We then can rewrite this probability as follows:

$$\begin{aligned} &\mathbf{P}\{|P_m(t|a) - P(t|a)| \geq \lambda\} \\ \equiv\ &\mathbf{P}\{|O_m(a, t) - \nu P(t|a)| \geq \nu\lambda\} \end{aligned}$$

$O_m(a, t)$ is the sum of outcomes of ν independent Bernoulli trials, each with success rate $P(t|a)$. This means that $\mathbf{E}\{O_m(a, t)\} = \nu P(t|a)$, so we can apply Hoeffding's inequality:

$$\mathbf{P}\{|P_m(t|a) - P(t|a)| \geq \lambda\} \leq 2e^{-2(\nu\lambda)^2},$$

so that Equation 5 is satisfied whenever

$$2e^{-2(\nu\lambda)^2} \leq \delta,$$

or equivalently, since $\nu = m/n$,

$$m > \frac{n}{\lambda\sqrt{2}} \left(\ln(2/\delta)\right)^{1/2}. \tag{6}$$

This is consistent with our expectations: if we demand greater certainty by decreasing δ, or greater accuracy by decreasing λ, or if we increase the number of tasks, we will need more observations.

Now we consider the likelihood of making a mistake given that $|P_m(t|a) - P(t|a)|$ is uniformly bounded above by λ. We can avoid inverting the order of actions a and b under Q_m when $Q(x, a) - Q(x, b) > \epsilon$ by requiring with high probability that the approximation error is at most $\epsilon/2$. We can bound this approximation error uniformly in terms of λ, as shown in Proposition 1.

Proposition 1. *If there is a constant λ such that for all tasks a and durations t, $|P_m(t|a) - P(t|a)| \leq \lambda$ and if there is a finite constant T such that $P(t|a) = 0$ whenever $t > T$, then for every state x and task a,*

$$|Q_m(x, a) - Q(x, a)| \leq \frac{\lambda T(T-1)}{(1-\gamma)^2}.$$

A proof of the proposition is given in Appendix A. This is similar to the Simulation Lemma of Kearns and Singh [14]; one notable difference is that their result depends on having a finite number of states and bounded costs, whereas our bound depends on each state having a finite number of successors, and a bound on the growth rate of costs. This is consistent with results from Kakade's thesis [15] indicating that the sample complexity of obtaining a good approximation should depend polynomially on the number of parameters to the transition model. In general MDPs this is quadratic in the number of states, but in our problem this is finite while the number of states is not.

In order to guarantee that we avoid expensive mistakes with tolerance ϵ, it is sufficient to require

$$\lambda T(T-1)/(1-\gamma)^2 \leq \epsilon/2,$$

or equivalently,

$$\lambda \leq (1-\gamma)^2\epsilon/[2T(T-1)].$$

Plugging this into Equation 6 indicates that m observations are sufficient to avoid inverting actions with $Q(x,a) - Q(x,b) > \epsilon$ with probability at least $(1-\delta)$, where

$$m > \frac{2T(T-1)n}{(1-\gamma)^2\epsilon\sqrt{2}}(\ln(2/\delta))^{1/2}.$$

This bound indicates that the number of observations increases linearly in the number of tasks. The number of observations also grows as distributions become more spread out, since the maximum duration T will have to increase. As we discount the future less, the discount factor γ approaches unity, and also requires a commensurate increase in the number of observations.

g) Empirical Comparisons:: The analytical results above give a sense of how the system will behave given a finite number of observations; however, we had to make a number of simplifying assumptions, so the bound is unlikely to be tight. These included assuming that observations were obtained by following an exploration strategy that selects each task an equal number of times. In practice, alternative exploration strategies may yield better performance than our bound would indicate. We investigated the empirical performance of these techniques in the context of the task scheduling problem by conducting experiments comparing ϵ-greedy, E^3, and an optimistic exploration strategy, which we now describe.

An ϵ-greedy exploration policy [10], [11] chooses an action uniformly at random with probability ϵ; otherwise, with probability $(1-\epsilon)$ the policy exploits its current knowledge by behaving greedily with respect to the optimal policy of the approximate model. This strategy approaches the optimal policy so long as ϵ is decayed appropriately, since at $\epsilon = 0$, the resulting policy is purely exploitive. ϵ-greedy exploration is perhaps the most commonly used exploration strategy in practice, likely due to its simplicity.

