A Polynomial-Time Approximation Scheme for Feasibility Analysis in Static-Priority Systems with Bounded Relative Deadlines*

Nathan Fisher and Sanjoy Baruah†

Department of Computer Science, CB #3175
The University of North Carolina
Chapel Hill, NC 27599
{fishern, baruah}@cs.unc.edu
Phone: (919) 962-1803 Fax: (919) 962-1416

Abstract

Current feasibility tests for the static-priority scheduling of periodic task systems run in pseudo-polynomial time. We present a polynomial-time approximation scheme (PTAS) for feasibility in static-priority systems where each task's relative deadline is constrained to be at most its period. This test is an approximation with respect to the amount of processor capacity that must be "sacrificed" for the test to become exact. We show that an arbitrary level of accuracy, ϵ, may be chosen for the approximation scheme, and present a run-time bound that is polynomial in terms of ϵ and the number of tasks, n.

Keywords: Real-time scheduling; Uniprocessor systems; Static-priority systems; Feasibility analysis; Polynomial-time approximation schemes.

*Supported in part by the National Science Foundation (Grant Nos. ITR-0082866, CCR-0204312, and CCR-0309825).

†Corresponding author

1 Introduction

An exact feasibility test for the preemptive uniprocessor scheduling of sets of periodic tasks, each with its deadline parameter equal to its period, was introduced by Lehoczky, Sha, and Ding [4]. The test determines whether a set of tasks is feasible using the *rate monotonic algorithm* [7]. The response time of each task is calculated, and checked against its deadline. Audsley *et al.* [2] developed a feasibility test for sets of tasks in which deadlines can be less than periods, and which are scheduled using the *deadline monotonic algorithm* [6]. Lehoczky [5] provided a more general feasibility test for periodic task systems where the relation between deadlines and periods may be arbitrary.

In each of the aforementioned feasibility tests, the running time of the test is dependent on the values of the parameters of the tasks in the task system. Thus, these are *psuedo-polynomial time* tests. Albers and Slomka [1] present a *polynomial-time approximation scheme (PTAS)* for feasibility of a sporadic task system using a dynamic-priority scheduling algorithm. The feasibility test accepts as input, the specifications of a task system and a constant ϵ, $0 < \epsilon < 1$, and is an approximation scheme in the following sense:

> If the test returns "feasible", then the task set is guaranteed to be feasible on the processor for which it had been specified. If the test returns "infeasible", the task set is guaranteed to be infeasible *on a slower processor*, of computing capacity $(1 - \epsilon)$ times the computing capacity of the processor for which the task system had been specified.

In this paper, we extend the results of Albers and Slomka to the domain of static-priority scheduling. That is, we present a PTAS for static-priority feasibility analysis that makes a performance guarantee similar to the one above: for any specified value of ϵ, the PTAS correctly identifies, in time polynomial in the number of tasks in the task system, all task systems that are static-priority feasible on a processor that has $(1 - \epsilon)$ times the computing capacity of the processor for which the task system is specified.

Since many static-priority feasibility-analysis algorithms (in particular, those based upon iterative convergence of response-time equations) have been observed to converge extremely rapidly in practice, it may be argued that such a PTAS is not particularly useful. However, the presence or otherwise of such a PTAS is interesting from a theoretical perspective as part of the ongoing debate concerning the relative merits of static-priority and dynamic-priority scheduling: since a PTAS was recently obtained for *dynamic-priority* uniprocessor feasibility analysis, it is of interest to know whether static-priority feasibility analysis could be approximated as efficiently as dynamic-priority analysis. The PTAS presented in this paper answers this question in the affirmative.

The running time of current feasibility tests for static-priority task systems depends on the ratio between the largest and smallest period. Therefore, the complexity of current feasibility tests for task systems with widely-varying periods may prohibit their use in automatic synthesis tools. The running time of the approximation proposed in this paper is completely independent of the tasks' periods, and depends only on the number of tasks and the accuracy constant, ϵ. Thus, the approximation offers a reduction in complexity for many task sets, and its predictable worst-case

run-time guarantees a quick estimate for automatic synthesis tools exploring a real-time system design space.

The remainder of this paper is organized as follows. We formally define our task model in the next section. We briefly summarize (Section 3.1) how the *request-bound function* abstraction, which plays a crucial role in the various static-priority feasibility tests mentioned above [4, 2, 5], can be approximated by a function that is easily computed, and which satisfies the property that its value "closely" tracks (in a sense that will be made clearer in Section 4) the exact value of the request-bound function. We use this approximation to the request-bound function to construct an approximation to the exact DM-feasibility test [3] in Section 3.2. In Section 3.3, we give an illustrative example of the approximate and exact feasibility tests. We prove the correctness of the approximate feasibility test in Section 4. The polynomial time complexity of this approximate test is derived in Section 5. In Section 6, we describe some of the challenges in applying the approximate test to a system with arbitrary relative deadlines, and briefly sketch a potential approach to developing an approximate feasibility test for this model.

