Report Number: wucse-2009-71

2009

Scalable Scheduling Policy Design for Open Soft Real-Time Systems

Robert Glaubius, Terry Tidewell, Braden Sidoti, David Pilla, Justin Meden, Christopher Gill, and William D. Smart

Open soft real-time systems, such as mobile robots, must respond adaptively to varying operating conditions, while balancing the need to perform multiple mission specific tasks against the requirement that those tasks complete in a timely manner. Setting and enforcing a utilization target for shared resources is a key mechanism for achieving this behavior. However, because of the uncertainty and non-preemptability of some tasks, key assumptions of classical scheduling approaches do not hold. In previous work we presented foundational methods for generating task scheduling policies to enforce proportional resource utilization for open soft real-time systems with these properties. However, these... Read complete abstract on page 2.

Scalable Scheduling Policy Design for Open Soft Real-Time Systems

Robert Glaubius, Terry Tidewell, Braden Sidoti, David Pilla, Justin Meden, Christopher Gill, and William D. Smart

Complete Abstract:

Open soft real-time systems, such as mobile robots, must respond adaptively to varying operating conditions, while balancing the need to perform multiple mission specific tasks against the requirement that those tasks complete in a timely manner. Setting and enforcing a utilization target for shared resources is a key mechanism for achieving this behavior. However, because of the uncertainty and non-preemptability of some tasks, key assumptions of classical scheduling approaches do not hold. In previous work we presented foundational methods for generating task scheduling policies to enforce proportional resource utilization for open soft real-time systems with these properties. However, these methods scale exponentially in the number of tasks, limiting their practical applicability. In this paper, we present a novel parameterized scheduling policy that scales our technique to a much wider range of systems. These policies can represent geometric features of the scheduling policies produced by our earlier methods, but only require a number of parameters that is quadratic in the number of tasks. We provide empirical evidence that the best of these policies are competitive with exact solution methods in small problems, and significantly outperform heuristic methods in larger ones.

2009-71

Scalable Scheduling Policy Design for Open Soft Real-Time Systems

Authors: Robert Glaubius, Terry Tidwell
Braden Sidoti, David Pilla, Justin Meden
Christopher Gill, William D. Smart

Abstract: Open soft real-time systems, such as mobile robots, must respond adaptively to varying operating conditions, while balancing the need to perform multiple mission specific tasks against the requirement that those tasks complete in a timely manner. Setting and enforcing a utilization target for shared resources is a key mechanism for achieving this behavior. However, because of the uncertainty and non-preemptability of some tasks, key assumptions of classical scheduling approaches do not hold. In previous work we presented foundational methods for generating task scheduling policies to enforce proportional resource utilization for open soft real-time systems with these properties. However, these methods scale exponentially in the number of tasks, limiting their practical applicability.

In this paper, we present a novel parameterized scheduling policy that scales our technique to a much wider range of systems. These policies can represent geometric features of the scheduling policies produced by our earlier methods, but only require a number of parameters that is quadratic in the number of tasks. We provide empirical evidence that the best of these policies are competitive with exact solution methods in small problems, and significantly outperform heuristic methods in larger ones.

Type of Report: Other

Scalable Scheduling Policy Design for Open Soft Real-Time Systems

Robert Glaubius, Terry Tidwell, Braden Sidoti, David Pilla, Justin Meden,
Christopher Gill, and William D. Smart
Department of Computer Science and Engineering
Washington University in St. Louis
Email: {rlg1@cse,ttidwell@cse,bts1@cec,dgp3@cec,jem6@cec,cdgill@cse,wds@cse}.wustl.edu

***Abstract*—Open soft real-time systems, such as mobile robots, must respond adaptively to varying operating conditions, while balancing the need to perform multiple mission specific tasks against the requirement that those tasks complete in a timely manner. Setting and enforcing a utilization target for shared resources is a key mechanism for achieving this behavior. However, because of the uncertainty and non-preemptability of some tasks, key assumptions of classical scheduling approaches do not hold. In previous work we presented foundational methods for generating task scheduling policies to enforce proportional resource utilization for open soft real-time systems with these properties. However, these methods scale exponentially in the number of tasks, limiting their practical applicability.**

In this paper, we present a novel parameterized scheduling policy that scales our technique to a much wider range of systems. These policies can represent geometric features of the scheduling policies produced by our earlier methods, but only require a number of parameters that is quadratic in the number of tasks. We provide empirical evidence that the best of these policies are competitive with exact solution methods in small problems, and significantly outperform heuristic methods in larger ones.

I. Introduction

Open soft real-time systems, such as mobile robots, must respond adaptively to varying operating conditions. Effective operation of these systems requires that sensing and actuation tasks are performed in a timely manner. In addition, execution of mission specific tasks such as imaging a room must be balanced against the need to perform more general tasks such as obstacle avoidance. Setting and enforcing a utilization target for shared resources is a key mechanism for striking this balance while ensuring timely execution of tasks.

Classical scheduling approaches are inapplicable to important tasks in the domains we consider. There are two reasons for this. First, some tasks are not efficiently preemptable; for instance an actuation task may involve moving a shared physical resource, such as a robotic arm or pan-tilt unit. Restoring the actuator state after a preemption would be equivalent to just restarting that task. For this reason, once a job of a task acquires the resource, it should retain it until it completes.

Second, the duration for which a task holds the resource may be stochastic, though we assume it obeys some known underlying distribution. Behaving optimally under these conditions requires that the system exploit this uncertainty by anticipating common events while reacting to early availability of the resource and hedging against delays.

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

In previous work [1] we presented methods for generating task scheduling policies for open soft real-time systems with these properties. However, the inherent complexity of generating these policies limits the usefulness of that approach to problem instances with a small number of tasks. In this paper we present a parameterized policy approach that allows us to compute policies for a significantly wider range of problem instances.

The paper is organized as follows: In Sections II and III we summarize our system and task models and our solution approach based on solving Markov decision processes (MDPs) as in [1]. As that approach can not directly solve problems involving a large number of tasks, in Section IV we propose alternative strategies for addressing these limitations. In Section V we introduce a new parameterized policy approach, which is the main contribution of this work. In Section VI we discuss experiments comparing our parameterized policy approach to heuristic policies and for small numbers of tasks to policies derived by solving our MDP model directly. Finally, in Section VII we present our conclusions and future work.

II. System Model

In previous work [1], [2], we proposed an abstract system model in which n tasks $(T_i)_{i=1}^n$ require mutually exclusive use of a single common resource. Each task T_i consists of an infinite sequences of jobs $(J_{i,j})_{j=0}^\infty$; $J_{i,0}$ is available at time 0, and each job $J_{i,(j+1)}$ becomes available immediately upon completion of $J_{i,j}$. Jobs can not be preempted; whenever a job is granted the resource, it occupies that resource for a finite, bounded, stochastic duration.

We make two simplifying assumptions regarding the distribution of each job's durations:

- **(A1)** Inter-task durations are independently distributed.
- **(A2)** Intra-task durations are independently and identically distributed.

When Assumption (A1) holds, the duration of job $J_{i,j}$ always obeys the same distribution regardless what jobs preceded it. This means that we do not need to know the system history in order to predict the behavior of a particular job. When

Assumption (A2) holds, consecutive jobs of the same task obey the same distribution. Thus, every task T_i has a *duration distribution* P_i from which the duration of every job of T_i is drawn.

In addition to the assumptions stated above, we require that each duration distribution has bounded support on the positive integers: that is, every task T_i has a worst-case execution time W_i such that $\sum_{t=1}^{W_i} P_i(t) = 1$.

Our goal is to schedule jobs dynamically in order to preserve *temporal isolation* [3] among tasks. We specify some target utilization u_i for each task that describes its intended resource share at any temporal resolution. More specifically, let $x_i(t)$ denote the number of quanta during which task T_i held the resource in the interval $[0, t)$. Our objective is to keep

$$|(t' - t)u_i - (x_i(t') - x_i(t))|$$

as small as possible over *every* time interval $[t, t')$. We require that each task's utilization target u_i is rational, and further, that the resource is completely divided among all tasks, so that $\sum_{i=1}^{n} u_i = 1$.