The E^3 algorithm [14] is a more principled approach to exploration for general Markov Decision Processes. It divides the state space into "known" and "unknown" states; a state is "known" when each action has been tried more than some minimum number of times. Unknown states are aggregated into a single state, but may become known as a sufficiently many observations become available in each state. A later extension, R-max, associated a reward with reaching the unknown state aggregate in order to encourage exploration [16]. Since transition distributions depend on the duration distributions of each task, which are independent of state, this additional incentive to explore becomes unnecessary: all of the states are known once we have tried each task a suitable number of times. Therefore, in the task scheduling domain, E^3 and R-max reduce to a balanced wandering strategy that simply tries each action a fixed number of times prior to choosing an exploitive strategy.

Optimism in the face of uncertainty [10], [17] is a heuristic for encouraging exploration by underestimating the cost (or equivalently, overestimating the reward) of states and actions that have been observed fewer times. This biases the controller towards taking exploratory actions while being more sensitive to past observations than ϵ-greedy exploration. We implement this strategy by choosing actions by estimating $P(t|a)$ using $O_m(a,t)/(O_m(a)+1)$; since the reward and value functions are negative, this strategy underestimates the cost of tasks that have been observed fewer times. This is equivalent to acting as though the agent received a new observation of a that resulted in a zero-cost state; this is similar in spirit to confidence interval based methods for exploration [18]. Since the number of observations $O_m(a)$ will eventually dominate the denominator, this method asymptotically approaches the exploitive policy.

We compare these exploration strategies to an policy that always exploits its current approximate model by choosing the apparent greedy action. We expect that this strategy will converge to the optimal policy in the task scheduling domain; no matter how poor our estimate of the duration of some task, eventually that task will be executed, since otherwise the cost of the states encountered would grow without bound. We measure the performance of exploration in terms of the cumulative number of mistakes as a function of the number of decisions.

For ease of exposition, we consider the two-task case. Empirically, it appears that in the two-task case behaving greedily with respect to reward is the optimal behavior. Therefore, we evaluate the number of observations needed to determine accurately the greedy policy π,

$$\pi(x) = \operatorname*{argmin}_{a\in\mathcal{A}} \left\{ \sum_t P(t|a)c(x+t\Delta_a) \right\}.$$

We expect two parameters of the task scheduling problem to dominate the mistake rate. First, the amount of overlap between task duration distributions should in part determine the difficulty of approximating the true greedy policy; for example, in many states we can choose the task with least expected cost more easily when the duration of one task always has shorter duration than the other. Second, the relative resource share for each task is also likely to influence the learning rate, since this will affect the magnitude of costs associated with running one task

relative to the other.

We compared the performance of the different exploration techniques. We compared the performance of ϵ-greedy using varying decay rates ρ, so that at decision epoch d the agent exploits with probability ρ^d. Notice that ϵ-greedy with a decay rate of 0.0 is equivalent to always exploiting our current estimate. We also examined the optimistic exploration policy and E^3 with a varying number of exploration rounds (the least-sampled task is run until every task has a sufficient number of observations, so that a round consists of running each task once); here as well, E^3 with 0 rounds of exploration is equivalent to the purely exploitive policy.

We evaluated the number of times that the approximate and true cost greedy policies differed by executing trajectories starting in the state $(0, 0)$ on randomly generated problems. Problems were generated by determining random utilization targets and task durations. The utilization target was determined by selecting integer values z_1 and z_2 uniformly at random from the range from 1 to 128 inclusive, then setting $u_i = z_i/(z_1+z_2)$. Distributions were generated by discretizing normal distributions with means selected uniformly at random from 1 to 32 inclusive and variances from 1 to 8 inclusive; each distribution was then truncated to the range 1 to 32 inclusive and normalized.

After generating an example problem, we then executed a trajectory by following each exploration policy from the initial state $(0, 0)$. Trajectories consisted of 16,384 decision epochs. At each decision epoch, we chose a task according to the exploration policy and approximate task model. We then simulated that task by emitting a duration according to its true distribution. This observation was then incorporated into the task model. We repeated this process for 1,000 problems.

h) Comparison Results:: The results of these tests are shown in Figure 2. We recorded the cumulative number of suboptimal actions taken during training; we report the mean at each decision epoch; 95% confidence intervals are shown. Figure 2(a) compares the performance of optimistic selection against the pure exploitation strategy. Figure 2(b) compares several settings of ϵ-greedy exploration with decay rates of 0, 0.1, 0.2, and 0.3; when this rate equals zero the exploration strategy is identical to pure exploitation. Figure 2(c) compares the performance of E^3 with 0, 30, 60, and 90 rounds of pure exploration prior to exploiting.