2 Task Model

We consider both *periodic* and *sporadic* task models. A *task* $\tau_i = (e_i, d_i, p_i)$ is characterized by a *worst-case execution requirement* e_i and *(relative) deadline* d_i. In the periodic task model, the *period* p_i represents the exact interval between the arrival of jobs of τ_i. In the sporadic task model, p_i represents the *minimum inter-arrival separation* between jobs of τ_i. Each job has a worst-case execution requirement equal to e_i and a deadline that occurs d_i time-units after its arrival time. A periodic task system is *synchronous* if the first job of every task is released at the same time.

A *feasibility test* is a necessary and sufficient set of conditions for determining whether a given task system will meet all of its deadlines. In the remainder of this paper, we study approximation schemes for feasibility of both sporadic and synchronous periodic task systems that are to be executed on a preemptive uniprocessor platform. We will assume that for all tasks τ_i, $d_i \leq p_i$, unless otherwise specified.

Static-Priority Scheduling Algorithms

In static-priority systems, each task is assigned a distinct priority, and all jobs of a task execute at the task's priority. More formally, a job is said to be *active* at a specified time-instant in a schedule, if it has remaining execution time and has not missed its deadline. When a scheduling algorithm is invoked at time t, it will select the job with highest priority out of the set of active jobs at time t. Two well-studied static priority scheduling algorithms are *rate monotonic*(RM) and *deadline monotonic*(DM). RM, introduced in [7], assigns each task a priority equal to the inverse of its period. DM, first presented in [6], assigns each task a priority equal to the inverse of its relative deadline.

A task system τ is *feasible* with respect to static-priority systems if there exists a task priority assignment such that when τ is scheduled according to this priority assignment, all deadlines are

met. A static-priority scheduling algorithm A is *optimal over all static-priority algorithms*, if for every feasible task system τ, A produces a schedule in which all deadlines are met. RM is known to be optimal for static-priority algorithms when deadlines are equal to periods. DM is optimal for static-priority algorithms when deadlines are bounded by periods. Since, we are assuming that $d_i \leq p_i$, we will concentrate on the DM algorithm because it is optimal in this case.

3 Feasibility Test

In this section, we introduce an approximate feasibility test for static-priority systems where each task's relative deadline is constrained to be less than or equal to its period. We begin in Section 3.1 by defining a *request-bound function* (RBF) that bounds the amount of execution time requested by a task (similarly defined in [4, 2, 5]). An approximation to the RBF is defined such that the deviation from the RBF is bounded.

In Section 3.2, we define both an exact and approximate *cumulative request-bound function* based respectively on the exact and approximate request-bound functions for a task τ_i. The functions describe the cumulative execution requests over a time interval for task τ_i and all tasks of higher priority. Lehoczky *et al.* [4] showed that in a sporadic or synchronous periodic system, the *smallest* fixed point of the exact cumulative request-bound function for task τ_i is the time at which the processor can satisfy τ_i's request. If the smallest fixed point of task τ_i's cumulative request-bound function is no larger than its relative deadline, then τ_i is schedulable. If the smallest fixed point exceeds τ_i's deadline, then τ_i is not schedulable according to any static-priority scheduling algorithm. The remainder of Section 3.2 describes how we use the approximate cumulative request-bound function for an approximate feasibility test for static-priority task system τ. Section 3.3 gives an example task system, and applies our approximate feasibility test.

3.1 Request-Bound Function

In a periodic synchronous task system, the total execution time requested by a task τ_i can be expressed as a function of time. Every time a task τ_i releases a job, e_i additional units of processor time are requested. The following function provides an upper bound on the total execution time requested by task τ_i at time t:

$$\text{RBF}(\tau_i, t) \stackrel{\text{def}}{=} \left\lceil \frac{t}{p_i} \right\rceil e_i \tag{1}$$

In a sporadic task system, $\text{RBF}(\tau_i, t)$ represents the total execution time requested by τ_i in its worst-case phasing. Figure 1 shows an example of a RBF. Notice that the "step" function, $\text{RBF}(\tau_i, t)$ increases by e_i units every p_i time units.