This objective is similar to the Pfair scheduling criterion [4], which enforces proportional fairness of tasks by requiring that each task is always within a fixed quantum of its weight. Pfair is concerned with the scheduling of deterministic, periodic or sporadic task systems [5], so the weight of a task is the ratio of its duration, abstracted as its worst-case execution time W_i, to its period p_i. Thus the Pfair condition can be stated as

$$|tW_i/p_i - x_i(t)| < 1.$$

In general, maintaining this criterion for tasks with duration longer than a single quantum requires preemption. For resources, such as actuators, that do not allow preemption, we must instead focus on minimizing this deviation from the target utilization at coarser temporal resolution. Since in our system model durations are stochastic and tasks are aperiodic, we allow utilization targets to be specified independently of the duration and period.

III. MDP Model Formulation

In previous work we proposed a stochastic dynamic programming approach to modeling and solving the scheduling problem described in the previous section [1], [6]. This was achieved by modeling the problem as a Markov decision process (MDP) [7]. An MDP has a set of states $\mathcal{X}$, a set of actions $\mathcal{A}$, a transition system P, and a cost function C. At each *decision epoch* $k \in \mathbb{N}$, a controller observes the current state x_k of the MDP and selects an action i_k from $\mathcal{A}$. Then the MDP transitions to state x_{k+1} according to the conditional distribution $P(x_{k+1} = y | x_k = x, i_k = i)$ (abbreviated as $P(y|x, i)$), and accrues cost $c_k = C(x_{k+1})$.

A solution to an MDP is a policy π that maps states to actions; given a discount factor $\gamma \in [0, 1)$, the *value* of a policy, written V^π, is the expected sum of long-term, discounted costs obtained while following that policy:

$$V^\pi(x) = \mathbf{E}\left\{ \sum_{k=0}^{\infty} \gamma^k c_k \middle| x_0 = x, i_k = \pi(x_k) \right\}. \quad (1)$$

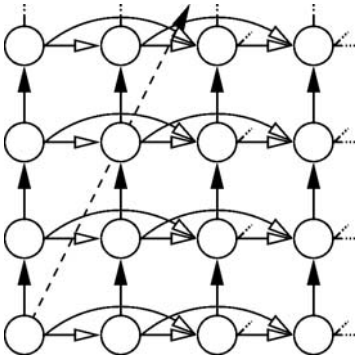


Fig. 1. The utilization state model for a two-task problem instance with utilization target $\mathbf{u} = (1, 2)^\top/3$. Task one (grey, open arrowheads) stochastically transitions to the right, while task two (black, closed arrowheads) deterministically transitions upward. The dashed ray corresponds to the utilization target.

We assume costs are never positive; thus an optimal policy π^* maximizes V^{π^*} in every state. It is well-known that if any optimal policy exists, then a deterministic optimal policy must exist [7]. This is satisfied, for example, when there are only finitely many states and actions, and costs are bounded.

We modeled this task scheduling problem as an MDP over a set of *utilization states* $\mathcal{X} = \mathbb{N}^n$ with each action i corresponding to the decision to dispatch task T_i. Each state $\mathbf{x}$ is an n-vector $\mathbf{x} = (x_1, \ldots, x_n)^\top$ such that component x_i is the total number of quanta during which task T_i occupied the shared resource since system initialization at decision epoch $k = 0$. We denote the total elapsed time in state $\mathbf{x}$ by the function

$$\tau(\mathbf{x}) = \sum_{i=1}^{n} x_i. \quad (2)$$

Transitions in this MDP are determined according to the duration distribution of each task, so that

$$P(\mathbf{y}|\mathbf{x}, i) = \begin{cases} P_i(t) & \mathbf{y} = \mathbf{x} + t\Delta_i \\ 0 & otherwise \end{cases} \quad (3)$$

where Δ_i is a column vector of all zeroes except that component i is equal to one. We define the cost function in terms of the target utilization criterion

$$C(\mathbf{x}) = -\sum_{i=1}^{n} |x_i - \tau(\mathbf{x})u_i|, \quad (4)$$

which encourages policies that maintain the MDP state near the target share for each task over time. Since costs are never positive, better policies have value closer to 0 in this model.

Figure 1 illustrates the utilization state model for an example problem with two tasks and a target utilization $\mathbf{u} = (1, 2)^\top/3$ – that is, task T_1 should receive $1/3$ of the processor, and task T_2 should receive the rest. The target utilization defines a *target utilization ray* $\{\lambda\mathbf{u} : \lambda \geq 0\}$. When the components of $\mathbf{u}$ are rational, this ray will periodically pass through utilization states. This can be seen in Figure 1, where the utilization ray (the dashed arrow) passes through both $(0, 0)$ and $(1, 2)$; if

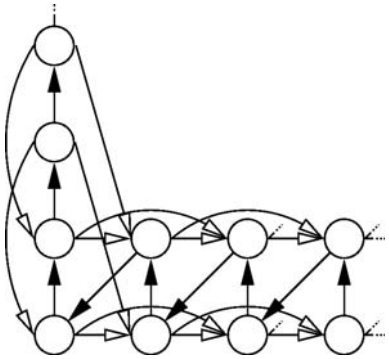


Fig. 2. The wrapped state model for the problem instance in Figure 1.

we continued the figure, the utilization ray would pass through every integer multiple of $(1, 2)$. Every state on this ray has zero cost, while states off this ray have non-zero cost. Any pair of states with the same displacement from the target utilization ray have equal cost.

We have demonstrated that there is an optimal policy for this MDP [1], and that we can formulate this MDP more compactly by taking advantage of a periodicity condition over the state space. Let $\mathbf{u}$ be the vector of utilization targets for each task. Then we can define equivalence classes over states,

$$\{\mathbf{x} + \lambda\mathbf{u} \in \mathcal{X} : \lambda \in \mathbb{R}\}.$$

We say that a function $f : \mathcal{X} \to \mathbb{R}$ is periodic iff $f(\mathbf{x} + \lambda\mathbf{u}) = f(\mathbf{x})$ for any state $\mathbf{x}$; in other words, f is periodic iff it is homogenous over each equivalence class. For example, the cost function is homogenous. Further, because each task's utilization target u_i is in the interval $(0, 1)$, the probability of transitioning between equivalence classes is homogenous. That is,

$$P(\mathbf{x} + \lambda\mathbf{u} + t\Delta_i|\mathbf{x} + \lambda\mathbf{u}, i) = P_i(t) = P(\mathbf{x} + t\Delta_i|\mathbf{x}, i);$$

less formally, if we can move from $\mathbf{x}$ to $\mathbf{y}$ with probability p, then we have the same probability of transitioning from $\mathbf{x} + \lambda\mathbf{u}$ to $\mathbf{y} + \lambda\mathbf{u}$ for any λ.

We are able to infer that any periodic policy has periodic value, and that there is an optimal, deterministic, periodic policy for this problem. Thus, using a stochastic bisimulation-style argument [8], [9], we are able to obtain a more compact MDP representation of the scheduling problem that represents each equivalence class as a state. We refer to this compact model as a *wrapped state* model, since we can interpret it as wrapping the state space into a tube so that all of the states that share an equivalence class map to a single point. Figure 2 demonstrates this wrapped state model using the example from Figure 1. We have replaced all states that share an equivalence class with a single exemplar state, and adjusted the transition graph by mapping transitions between non-exemplars to their corresponding exemplar states.

This wrapped model reduction removes infinitely many states from the utilization state model. Although it still has infinitely many states (since every state $t\Delta_i$ for $t \in \mathbb{N}$ is in a different equivalence class), for any bound there are only finitely many states with cost that exceed that bound [6]. Intuitively, a good scheduling policy will only visit states with costs relatively close to zero, assuming we initialize the system in the state $\mathbf{x} = \mathbf{0}$. These policies thus can only visit finitely many states. Using this intuition, we have proposed methods for approximating the optimal policy using a bounded subset of the wrapped model state space [1], and an algorithm that iteratively constructs minimal state spaces necessary to evaluate and improve policies that are guaranteed to visit only finitely many states [6].