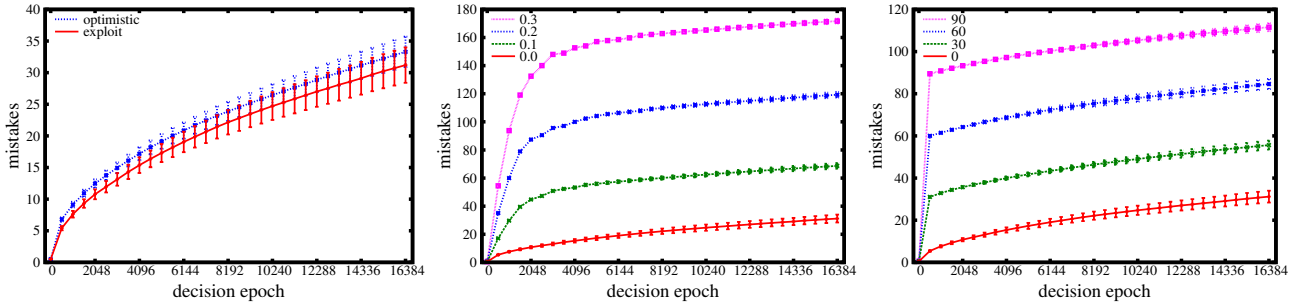
From Figure 2 we see that the exploitive policy outperforms any of these dedicated exploration strategies in the task scheduling domain. We attribute this result to a pair of factors. First, unlike the general MDP setting, learning the distribution well for each task in one state is sufficient to estimate the transition distribution well in *every* state. Therefore, we need far fewer exploratory actions in order to construct an accurate model of the environment than would be necessary in general. This reduces the importance of dedicated exploration to some degree.

The second factor is that no matter how badly we overestimate the duration of some task, we will always eventually reach a state where the exploitive policy chooses that action – eventually that task will be underutilized to such an extent that it must be run in order to reduce cost. This means that the exploitive policy will eventually dispatch each task enough times to build accurate distribution models.

It is consistent with these observations that E^3 and ϵ-greedy strategies perform better as we decrease the amount of dedicated exploration actions taken – by reducing the decay parameter to ϵ-greedy or by decreasing the number of balanced wandering steps in E^3. Optimistic exploration acts on a policy that is nearly identical to the exploitive policy except that it slightly underestimates costs, but this strategy still does not outperform the exploitive policy.

In order to relate these results to the analytic bounds given earlier in this section, keep in mind that in order to simplify the derivation, we assumed that observations were obtained by balanced wandering, and that the learner ceased to modify its estimates after the initial exploration phase. This corresponds to the E^3 strategy in our empirical results except that after the exploration stage, the mistake curve would be linearly increasing on average. Based on these results, we would expect the analytic bounds to also apply to the pure exploitation strategy.

i) Problem Parameter Interaction:: The results in Figure 2 were obtained by generating several random problems in order to control for the effects of problem parameters. We are also interested in understanding how the problem parameters interact with the mistake rate. In particular, it is of interest to know how the choice of utilization target impacts learning. We performed another set of experiments to examine this. We considered utilization targets of the form $u = (1/(1 + j), j/(1 + j))$, so that the ratio of the utilization targets is j. This lets us study the effect of giving one task a progressively larger share of the resource. For each utilization target, we generated 500 random task models as above; we ran a single trajectory on each model following a purely exploitive strategy and recorded the total number of mistakes made in the first 16,384 steps. We report the results of these experiments in Figure 3. From there we can see that it is more difficult to behave optimally when one task is given a greater share. This is a bit surprising since we expected that as the utilization became more unfair, there would be a corresponding skew in costs, making it easier to distinguish the task with better cost. We are continuing to investigate potential explanations for this behavior.



(a) Optimism (b) ϵ-greedy (c) E^3

Fig. 2. Simulation comparison of exploration techniques.