Approximating the RBF

The function $\text{RBF}(\tau_i, t)$ is discontinuous every p_i time units. We call these discontinuities *steps*. We define an approximation that computes the first $(k-1)$ steps of $\text{RBF}(\tau_i, t)$ exactly (where k is

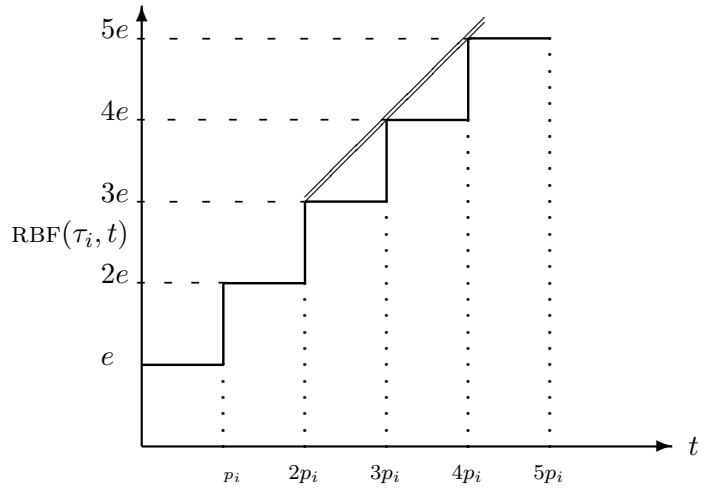

Figure 1: The step function denotes a plot of $\text{RBF}(\tau_i, t)$ as a function of t. The double line represents the function $\delta(\tau_i, t)$, approximating $\text{RBF}(\tau_i, t)$, where $k = 3$; for $t \leq 2p_i$, $\delta(\tau_i, t) \equiv \text{RBF}(\tau_i, t)$.

a constant, defined below), and is a linear approximation of $\text{RBF}(\tau_i, t)$, thereafter. The bounded number of steps $(k-1)$ for the approximation function will ultimately lead to a polynomial time complexity for the approximate feasibility test (shown in Section 5).

Recall the "accuracy" constant ϵ, $0 < \epsilon < 1$, discussed in the introduction. ϵ represents the desired accuracy of our approximate feasibility test. We choose a constant k based on ϵ. For the remainder of the paper, assume the integer constant k is defined as follows:

$$k \stackrel{\text{def}}{=} \lceil 1/\epsilon \rceil - 1 \tag{2}$$

We now define the following function $\delta(\tau_i, t)$; in section 4, we will prove that $\delta(\tau_i, t)$ closely approximates the function $\text{RBF}(\tau_i, t)$:

$$\delta(\tau_i, t) = \begin{cases} \text{RBF}(\tau_i, t) \ , & \text{for } t \leq (k-1)p_i \\ e_i + \frac{te_i}{p_i} \ , & \text{for } t > (k-1)p_i \end{cases} \tag{3}$$

Figure 1 shows that $\delta(\tau_i, t)$ is exactly $\text{RBF}(\tau_i, t)$ up to $(k-1)p_i$, (in this example, $k = 3$) and then is a linear approximation $t > (k-1)p_i$ that "bounds" $\text{RBF}(\tau_i, t)$ from above.

3.2 Description of Feasibility Test

Exact Test

For static-priority task systems with relative deadlines bounded by periods, Liu and Layland [7] showed that the *worst-case* response time for a job of task τ_i occurs when all tasks of priority greater than τ_i release a job simultaneously with τ_i. If a task τ_i releases a job J simultaneously with all higher priority tasks and each higher priority task τ_j releases subsequent jobs at the earliest legal

opportunity (i.e. the inter-arrival separation between jobs of higher-priority task τ_j is *exactly* p_j), then J has the largest response time of any job of task τ_i. In a sporadic or synchronous periodic task system with relative deadlines bounded by periods, it is necessary and sufficient to only check the response time of the first job of each task. If the response time of the first job of task τ_i is at most its relative deadline, then τ_i is schedulable; else, it is not schedulable. A task system τ is feasible on a uniprocessor *if and only if* the first job of each task τ_i has a worst-case response time at most d_i.

In order to determine the response-time for the first job of task τ_i, we must consider execution requests of τ_i and all jobs of tasks which may preempt τ_i. We define the following *cumulative request-bound function* based on RBF. Let $\mathbf{T}_{-i}$ be the set of tasks with priority greater than τ_i. Then, the cumulative request-bound function is defined as:

$$W_i(t) \stackrel{\text{def}}{=} e_i + \sum_{\tau_j \in \mathbf{T}_{-i}} \text{RBF}(\tau_j, t) \tag{4}$$

The cumulative request-bound function $W_i(t)$ is simply the total execution requests of all tasks of higher priority than τ_i over the interval $(0, t]$, and the execution request of one job of τ_i.