Although the wrapped state model provides a foundation for establishing *finite state approximation* methods, those algorithms rely on enumerating a portion of the state space that grows exponentially with the number of tasks. This restricts their usefulness to problem instances with only a few tasks. In Sections IV and V we examine this scalability issue in detail and propose a parameterized class of scheduling policies, called *conic policies*, that can be represented and evaluated efficiently even for large numbers of tasks.

IV. Scaling up MDP-based Scheduling

In general, techniques for computing the optimal policy of a discrete MDP rely on explicit enumeration of the state space as a lookup table [7]. The wrapping approach described in Section III performs state aggregation by reducing each equivalence class of states to a single exemplar state while retaining the optimal policy. State aggregation is a useful tool for decreasing the representational and computational cost. However, the amount of aggregation that can be achieved without sacrificing optimality is fundamentally limited [10]. It appears unlikely that we can perform any additional aggregation (as defined in Sections II and III) while preserving optimality.

Three approaches appear promising to increase the scalability of MDP-based scheduling; we discuss two of them in this paper. The first is to use heuristic scheduling policies that make decisions based on short-term statistics of task behavior. Though efficient to compute, these heuristic policies may perform poorly, as our evaluations show in Section VI. Another approach is to perform aggregation at a higher level of abstraction by grouping tasks together. While this method is beyond the scope of the current work, it represents an important direction for future research. The third approach, to develop a class of policies that can be represented compactly using a set of parameters that grows polynomially with the number of tasks, is the main contribution of this work.

In problems with a small number of tasks, we can compare the performance of parameterized scheduling policies to those obtained using the explicit state enumeration techniques of the previous section. When the number of tasks grows larger, however, we must instead compare the parameterized policies to heuristic scheduling policies, since approximating the optimal policy becomes intractable.

We consider two heuristic scheduling policies for comparison: a utilization-based scheduling policy that always

dispatches the task that is farthest behind its target utilization, and a greedy policy that chooses the action that yields the best immediate cost in expectation.

In a utilization state (or wrapped state) $\mathbf{x}$, we say task T_i is *underutilized* iff $x_i < \tau(\mathbf{x})u_i$ and is *overutilized* iff $x_i > \tau(\mathbf{x})u_i$. The task is *on time* if it is neither underutilized nor overutilized. The utilization-based policy π_u always runs the most underutilized task,

$$\pi_u(\mathbf{x}) = \underset{i=1,\ldots,n}{\operatorname{argmin}}\{\tau(\mathbf{x})u_i - x_i\}. \tag{5}$$

The greedy policy π_g myopically chooses the action that puts the system closest to the target utilization in expectation. This is equivalent to choosing the task that gives the best expected cost from $\mathbf{x}$, and is defined

$$\begin{aligned}\pi_g(\mathbf{x}) &= \underset{i=1,\ldots,n}{\operatorname{argmax}}\{\mathbf{E}_{t\sim P_i}\{C(\mathbf{x}+t\Delta_i)\}\} \\ &= \underset{i=1,\ldots,n}{\operatorname{argmax}}\left\{\sum_{t=1}^{W_i} P_i(t)C(\mathbf{x}+t\Delta_i)\right\}.\end{aligned} \tag{6}$$

V. Conic Scheduling Policies

The main contribution of this paper is the development of a class of compactly parameterized policies for the utilization state model. This approach circumvents the major limitation of the MDP formulations, the explicit enumeration of exponentially growing state spaces, while retaining the ability to evaluate and compare policies in the value-based framework described in Section III. In this section, we describe a family of *conic* policies that can be defined using just $\Theta(n^2)$ parameters, where n is the number of tasks. This class contains stable scheduling policies that tend to outperform the heuristic approaches described in Section IV in problems with large numbers of tasks, and for problems with small numbers of tasks, are competitive with scheduling policies found using the finite state approximation techniques.

One well-known approach for scaling up stochastic planning and reinforcement learning to high-dimensional state spaces is to restrict solutions to some compactly parameterizable policy class [11], [12]. These methods trade off optimality guarantees in favor of good practical performance, and so are able to address problems that are much larger than can be solved using methods that require state enumeration. Choosing an appropriate policy class is admittedly more art than science, and requires understanding the application domain and the properties of good policies. With this in mind, we first illustrate examples of scheduling policy behavior observed using the finite state approximation techniques described in Section III.

Figure 3 shows an approximation to the optimal scheduling policy for a problem instance with two tasks, restricted to states near the initial state $\mathbf{x} = \mathbf{0}$ (lower left corner). Each point denotes a state in the wrapped state space. The target utilization ray is shown as a dashed ray labeled $\lambda\mathbf{u}$, and corresponds to a target utilization of $\mathbf{u} = (7,5)^\top/12$. Each task has a different duration distribution supported on the interval $(0, 8]$. Task T_1 advances along the horizontal axis,

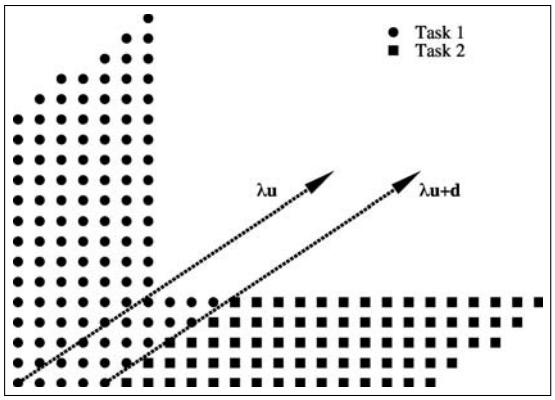


Fig. 3. The scheduling policy for this two-task problem can be defined by partitioning the state space along a ray parallel to the utilization ray $\lambda\mathbf{u}$.

and task T_2 advances along the vertical axis. The scheduling policy selects task T_1 in states denoted by closed circles, and task T_2 in those denoted by closed squares. Notice that the *decision boundary* – the surface separating regions of state space where the policy is homogenous – can be described using a ray parallel to the utilization ray. This is illustrated by the dashed ray labeled $\lambda\mathbf{u} + \mathbf{d}$. We will describe the offset $\mathbf{d}$ below.

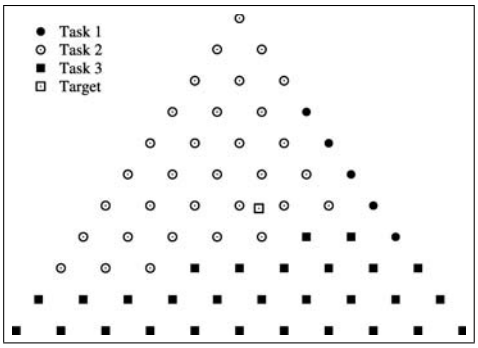
It is more difficult to illustrate the policy in three-task problems, since the state space is three dimensional. To establish an intuition for what is occurring in three-space, consider the set of utilization states that the system may reach after exactly t time quanta have elapsed:

$$H_t = \{\mathbf{x} \in \mathcal{X} : \tau(\mathbf{x}) = t\}.$$

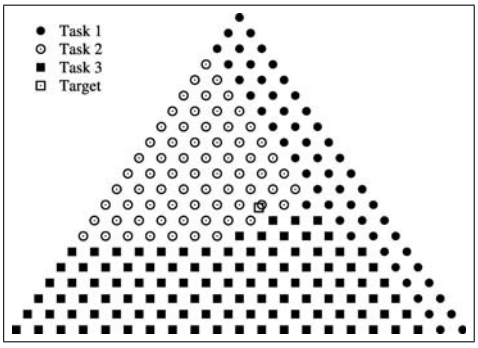
We call H_t a *time horizon*, and it consists of all of the utilization states (i.e., integral points) in a $(n-1)$-simplex with vertices $\{t\Delta_i : i = 1,\ldots,n\}$. In two dimensions, H_t is the set of states that lie on the line segment joining $(t, 0)$ and $(0, t)$. In three dimensions, H_t is the set of states contained in the equilateral triangle with vertices $(t, 0, 0)$, $(0, t, 0)$, and $(0, 0, t)$. The target utilization at time t is $t\mathbf{u}$, and is the *ideal point* that the system should be near in H_t.