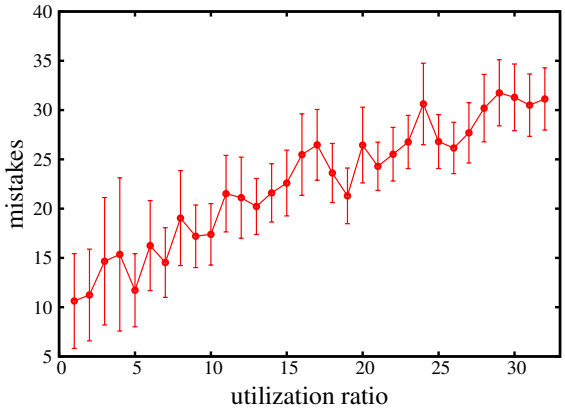


Fig. 3. Simulation with varying ratios of utilization targets.

B. Mode Selection in Multi-Mode Systems

Mode selection, as described in Section III is dependent on the selection of a value for the window size parameter. This parameter affects two notable system quantities: (1) the overall error rate for confusing modes and (2) the lag from a mode change to the detection of that mode change. In order to test the effect of window size on these system quantities, we ran a series of simulations of a simple system with two modes and one task.

To measure the effect of window size on error rate, we first generated 30 systems with randomized task duration distributions. For each system we ran 30 simulations for each of the 2 system modes and for each window size between 1 and 30. A simulation consisted of holding the system in a single mode for 4000 decision epochs, after which the percentage of epochs where the system incorrectly identified the current mode was recorded.

This raw error rate was highly dependent on the overlap of the task duration distributions in each of the system's modes. Modes whose task duration distributions had low overlap were highly distinct and thus for any window size had very low error rates. Modes whose task duration distributions overlapped greatly had comparatively high error rates. In order to generalize our results across the set of systems in our experiment, we normalized each error rate with respect to the average rate seen for that system at a window size of one. Figure 4 shows the results of this experiment: normalized error rates for each window size are shown with 95% confidence intervals. Because of our normalization all systems had an average error rate of 1 at a window size of 1. The normalized error rate for a system with window size 2 was shown to be around 40% meaning that changing from a window size of 1 to a window size of 2 resulted in, on average, a 60% reduction of mode selection error.

To measure the effect of window size on lag, we then generated 100 systems with randomized task duration distributions. For each system we ran 100 simulations for each window sizes between 1 and 30. A simulation consisted of holding the system in the first mode for a number of decision epochs equal to the system window size and then changing system modes, and running until the system correctly identified the new mode. The number of decision epochs between the mode change and the correct identification of the mode was measured as the experienced lag.

Figure 5 shows the results of this experiment: the lag, quantified in elapsed decision epochs, is shown with 95% confidence intervals for each window size. The average lag time and the 95% confidence bounds grew linearly with respect to the window size. The average lag time was roughly half the window size, which matches the intuition that it would take roughly an equal amount of observations from the new mode before the system had gathered enough evidence to justify moving to that new mode.

For both of these experiments the task duration distribution for each mode was chosen from normal distributions with means selected uniformly at random from 17 to 48 inclusive and standard deviations from 1 to 4 inclusive. Because mode selection is trivial when the distributions

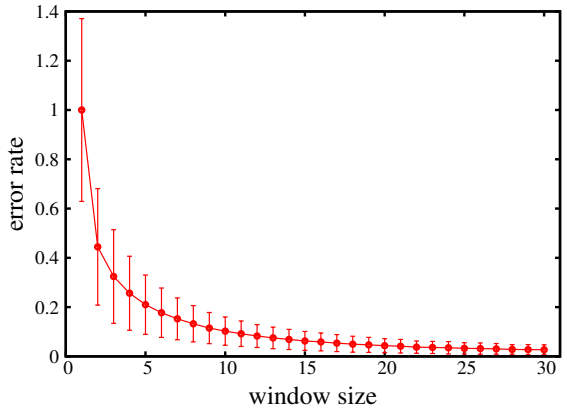


Fig. 4. Simulation results for effect of window size on normalized error rate.

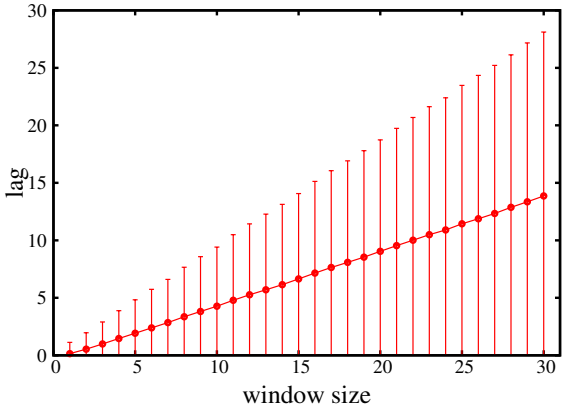


Fig. 5. Simulation results for effect of window size on lag time measured in decision epochs.

have disjoint support, we did not include any systems in our experiment for which the unnormalized error rate after 30 simulations for a window size of 1 was 0.