Figure 4 illustrates an approximation to the optimal scheduling policy for a problem instance with three tasks. The policy is illustrated by plotting it at three different time horizons, H_{10}, H_{20}, and H_{30}. The target utilization is $\mathbf{u} = (6,8,9)^\top/23$, and corresponds to a point in each horizon, shown as an open box. Tasks are non-identical, but each has duration supported on the interval $(0, 8]$. The policy executes T_1 in closed circle states, T_2 in open circle states, and T_3 in closed square states.

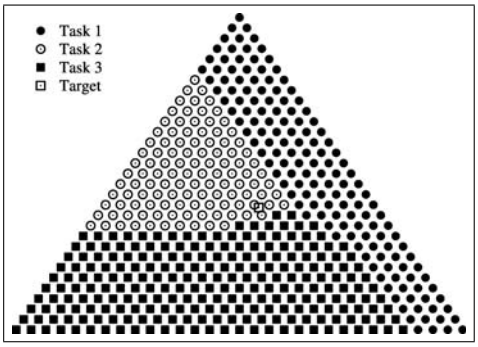
As in the two task case, the policy partitions each time horizon into one homogenous region for each task. This appears to be representative behavior. Together with the observation of periodicity from Section III, this gives us two criteria for designing a parametric class of scheduling policies: (1) the policy should be periodic, so that it chooses the same action at $\mathbf{x}$ and every utilization state along $\mathbf{x}+\lambda\mathbf{u}$; and (2) the policy should divide each time horizon into n homogenous regions. This now leads to our formulation of conic policies.



(a) H_{10}



(b) H_{20}



(c) H_{30}

Fig. 4. Near-optimal scheduling policies for a three task problem, shown at time horizons H_{10}, H_{20}, and H_{30}. The leftmost state in each is $(t, 0, 0)$, the rightmost is $(0, t, 0)$, and the topmost is $(0, 0, t)$.

A. Conic Policies

We define two types of parameters for conic policies. The first n parameters describe a *decision offset* vector $\mathbf{d}$; this is a vector perpendicular to $\mathbf{1}$, the vector of all ones, (i.e., $\mathbf{d}$ is parallel to each time horizon) that roots the partition relative to the utilization target.

The remaining n^2 parameters define a collection of *action vectors* $\mathbf{A} = [\mathbf{a}_1 \ \ldots \ \mathbf{a}_n]$ that are used to determine the regions where each action is taken. For each task T_i, the action vector $\mathbf{a}_i$ is an n-vector perpendicular[1] to $\mathbf{1}$. We use the decision offset and action vectors to partition each time horizon into homogenous regions as follows.

Informally, a conic policy selects a task to dispatch in state $\mathbf{x}$ by determining which action vector most points towards $\mathbf{x}$ from the decision offset. More formally, $\tau(\mathbf{x})\mathbf{u}$ is the ideal utilization point at time $\tau(\mathbf{x})$. We root the policy decision boundaries on the *decision ray* $\lambda\mathbf{u}+\mathbf{d}$ at that time, $\tau(\mathbf{x})\mathbf{u}+\mathbf{d}$. Let

$$\mathbf{z}(\mathbf{x}) = \mathbf{x} - \tau(\mathbf{x})\mathbf{u} - \mathbf{d} \tag{7}$$

be the displacement vector from the offset of the ideal utilization point to $\mathbf{x}$. Then we choose to run the task T_i if its action vector is well-aligned with $\mathbf{z}(\mathbf{x})$ – that is, if the *response* $\mathbf{a}_i^\top \mathbf{z}(\mathbf{x})$ is maximal among all action vectors. We state this formally in the following definition.

Definition 1. *The conic policy $\pi(\cdot; \mathbf{d}, \mathbf{A})$ with decision offset $\mathbf{d}$ and a matrix of action vectors $\mathbf{A}$ chooses actions in each utilization state $\mathbf{x}$ according to*

$$\pi(\mathbf{x}; \mathbf{d}, \mathbf{A}) \in \underset{i=1,\ldots,n}{\operatorname{argmax}}\{\mathbf{a}_i^\top \mathbf{z}(\mathbf{x})\}$$

Notice that it is possible for multiple action vectors to have the same response at $\mathbf{x}$. We recommend breaking ties uniformly at random in this case. Under this tie breaking convention, the conic policy degenerates to a uniform random scheduling policy whenever all of the action vectors are equal.

Figure 5 illustrates this policy in the state space of two- and three-task problems. Figure 5(a) shows an example policy for a two-task problem instance. The decision offset $\mathbf{d}$ shifts the decision boundary parallel to the target utilization ray, while the action vectors determine the policy action on either side of the decision boundary. In two dimensions, the action vectors are not strictly necessary, since any policy that runs task T_2 above the decision boundary and T_1 below it will diverge, reaching states with arbitrarily negative costs, since this policy eventually dispatches the same task repeatedly.

The action vectors are necessary when there are three or more tasks. This is illustrated in Figures 5(b) and 5(c). In Figure 5(b), we show the time horizon H_t situated in three-space. The action vectors are shown in the plane. Figure 5(c) shows the perpendicular projection of the time horizon onto the plane; the decision boundaries for each action are shown, and consist of the region where the offset between a state and $t\mathbf{u} + \mathbf{d}$ is well-aligned with one of the action vectors. Notice that the policy breaks this simplex into three conic regions emanating from $t\mathbf{u} + \mathbf{d}$.

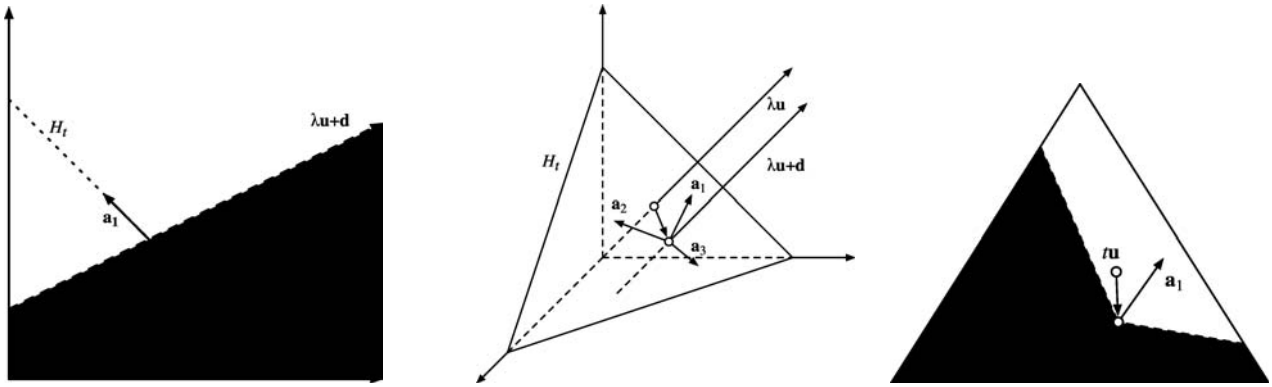
For any number of tasks n, the decision offset and action vectors act to partition each time horizon H_t into n distinct cones whenever the action vectors are distinct. We demonstrate this formally in the proof of Lemma 1.

Lemma 1. *For any decision offset $\mathbf{d}$, action vectors $\mathbf{A}$, time horizon H_t, and task T_i, the set of states*

$$\Lambda_{t,i} = \{\mathbf{x} \in H_t : i \in \operatorname{argmax}_j\{\mathbf{a}_j^\top \mathbf{z}(\mathbf{x})\}\}$$

is a cone with apex $t\mathbf{u} + \mathbf{d}$.

[1]Since we constrain the decision offset $\mathbf{d}$ and each action vector $\mathbf{a}_i$ to be parallel to the time horizons, they actually lie in an $(n-1)$-dimensional space and could be parameterized using $(n-1)$-vectors instead of n-vectors. We adopt the higher-dimensional representation here for expository purposes, since it can be communicated more concisely.



(a) two tasks (b) three tasks (c) three task horizon H_t

Fig. 5. Illustration of conic policies in two dimensions (Figure 5(a)) and three dimensions (Figures 5(b) and 5(c)). Figure 5(c) shows the time horizon H_t from Figure 5(b) with decision boundaries between regions where the policy is homogenous.

Proof: A set $\mathcal{Y}$ is a cone with apex $\mathbf{v}$ iff $\mathcal{Y}$ is convex and for any $\mathbf{y} \in \mathcal{Y}$, any $\mathbf{y}'$ on the ray from $\mathbf{v}$ through $\mathbf{y}$ is also in $\mathcal{Y}$. We demonstrate below that these two properties hold for $\Lambda_{t,i}$ relative to $t\mathbf{u} + \mathbf{d}$. We make use of the fact that for any real-valued n-vector $\mathbf{v}$, $\tau(\mathbf{v}) \equiv \mathbf{v}^\top \mathbf{1}$ is a linear map.

Convexity: Suppose that $\mathbf{x}$ and $\mathbf{x}'$ are states in $\Lambda_{t,i}$ and that $\mathbf{y} = \alpha\mathbf{x} + \beta\mathbf{x}'$ is a utilization state for some $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$. $\mathbf{z}(\cdot)$ is convex, *i.e.,*

$$\begin{aligned}\mathbf{z}(\alpha\mathbf{x} + \beta\mathbf{x}') &= \alpha\mathbf{x} + \beta\mathbf{x}' - \tau(\alpha\mathbf{x} + \beta\mathbf{x}')\mathbf{u} - \mathbf{d} \\ &= \alpha\mathbf{x} + \beta\mathbf{x}' - \alpha\tau(\mathbf{x})\mathbf{u} - \beta\tau(\mathbf{x}')\mathbf{u} - \alpha\mathbf{d} - \beta\mathbf{d} \\ &= \alpha\mathbf{z}(\mathbf{x}) + \beta\mathbf{z}(\mathbf{x}'),\end{aligned}$$

therefore

$$\mathbf{A}^\top\mathbf{z}(\mathbf{y}) = \alpha\mathbf{A}^\top\mathbf{z}(\mathbf{x}) + \beta\mathbf{A}^\top\mathbf{z}(\mathbf{x}').$$

Since $\mathbf{a}_i$ maximizes each term in the right-hand side, it also maximizes the left-hand side, implying that $\mathbf{y}$ is in $\Lambda_{t,i}$.

Homogeniety: For any $\mathbf{x}$ in $\Lambda_{t,i}$, a state $\mathbf{y}$ in H_t is along the ray through $\mathbf{x}$ from $t\mathbf{u} + \mathbf{d}$ iff $\mathbf{y} = \lambda\mathbf{z}(\mathbf{x}) + t\mathbf{u} + \mathbf{d}$ for some $\lambda > 0$. Since $\mathbf{z}(\mathbf{x})$ and $\mathbf{d}$ are perpendicular to $\mathbf{1}$, $\tau(\mathbf{z}(\mathbf{x})) = \tau(\mathbf{d}) = 0$. Therefore,

$$\begin{aligned}\mathbf{z}(\mathbf{y}) &= \mathbf{z}(\lambda\mathbf{z}(\mathbf{x}) + t\mathbf{u} + \mathbf{d}) \\ &= \lambda\mathbf{z}(\mathbf{x}) + t\mathbf{u} + \mathbf{d} - \tau(\lambda\mathbf{z}(\mathbf{x}) + t\mathbf{u} + \mathbf{d})\mathbf{u} - \mathbf{d} \\ &= \lambda\mathbf{z}(\mathbf{x}) + t\mathbf{u} + \mathbf{d} - \lambda\tau(\mathbf{z}(\mathbf{x}))\mathbf{u} - t\mathbf{u} - \tau(\mathbf{d})\mathbf{u} - \mathbf{d} \\ &= \lambda\mathbf{z}(\mathbf{x}),\end{aligned}$$

and so

$$\mathbf{A}^\top\mathbf{z}(\mathbf{y}) = \lambda\mathbf{A}^\top\mathbf{z}(\mathbf{x}).$$

It follows that if $\mathbf{a}_i$ maximizes $\mathbf{A}^\top\mathbf{z}(\mathbf{x})$, it also maximizes $\mathbf{A}^\top\mathbf{z}(\mathbf{y})$, and so $\mathbf{y}$ is in $\Lambda_{t,i}$. ■

Above, in the statement of Lemma 1 we define the cone $\Lambda_{t,i}$ as the set of states in the horizon where $\mathbf{a}_i$ has maximal response. It might appear more natural to define $\Lambda_{t,i}$ as the set of states in horizon H_t where the parameterized policy dispatches task T_i. However, this latter definition breaks down at the decision boundaries, since if we use a nondeterministic tie breaking procedure, the policy may not be homogenous along these boundaries.

Conic policies select among the tasks whose action vectors yield the greatest response in any given state. Our proof of Lemma 2 formalizes periodicity of the conic policy by showing that the set of action vectors that gives the greatest response $\mathbf{x}$ also yields the greatest response at any utilization state along the ray $\{\mathbf{x} + \lambda\mathbf{u} : \lambda \geq 0\}$.

Lemma 2. *For any decision offset* $\mathbf{d}$*, action vectors* $\mathbf{A}$*, and utilization state* $\mathbf{x}$*, if* T_i *maximizes* $\mathbf{a}_i^\top\mathbf{z}(\mathbf{x})$ *among all tasks, then* T_i *maximizes* $\mathbf{a}_i^\top\mathbf{z}(\mathbf{x} + \lambda\mathbf{u})$ *for any real scalar* λ*.*

Proof: Since

$$\begin{aligned}\mathbf{z}(\mathbf{x} + \lambda\mathbf{u}) &= \mathbf{x} + \lambda\mathbf{u} - \tau(\mathbf{x} + \lambda\mathbf{u})\mathbf{u} - \mathbf{d} \\ &= \mathbf{x} + \lambda\mathbf{u} - \tau(\mathbf{x})\mathbf{u} - \lambda\mathbf{u} - \mathbf{d} \\ &= \mathbf{z}(\mathbf{x}),\end{aligned}$$

so for any task T_i,

$$\mathbf{a}_i^\top\mathbf{z}(\mathbf{x}) = \mathbf{a}_i^\top\mathbf{z}(\mathbf{x} + \lambda\mathbf{u}),$$

and so if $\mathbf{a}_i^\top\mathbf{z}(\mathbf{x})$ is maximal among all tasks, then it also maximizes $\mathbf{a}_i^\top\mathbf{z}(\mathbf{x} + \lambda\mathbf{u})$. ■

The example policy shown in Figure 4 is periodic but *not* conic: in order for the partition induced by the policy to be conic, we would need to be able to represent the decision boundary by exactly three rays emanating from a common point. Thus, the conic partition of the parameterized policy is not optimal in general. In Section V-B, we show that while the best conic policy may not be optimal among all possible scheduling policies, it contains stable policies that maintain the system near target utilization. In Section VI we also demonstrate that we can find conic policies that perform well; particularly, there are conic policies that consistently outperform the heuristic scheduling policies described in Section IV on problems that are too large to solve using finite state approximations.

B. Stable Conic Policies

Above, we described a case in which the system might diverge to states with arbitrarily negative costs. In Figure 3, if the policy instead dispatched task T_1 in every state below the decision boundary, upon reaching any of these states, the policy would then repeatedly run that task forever. We say that such a policy is unstable, since it never settles into a region with bounded cost. We now derive a sufficient condition under which a conic policy is guaranteed to be stable. Informally, a stable policy is one under which the system state is guaranteed to converge to a region with finite, bounded costs.

A *trajectory* generated under policy π is a sequence of utilization states $(\mathbf{x}_k)_{k=0}^{\infty}$ such that $\mathbf{x}_{k+1}$ is distributed according to $P(\cdot|\mathbf{x}_k, \pi(\mathbf{x}_k))$. We say that a scheduling policy is *stable* iff we can guarantee that any trajectory generated under that policy is eventually within some bounded neighborhood of the utilization ray. The cost function in Equation 4 is defined as the negative Manhattan distance between the state $\mathbf{x}$ and the ideal point $\tau(\mathbf{x})\mathbf{u}$ on the utilization ray. This means that a stable policy maintains the system in states with costs relatively near zero. This in turn ensures that every task makes progress relative to one another.