V. Related Work

In [19] the authors describe how to handle scenario changes in hard real-time systems. While similar, the concept of scenarios presented there differs from our concept of modes in several key ways. First, scenarios changes are planned, unlike system modes which are known only through observation. Second, the set of tasks across scenarios is transient while the set of tasks across modes is fixed.

In [20] the authors use probabilistic system models to synthesize real-time system design parameters. That work shares several interesting features with our own, including use of a similar underlying system representation. In addition to parameter estimation, the work presented in this paper incorporates the learned distributions into scheduling policy design.

Several efforts [21], [22], [23] have been made to classify real-time task behavior from observations, mostly in the context of specifying worst case execution time bounds for tasks. MABERA [21] is a tool for estimating worst-case execution time for complex real-time systems by guided simulation and exploration. Hybrid approaches [22], [23] combines measurements of execution time bounds for program segments with static analysis. In [22] these techniques are used to derive loose but safe bounds, while [23] refines these techniques in order to avoid underestimation of observed execution times. These techniques are not directly applicable to our extended system model, in which task characterization must be learned fully online, and not only WCET, but the entire task duration distribution must be considered.

Relaxing the assumption that we have complete knowledge of the system model leads to questions about balancing exploration and exploitation. Heuristic strategies, such as ϵ-greedy exploration [10], [11], and philosophies like optimism in the face of uncertainty [10], [17], [24] are well-established and commonly used methods to encourage exploration. Fiechter [25] and Kearns and Singh [14] were among the first to study this problem from a computational learning theory standpoint, providing finite-sample guarantees for learning performance; the latter proposed the E^3 algorithm and provided a PAC-bound on its sample complexity. Brafman and Tennenholz [16] proposed R-max, an extension of E^3 for *stochastic games*, a generalization of MDPs. Sample complexity results for these algorithms are restricted to finite state spaces; Kakade [15] observed that the complexity scales polynomially with the number of parameters of the transition model.

VI. Conclusions and Future Work

The techniques presented in this paper constitute important advances in the state of the art in scheduling policy design for open soft real-time systems. By expanding the foundations established in our previous work to include system actions whose execution time distributions (1) only can be obtained through run-time observation or (2) have modal structure, these contributions have brought us closer to the long-term goal of our research: placing all system actions under the control of a common scheduling policy, even in open soft real-time systems operating in highly variable environments.

To realize that goal fully, however, further open research problems need to be pursued as future work. The first open research problem we plan to investigate is how online learning of scheduling policies can be achieved when the probabilities of switching between modes are arbitrary, which has a significant impact on the notion of optimality of a policy and on how policies should be constructed. In particular, we intend to examine how the structure of the transitions between modes affects the design of optimal scheduling policies, along with other challenges

that arise in the absence of information about both the mode structure and the behavior of tasks within modes. While we have addressed these issues separately in this paper, considering these problems together will allow us to apply our techniques to an even more general class of open soft real-time systems.

REFERENCES

[1] T. Tidwell, R. Glaubius, C. Gill, and W. D. Smart, "Scheduling for Reliable Execution in Autonomic Systems," in 5^{th} *International Conference on Autonomic and Trusted Computing (ATC '08)*, Oslo, Norway, Jun. 2008.

[2] R. Glaubius, T. Tidwell, W. D. Smart, and C. Gill, "Scheduling Design and Verification for Open Soft Real-time Systems," in 29^{th} *IEEE Real-time Systems Symposium*, Barcelona, Spain, Nov. 2008.

[3] R. Glaubius, T. Tidwell, C. Gill, and W. D. Smart, "Scheduling policy design for autonomic systems," *International Journal of Autonomous and Adaptive Communications Systems*, vol. 2, no. 3, pp. 276–296, To Appear 2009.

[4] C. Liu and J. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-time Environment," *JACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[5] T. Tidwell, C. Gill, and V. Subramonian, "Scheduling induced bounds and the verification of preemptive real-time systems," Department of Computer Science and Engineering, Washington University in St.Louis, Tech. Rep. WUCSE-2007-34, 2007.