Theorem 1 provides a sufficient condition on the decision offset and action vectors to guarantee that the corresponding conic policy is stable. The proof is stated in terms of the Euclidean distance rather than the Manhattan distance; since the two norms are topologically equivalent, stability in Euclidean distance also implies stability in the Manhattan distance.

Theorem 1. *If $\pi = \pi(\cdot; \mathbf{d}, \mathbf{A})$ is a conic policy, and there is some $\varepsilon > 0$ such that for every utilization state $\mathbf{x}$,*

$$(\Delta_{\pi(\mathbf{x})} - \mathbf{u})^{\top} \mathbf{z}(\mathbf{x}) \leq -\varepsilon \left\| \mathbf{z}(\mathbf{x}) \right\| \tag{8}$$

where $\|\cdot\|$ is the Euclidean norm, then π is stable.

We defer a formal proof of Theorem 1 to the appendix, and provide only a brief sketch here. The vector $(\Delta_{\pi(\mathbf{x})} - \mathbf{u})$ is the instantaneous change in state at $\mathbf{x}$ when following π. The precondition of the theorem requires that the angle between this state derivative and the displacement $\mathbf{z}(\mathbf{x})$ between $\mathbf{x}$ and the decision ray is negative. This guarantees that there is always an instantaneous reduction in distance from the decision ray under π.

However, since the state changes in discrete jumps according to the task durations, it is possible for the policy to move the system from a state that is close to the decision ray to one that is farther away. Since tasks have bounded worst-case durations, the policy will always move the system closer to the decision ray from a state that is far enough away. Closer to the decision ray, the policy may throw the state farther from the decision ray, but because of the duration bounds, the distance of this successor state from the decision ray is bounded. Therefore, we can draw a cylinder with finite radius centered around the decision ray such that any trajectory starting from a state inside the cylinder must stay inside, while trajectories originating outside of the cylinder are eventually pulled inside, and then stay there.

We can conclude that the system state must always enter into a bounded neighborhood of the decision ray. Since the utilization ray is at a fixed distance from the decision ray, this result also implies that the policy converges to a bounded region about the target utilization.

While Theorem 1 provides a sufficient condition to guarantee stability, it does not supply us with a stable parameterization. Corollary 1 provides an example of a stable policy.

Corollary 1. *A conic policy $\pi(\cdot; \mathbf{d}, \mathbf{A})$ with action vectors*

$$\mathbf{a}_i = (\mathbf{u} - \Delta_i) / \left\| \mathbf{u} - \Delta_i \right\| \tag{9}$$

is stable for any choice of decision offset $\mathbf{d}$.

The proof of this corollary is surprisingly involved, and so we defer this to the appendix as well. The intuition behind the proof is relatively straightforward, however. By construction, each action vector $\mathbf{a}_i$ defined by Equation 9 points exactly opposite the direction of travel relative to the utilization target, $(\Delta_i - \mathbf{u})$; that is, the angle between these two vectors is $180°$. Because of this relationship, we can conclude that if the angle between an action vector $\mathbf{a}_i$ and the displacement $\mathbf{z}(\mathbf{x})$ is "comfortably" smaller than $90°$ (expressed as a constraint on their dot product in terms of ε), then the angle between $\mathbf{z}(\mathbf{x})$ and the $(\Delta_i - \mathbf{u})$ must be more than $90°$, satisfying the precondition of Theorem 1. Figure 6 illustrates these action vectors for a three-task problem instance.

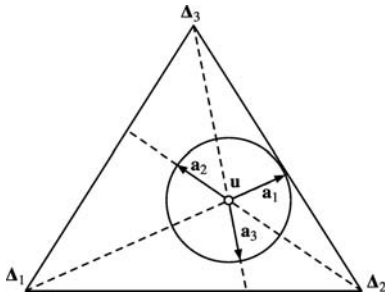


Fig. 6. Scaled action vectors $\mathbf{a}_i = (\mathbf{u} - \Delta_i) / \left\| \Delta_i - \mathbf{u} \right\|$.

The choice of decision offset is superficial in determining stability. If a policy is stable with offset $\mathbf{d}$, it will also be stable with offset $\mathbf{d}'$. This is because the state derivative when dispatching a particular task is independent of the decision offset. A stable choice of action vectors causes the system state to enter a stochastic orbit around the decision ray; moving the decision offset just moves that orbit through the state space.

This line of reasoning seems to suggest that we should always choose $\mathbf{d} = \mathbf{0}$ as the decision offset, so that the system orbits around the utilization ray. This is not the case, however. In practice, the state stochastically orbits the decision ray, but because of the difference in durations between tasks and because there is a non-smooth change in the direction of travel when changing actions, the average location tends to differ from the decision ray. Selecting a good conic policy parameterization appears to consist of establishing a decision offset and action vectors so that this average state is as close as possible to the utilization target.

VI. Experimental Evaluation

Finding the optimal conic policy analytically appears to be a difficult problem. Rather than approaching the problem from this angle, we instead employ stochastic optimization techniques to select good parameterizations of the conic policy. We discuss these methods and evaluate them empirically in this section.

We ran tests comparing the performance of selected conic policies to the heuristic scheduling policies described in Section IV and where possible to the wrapped state model solutions from Section III. To find good conic policies we implemented the hill climbing and policy gradient search methods described by Kohl and Stone [12].

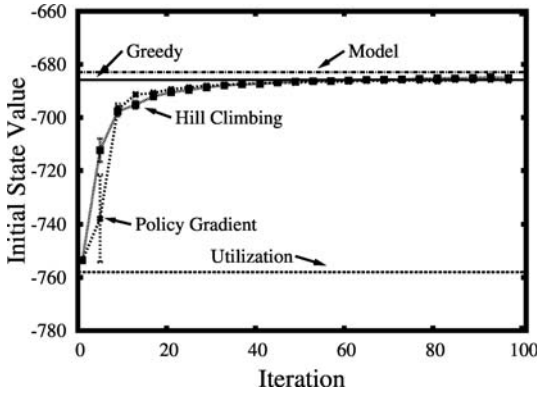
Both of these search methods follow a similar outline. We begin with an initial policy parameterization; in each case, the stable conic policy in Equation 9 with a decision offset of $\mathbf{d} = \mathbf{0}$. At each iteration, we generate a population of nearby policies by adding small random perturbations to each parameter. In our experiments, we found that a population size of $3n(n+1)$ works well for either search strategy, where n is the number of tasks. These policies are evaluated by performing Monte Carlo evaluation [13] – i.e., by repeatedly simulating the policy from the initial state $\mathbf{x} = \mathbf{0}$ to estimate Equation 1. This population is then used to determine a policy for the next iteration.

Hill climbing search chooses the policy at each iteration by selecting the policy with greatest estimated value among this population. As in Kohl and Stone's work [12], we select the best policy among these perturbed parameter settings regardless of whether that policy is worse than the current policy. This affords the algorithm some limited ability to escape local optima. In our experiments, parameter perturbations were drawn uniformly at random from the interval $\pm(1/m+0.98^m)$ at iteration m.

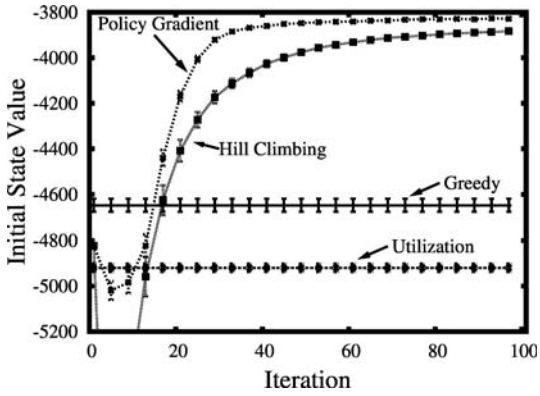
Policy gradient search instead uses these perturbations to estimate the gradient of the value with respect to the policy parameters. The policy at the next iteration is determined by stepping a fixed distance along the gradient. In our experiments, parameters were perturbed by adding a value at random from among $\{0, \pm m^{-1/6}\}$, while the step size along the gradient was $1/m+0.98^m$. We decay the step size and the perturbation size in the hill climbing experiments so that we eventually settle on some policy. The specific decay rates were chosen, loosely speaking, to keep this from happening too quickly. A formal discussion of appropriate decay strategies can be found, for example, in [14].

We performed two experiments comparing the performance of conic policies, heuristic policies, and when possible policies obtained by finite state approximation methods. The first experiment examines how the value of each conic policy evolves with each iteration of the policy search method. The second experiment examines the performance of the different policies for problem instances as a function of the number of tasks.

In each problem instance the task duration distributions were random histograms with worst-case execution time W_i in the interval $[2, 32]$. The utilization targets for each problem instance were selected by choosing integers $\mathbf{q} \in [2, 32]^n$ uniformly at random, so that the utilization target is $\mathbf{u} = \mathbf{q} \cdot (\sum_i q_i)^{-1}$. For both the heuristic and conic policies, evaluating the value function at the initial state was carried out using Monte Carlo evaluation.



(a) 4 Task Problem Instance



(b) 10 Task Problem Instance

Fig. 7. Experimental results showing the performance of policy search strategies as a function of the number of iterations performed. Values labeled "Hill climbing" and "Policy Gradient" show the eponymous search performance, "Greedy" and "Utilization" show the heuristic policy performance, and "Model" the finite state approximation of optimal.

In the first experiment, we generated a single problem instance with 4 tasks. We performed 100 iterations of policy search using both hill climbing and policy gradient search and estimated the value at each iteration. We compared their performance to the utilization policy (π_u from Equation 5), the greedy policy (π_g from Equation 6), and a finite state approximation to the optimal policy, as described in Section III. These values are shown in Figure 7(a) as "Hill Climbing," "Policy Gradient," "Utilization," "Greedy," and "Model," respectively. Since costs are negative, policy values closer to zero indicate better performance. 95% confidence intervals were obtained by averaging across 30 repetitions of each search strategy.

Figure 7(a) shows that explicitly solving the MDP model gives the best results. However, Hill Climbing, Policy Gradient and Greedy produce comparable results. Utilization performs relatively poorly.

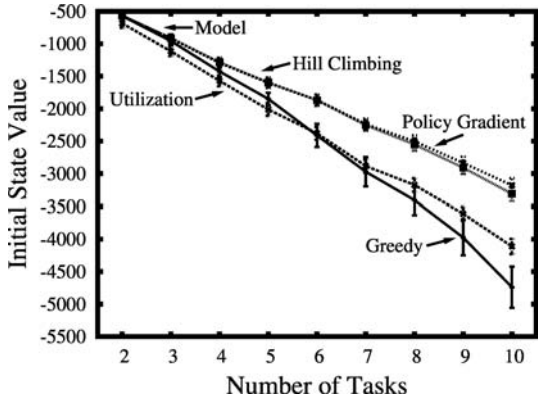


Fig. 8. Comparison of policy performance for varying numbers of tasks. Finite state approximation of the optimal policy is shown only for two and three tasks.

Figure 7(b) shows the results of repeating this experiment using a 10 task problem instance. In this case solving the MDP model is intractable. Computing the exact value of either heuristic policy also requires enumerating their set of reachable states, and thus is also intractable. Instead Monte Carlo simulation is used to estimate their values, which are shown with 95% confidence intervals. In this case Hill Climbing outperforms either heuristic policy. The additional structure of Policy Gradient search allows it to outperform Hill Climbing.

In the second experiment we compared the performance of these policies across problem instances with varying number of tasks. For each number of tasks we generated 100 independent problem instances with the same method described above. Average values with 95% confidence intervals are shown in Figure 8. We report the policy found after 100 iterations of search. Results are shown for the finite state approximation for two and three task problem instances.

In two or three task problem instances, Greedy, Model, Hill Climbing and Policy Gradient all perform similarly. With more tasks the conic policy clearly outperforms either heuristic policy. This supports the use of conic policies for scaling to larger problem instances.

VII. Conclusion

In this paper we have introduced a scalable conic scheduling policy design technique that compactly approximates the geometric structure of policies obtained using direct solution techniques. This technique allows us to derive good scheduling policies for open soft real-time systems with large numbers of tasks. Our results indicate that direct solution techniques are most appropriate when tractable, while conic policies provide strong scalable performance where direct solution methods fail.

Our experiments demonstrate that search is able to find a good conic policy when initialized with the stable policy from Equation 9. However, it is unclear whether these methods are likely to converge to a global optimum among conic policies when restricted to this initial parameterization. Typically, this would be addressed by using randomized restarts in order to sample many local optima. However, that approach fails for our task scheduling problem, as most random conic policies are unstable and policies in their neighborhood also tend to be unstable. Since unstable policies reach states with large magnitude cost, these policies have almost uniformly low value, so there is no clear direction that search can follow to reach a good parameterization.

We plan to consider two approaches to address this problem in future work. One is to use a richer policy representation; for example, choosing to dispatch tasks at random, with probability proportional to to the action vector responses, may provide a more informative value gradient [11]. A second method is to derive a more comprehensive characterization of stable conic policies, which would allow us to sample safely from a wider variety of initial conditions.

Appendix

Proof of Theorem 1: Let $(\mathbf{x}_k)_{k=0}^{\infty}$ be a trajectory, with $\mathbf{x}_0$ arbitrary and $\mathbf{x}_{k+1}$ determined by executing $\pi(\mathbf{x}_k)$ in $\mathbf{x}_k$. We need to prove that this (arbitrary) trajectory converges to a bounded neighborhood of the utilization ray. Our proof consists of two parts: first we show that the trajectory eventually enters this neighborhood, and then we show that the trajectory can not escape this neighborhood.

Let $\mathbf{x}$ be an arbitrary state and let $\pi(\mathbf{x}) = i$. Suppose that $\mathbf{y} = \mathbf{x} + t\Delta_i$ is a successor of $\mathbf{x}$ under π. We can write the displacement $\mathbf{z}(\mathbf{y})$ between $\mathbf{y}$ and the decision ray in terms of the displacement at $\mathbf{x}$,

$$\begin{aligned}\mathbf{z}(\mathbf{y}) &= \mathbf{x} + t\Delta_i - \tau(\mathbf{x})\mathbf{u} - t\mathbf{u} - \mathbf{d} \\ &= \mathbf{z}(\mathbf{x}) + t(\Delta_i - \mathbf{u}).\end{aligned}$$

This allows us to derive an upper bound on the squared magnitude of $\mathbf{z}(\mathbf{y})$, as

$$\begin{aligned}\|\mathbf{z}(\mathbf{y})\|^2 &= (\mathbf{z}(\mathbf{x}) - t(\Delta_i - \mathbf{u}))^\top (\mathbf{z}(\mathbf{x}) - t(\Delta_i - \mathbf{u})) \\ &= \mathbf{z}(\mathbf{x})^\top \mathbf{z}(\mathbf{x}) + t^2(\Delta_i - \mathbf{u})^\top(\Delta_i - \mathbf{u}) \\ &\quad + 2t(\Delta_i - \mathbf{u})^\top \mathbf{z}(\mathbf{x}) \\ &\leq \|\mathbf{z}(\mathbf{x})\|^2 + t^2 \|\Delta_i - \mathbf{u}\|^2 - 2t\varepsilon \|\mathbf{z}(\mathbf{x})\|.\end{aligned}$$

Defining

$$\eta(\mathbf{x}, t) \equiv t^2 \left\|\Delta_{\pi(\mathbf{x})} - \mathbf{u}\right\|^2 - 2t\varepsilon \|\mathbf{z}(\mathbf{x})\|$$

allows us to write the inequality above more concisely as

$$\|\mathbf{z}(\mathbf{y})\|^2 \leq \|\mathbf{z}(\mathbf{x})\|^2 + \eta(\mathbf{x}, t). \tag{10}$$

We define $M = \max_i\{W_i^2 \|\Delta_i - \mathbf{u}\|^2\}$ to bound the first term of $\eta(\mathbf{x}, t)$ above (recall that W_i is the worst-case execution time of Task T_i). For any $\alpha > 0$ let $\rho_\alpha = (M + \alpha)/(2\varepsilon)$ be the radius of a cylinder centered on the decision ray. Then if $\|\mathbf{z}(\mathbf{x})\| \geq \rho_\alpha$,

$$\eta(\mathbf{x}, t) \leq M - 2\varepsilon(M + \alpha)/(2\varepsilon) = -\alpha.$$

We can substitute this inequality into Equation 10 to get $\|\mathbf{z}(\mathbf{y})\|^2 \leq \|\mathbf{z}(\mathbf{x})\|^2 - \alpha$. In other words, executing the policy

action always reduces the distance between $\mathbf{x}$ and the decision ray if $\mathbf{x}$ is far enough away.