[6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Interscience, 1994.

[7] J. Rust, "Numerical Dynamic Programming in Economics," in *Handbook of Computational Economics*, H. M. Amman, D. A. Kendrick, and J. Rust, Eds. Elsevier, 1996, vol. 1, ch. 14, pp. 619–729.

[8] M. J. Kearns and S. P. Singh, "Finite-sample convergence rates for q-learning and indirect algorithms," in *Advances in Neural Information Processing Systems*, vol. 11. MIT Press, 1999, pp. 996–1002.

[9] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[10] L. P. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[12] L. G. Valiant, "A theory of the learnable," in *STOC '84: Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*. ACM, 1984, pp. 436–445.

[13] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[14] M. J. Kearns and S. P. Singh, "Near-optimal reinforcement learning in polynomial time," in *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 1998, pp. 260–268.

[15] S. M. Kakade, "On the sample complexity of reinforcement learning," Ph.D. dissertation, Gatsby Computational Neuroscience Unit, University College London, London, UK, 2003.

[16] R. I. Brafman and M. Tennenholtz, "R-MAX – a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2003.

[17] E. Even-Dar and Y. Mansour, "Convergence of optimistic and incremental q-learning," in *Advances in Neural Information Processing Systems*, vol. 13, 2001, pp. 1499–1506.

[18] A. L. Strehl and M. L. Littman, "A theoretical analysis of model-based interval estimation," in *ICML '05: Proceedings of the 22nd International Conference on Machine Learning*. New York, NY, USA: ACM, 2005, pp. 856–863.

[19] R. Henia and R. Ernst, "Scenario Aware Analysis for Complex Event Models and Distributed Systems," in 28^{th} *IEEE International Real-Time Systems Symposium*, Tuscon, AZ, Dec. 2007.

[20] T. Han, J.-P. Katoen, and A. Mereacre, "Approximate Parameter Synthesis for Probablistic Time-Bounded Reachability," in 29^{th} *IEEE International Real-Time Systems Symposium*, Barcelona, Spain, Nov. 2008.

[21] J. Kraft, Y. Lu, C. Norström, and A. Wall, "A Metaheuristic Approach for Best Effort Timing Analysis targeting Complex Legacy Real-Time Systems," in 14^{th} *IEEE Real-Time and Embedded Technology and Applications Symposium*, St. Louis, MO, Apr. 2008.

[22] J. A. Colmenares, C. Im, K. H. K. Kim, R. Klefstad, and C.-D. Lim, "Measurement Techniques in a Hybrid Approach for Deriving Tight Execution-time Bounds of Program Segments in Fully-featured Processors," in 14^{th} *IEEE Real-Time and Embedded Technology and Applications Symposium*, St. Louis, MO, Apr. 2008.

[23] B. Lisper and M. Santos, "Model Identification for WCET Analysis," in 15^{th} *IEEE Real-Time and Embedded Technology and Applications Symposium*, San Francisco, CA, Apr. 2009.

[24] I. Szita and A. Lőrincz, "The many faces optimism: a unifying approach," in *ICML '08: Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008, pp. 1048–1055.

[25] C.-N. Fiechter, "Expected mistake bound model for on-line reinforcement learning," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 116–124.

APPENDIX

Proof of Proposition 1:

Lemma 1. *For any state x, task a, and duration t,*

$$|c(x + t\Delta_a) - c(x)| \leq tc(\Delta_a).$$

Proof: The claim follows from the triangle inequality, since $c(y)$ is the l_1-norm between y and $\tau(y)u$. ■

Lemma 2. *For any state x, task a, and duration t,*

$$|V(x + t\Delta_a) - V(x)| \leq tc(\Delta_a)/(1 - \gamma).$$

Proof: Note that

$$|V(x + t\Delta_a) - V(x)| \leq \max_b |Q(x + t\Delta_a, b) - Q(x, b)|.$$

By expanding Q and Q_m according to Equation 4, then grouping cost and value terms and applying the inequality $|u - v| \leq |u| + |v|$, we get the inequality

$$\begin{aligned} & |Q(x + t\Delta_a, b) - Q(x, b)| \\ \leq & \sum_s P(s|b) \left|c(x + s\Delta_b) - c(x + t\Delta_a + s\Delta_b)\right| \\ & + \gamma \sum_s P(s|b) \left|V(x + t\Delta_a + s\Delta_b) - V(x + s\Delta_b)\right|. \end{aligned}$$

The first term simplifies to $tc(\Delta_a)$ by applying Lemma 1. The absolute value in the second term has the same form as our initial condition, so recursively applying this same argument yields

$$|V(x + t\Delta_a) - V(x)| \leq \sum_{j=0}^{\infty} \gamma^j tc(\Delta_a) = tc(\Delta_a)/(1 - \gamma).$$

■

Lemma 3. *For any real-valued function f, if $|f(x + t\Delta_a) - f(x)| \leq \beta t$ for some constant $\beta \geq 0$ and every state x, task a, and duration t, then*

$$\left|\sum_t [P_m(t|a) - P(t|a)] f(x + t\Delta_a)\right| \leq \beta \sum_t |P_m(t|a) - P(t|a)|\, t$$

Proof: This follows by decomposing f into $f(x)$ and $\pm\beta t$; since $\sum_t [P_m(t|a) - P(t|a)] = 0$, the term involving $f(x)$ vanishes. ∎

Let C be the expected cost, with C_m defined similarly:

$$C(x, a) = -\sum_t P(t|a) c(x + t\Delta_a); \tag{7}$$

we'll also denote the expected successor values by

$$\mathbf{E}_P \{V(x + t\Delta_a)\} = \sum_t P(t|a) V(x + t\Delta_a). \tag{8}$$

Using Equations 7 and 8, we can write the definition of Q more concisely as

$$Q(x, a) = C(x, a) + \gamma \mathbf{E}_P\{V(x + t\Delta_a)\}.$$

We now use these definitions and lemmas to demonstrate our central claim. Let (x, a) be a state-task pair. Then

$$\begin{aligned} & |Q_m(x, a) - Q(x, a)| \\ \leq\ & |C_m(x, a) - C(x, a)| \\ & + \gamma \left|\mathbf{E}_{P_m}\{V_m(x + t\Delta_a)\} - \mathbf{E}_P\{V(x + t\Delta_a)\}\right| \end{aligned} \tag{9}$$

We can simplify the error in approximating expected cost, $|C_m(x, a) - C(x, a)|$, using Lemmas 1 and 3,

$$\begin{aligned} |C_m(x, a) - C(x, a)| &\leq c(\Delta_a) \sum_t |P_m(t|a) - P(t|a)|\, t \\ &< \lambda T(T - 1); \end{aligned}$$

this last inequality follows because $c(\Delta_a) < 2$ and $\sum_t |P(t|a) - P_m(t|a)|\, t \leq \lambda T(T - 1)/2$ when the maximum duration T is finite. The difference in expected future values can be decomposed into components corresponding to approximation errors on the future state value and on the duration distributions,

$$\begin{aligned} & \left|\mathbf{E}_{P_m}\{V_m(x + t\Delta_a)\} - \mathbf{E}_P\{V(x + t\Delta_a)\}\right| \\ \leq\ & \sum_t P_m(t|a) \left|V_m(x + t\Delta_a) - V(x + t\Delta_a)\right| \\ & + \left|\sum_t [P_m(t|a) - P(t|a)] V(x + t\Delta_a)\right|. \end{aligned}$$

The second term here simplifies by applying Lemmas 2 and 3,

$$\left|\sum_t [P_m(t|a) - P(t|a)] V(x + t\Delta_a)\right| \leq \lambda T(T-1)/(1-\gamma).$$

We can then plug these terms into Equation 9,

$$\begin{aligned} & |Q_m(x, a) - Q(x, a)| \\ \leq\ & \lambda T(T - 1) + \gamma \lambda T(T - 1)/(1 - \gamma) \\ & + \sum_t P_m(t|a) \left|V_m(x + t\Delta_a) - V(x + t\Delta_a)\right| \end{aligned}$$

since we can bound $|V_m(x + t\Delta_a) - V(x + t\Delta_a)|$ by $\max_b |Q_m(x + t\Delta_a, b) - Q(x + t\Delta_a, b)|$ and substitute according to 9, we obtain the bound

$$\begin{aligned} |Q_m(x, a) - Q(x, a)| &\leq \sum_{j=0}^{\infty} \gamma^j \lambda T(T - 1)/(1 - \gamma) \\ &\leq \lambda T(T - 1)/(1 - \gamma)^2, \end{aligned}$$

completing the proof.