This result guarantees that the trajectory is eventually within ρ_α of the decision ray for any $\alpha > 0$. If this were not the case, we would be able to find some K such that any $k \geq K$ has $\|\mathbf{z}(\mathbf{x}_k)\| \geq \rho_\alpha$, but for any $m > 0$, we have

$$\begin{aligned}\|\mathbf{z}(\mathbf{x}_{K+m})\|^2 &\leq \|\mathbf{z}(\mathbf{x}_{K+m-1})\|^2 - \alpha \\ &\vdots \\ &\leq \|\mathbf{z}(\mathbf{x}_K)\|^2 - m\alpha,\end{aligned}$$

so $\mathbf{x}_{K+m}$ is within ρ_α of the decision ray for large m.

By the triangle inequality, we have

$$\begin{aligned}\|\mathbf{z}(\mathbf{y})\| &= \|\mathbf{z}(\mathbf{x}) + t(\Delta_i - \mathbf{u})\| \\ &\leq \|\mathbf{z}(\mathbf{x})\| + t\,\|\Delta_i - \mathbf{u}\| \\ &\leq \|\mathbf{z}(\mathbf{x})\| + M^{1/2},\end{aligned}$$

so if $\|\mathbf{z}(\mathbf{x})\| \leq \rho_\alpha$, $\|\mathbf{z}(\mathbf{y})\| \leq \rho_\alpha + M^{1/2}$. Since the state gets closer to the decision ray when $\|\mathbf{z}(\mathbf{x})\|$ is greater than ρ_α, and cannot get farther than $\rho_\alpha + M^{1/2}$ when the state is inside this neighborhood, the trajectory must eventually enter and stay within distance $\rho_\alpha + M^{1/2}$ of the decision ray for any $\alpha > 0$. Since the trajectory is arbitrary, this must hold for any trajectory generated while following π. ■

Proof of Corollary 1: To simplify notation, let $\mathbf{z}$ denote $\mathbf{z}(\mathbf{x})$ for some arbitrary state $\mathbf{x}$. Under Equation 9,

$$\mathbf{a}_i^\top \mathbf{z} = -(\Delta_i - \mathbf{u})^\top \mathbf{z} / \|\Delta_i - \mathbf{u}\|.$$

Therefore, we just need to show that for some $\alpha > 0$,

$$\max_i \{\mathbf{a}_i^\top \mathbf{z}\} \geq \alpha \|\mathbf{z}\|$$

for every $\mathbf{z}$, since then $(\Delta_{\pi(\mathbf{x})} - \mathbf{u})^\top \mathbf{z} \leq -\alpha \|\mathbf{z}\| \|\Delta_i - \mathbf{u}\|$, satisfying Theorem 1's precondition. To simplify the discussion, we assume that $\min_i\{\|\Delta_i - \mathbf{u}\|\}$ is a factor of α. This term is guaranteed to be positive because no single task is assigned the entire processor. Demonstrating the claim therefore reduces to demonstrating that in every state there is a task T_i such that $(\mathbf{u} - \Delta_i)^\top \mathbf{z} \geq \alpha \|\mathbf{z}\|$.

For the sake of contradiction, suppose that for all $\alpha > 0$, there is a state $\mathbf{x}$ (and corresponding displacement $\mathbf{z}$) such that

$$\max_i \{(\mathbf{u} - \Delta_i)^\top \mathbf{z}\} < \alpha \|\mathbf{z}\|. \tag{11}$$

We can rewrite the left-hand side of Equation 11 according to

$$\max_i \{(\mathbf{u} - \Delta_i)^\top \mathbf{z}\} = \mathbf{u}^\top \mathbf{z} - \min_i \{z_i\}$$

(recall that $\Delta_i^\top \mathbf{z} = z_i$). We will procede to show that satisfying Equation 11 for arbitrarily small α requires an invalid utilization target. To achieve this, we first need an upper bound on the left-hand side of the equation.

To obtain this bound, recall that $\mathbf{z}$ is the displacement between $\mathbf{x}$ and the decision ray (see Equation 7). This lies in a plane perpendicular to $\mathbf{1}$, and so $\sum_{i=1}^n z_i = 0$. Let ζ be the sum of positive components of $\mathbf{z}$, then $-\zeta$ is the sum of its negative components.

Without loss of generality, we may assume that the components of the utilization target are ordered with $u_1 \geq u_2 \geq \ldots \geq u_n$. This implies that u_n lies in the interval $(0, 1/n]$ and that $u_1 \leq 1 - u_n(n-1)$, as otherwise $\mathbf{u}$'s components would not sum to one.

Using these observations, it is straightforward to verify that for a fixed utilization target, the left-hand side of Equation 11 is maximized by fixing $z_1 = \zeta$ and $z_n = -\zeta$, and setting the other components to zero. Then

$$\begin{aligned}\mathbf{u}^\top \mathbf{z} - \min_i\{z_i\} &\leq \zeta u_1 - \zeta u_n + \zeta \\ &\leq \zeta(2 - n \cdot u_n)\end{aligned}$$

Substituting this bound into Equation 11 yields the inequality $\zeta(2 - n \cdot u_n) < \alpha \|\mathbf{z}\|$, or equivalently,

$$u_n > 2/n - \alpha \|\mathbf{z}\| / (\zeta n).$$

Thus as α approaches zero, the smallest utilization target u_n must exceed $1/n$, a contradiction. ■

REFERENCES

[1] R. Glaubius, T. Tidwell, W. D. Smart, and C. Gill, "Scheduling design and verification for open soft real-time systems," in *Proceedings of the 2008 Real-Time Systems Symposium (RTSS 2008)*, 2008, pp. 505–514.

[2] T. Tidwell, R. Glaubius, C. Gill, and W. D. Smart, "Scheduling for reliable execution in autonomic systems," in *Proceedings of the 5th International Conference on Autonomic and Trusted Computing (ATC 2008)*, ser. Lecture Notes in Computer Science, vol. 5060, 2008, pp. 149–161.

[3] A. Srinivasan and J. H. Anderson, "Efficient scheduling of soft real-time applications on multiprocessors," *Journal of Embedded Computing*, vol. 1, no. 2, pp. 285–302, 2005.

[4] S. K. Baruah, N. K. Cohen, C. G. Plaxton, and D. A. Varvel, "Proportionate progress: A notion of fairness in resource allocation," *Algorithmica*, vol. 15, no. 6, pp. 600–625, 1996.

[5] A. Srinivasan and J. H. Anderson, "Optimal rate-based scheduling on multiprocessors," *Journal of Computer and Systems Science*, vol. 72, no. 6, pp. 1094–1117, 2006.

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

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

[8] T. Dean and R. Givan, "Model minimization in markov decision processes," in *AAAI '97: Proceedings of the 14th National Conference on Artificial Intelligence*. AAAI, 1997, pp. 1006–1111.

[9] R. Givan, T. Dean, and M. Greig, "Equivalence notions and model minimization in markov decision processes," *Artificial Intelligence*, vol. 147, no. 1-2, pp. 163–223, 2003.

[10] L. Li, T. J. Walsh, and M. L. Littman, "Towards a unified theory of state abstraction for MDPs," in *In Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, 2006, pp. 531–539.

[11] R. S. Sutton, D. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*, vol. 12, 2000, pp. 1057–1063.

[12] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *AAAI '04: Proceedings of the 19th National Conference on Artificial Intelligence*, 2004, pp. 611–616.

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

[14] M. C. Fu, "Stochastic gradient estimation," in *Handbooks in Operations Research and Management Science: Simulation*, S. G. Henderson and B. L. Nelson, Eds. Elsevier Science Publishers, Ltd., 2006, pp. 575–616.