Audsley *et al.* [3] presented an exact feasibility test for task τ_i using DM: a task is feasible if and only if there exists a fixed point, t, of $W_i(t)$ such that t occurs before τ_i's deadline . The following theorem restates their test:

Theorem 1 (from [3]) *In a synchronous periodic (or sporadic) task system, task τ_i is feasible using* DM *if and only if $\exists t \in (0, d_i]$ such that $W_i(t) \leq t$.*

■

Approximate Test

The goal of using a linear approximation in $\delta(\tau_i, t)$ is to bound the number of steps in the approximation function. Since $\delta(\tau_i, t)$ has at most $k - 1$ steps for all τ_i, a *superposition* of $\delta(\tau_i, t)$'s (i.e. summation of δ functions) will have a polynomially bounded number of steps in terms of k and the number of functions in the superposition. The following equation defines a superposition which we will use as the approximate cumulative request-bound function for the approximate feasibility test:

$$\widehat{W}_i(t) \stackrel{\text{def}}{=} e_i + \sum_{\tau_j \in \mathbf{T}_{-i}} \delta(\tau_j, t) \tag{5}$$

In Section 4, we will see that the value of $\widehat{W}_i(t)$ is always at least that of $W_i(t)$. If we use the approximate cumulative request-bound function, $\widehat{W}_i(t)$ to find a fixed point as in Theorem 1, it is possible for the approximate function's smallest fixed point to exceed the exact function's smallest fixed point; therefore, the test is no longer necessary and sufficient. Instead, we will have sufficient test for feasibility tests, as the following theorem states (proved in Section 4):

Theorem 2 *In a synchronous periodic (or sporadic) task system, task τ_i is feasible using* DM *if* $\exists t \in (0, d_i]$ *such that* $\widehat{W}_i(t) \leq t$.

■

Using the approximate cumulative request-bound function also no longer gives an exact check for infeasibility. Instead, if we cannot find a $t \in (0, d_i]$ such that $\widehat{W}_i(t) \leq t$, then τ_i is infeasible on a lower capacity processor. In fact, we can quantify a smaller capacity processor for which this approximate feasibility test would become exact. The following theorem (also proved in Section 4) derives this capacity:

Theorem 3 *If* $\forall t \in (0, d_i]$, $\widehat{W}_i(t) > t$, *then* τ_i *is infeasible using* DM *on a processor of* $(1 - \epsilon)$ *capacity.*

■

The preceding theorem states we must effectively ignore $(1 - \epsilon)$ of the processor capacity for the test to become exact.

Together, theorems 2 and 3 provide an approximate test for feasibility of task system τ. The test is:

> If for all tasks, $\tau_i \in \tau$, there is a time $t \leq d_i$ such that $\widehat{W}_i(t) \leq t$, then τ *is feasible.*
> Otherwise, τ is guaranteed to be infeasible only on a processor of $(1 - \epsilon)$ capacity.

Section 5 will show that the complexity of the approximate test is polynomial-time in terms of the number of tasks and the accuracy parameter, ϵ.

3.3 Example

We now consider an example system τ, to illustrate both the exact and approximate feasibility tests. Let τ be comprised of the following tasks:

$$\begin{aligned} \tau_1 &= (1, 3, 3) \\ \tau_2 &= (2, 5, 5) \\ \tau_3 &= (2, 12, 12) \end{aligned}$$

Let us first demonstrate the exact feasibility test. We must calculate the fixed points of the the cumulative request-bound function (if they exist). The fixed points are: $W_1(1) = 1$, $W_2(3) = 3$, and $W_3(9) = 9$. Since, for all τ_i, there exists a t such that $W_i(t) \leq t$ and $t \in (0, d_i]$, then τ is feasible on a uniprocessor. Figure 2(b) shows the cumulative request-bound function for τ_3. Observe from the graph you can quickly identify the smallest fixed point of $W_3(t)$.

For the approximate test, $\widehat{W}_1(1) = 1$, and $\widehat{W}_2(3) = 3$. However, it is easy to see from Figure 2(a) that there does not exist a $t \in (0, 12]$ such that $\widehat{W}_3(t) \leq t$. Therefore, by Theorem 3, τ is infeasible on a uniprocessor of $(1-\epsilon)$ capacity. The fact that $W_3(t)$ does not fall below the line $f(t) = (1-\epsilon)t$ in Figure 2(b) illustrates the infeasibility of τ on this smaller capacity processor.

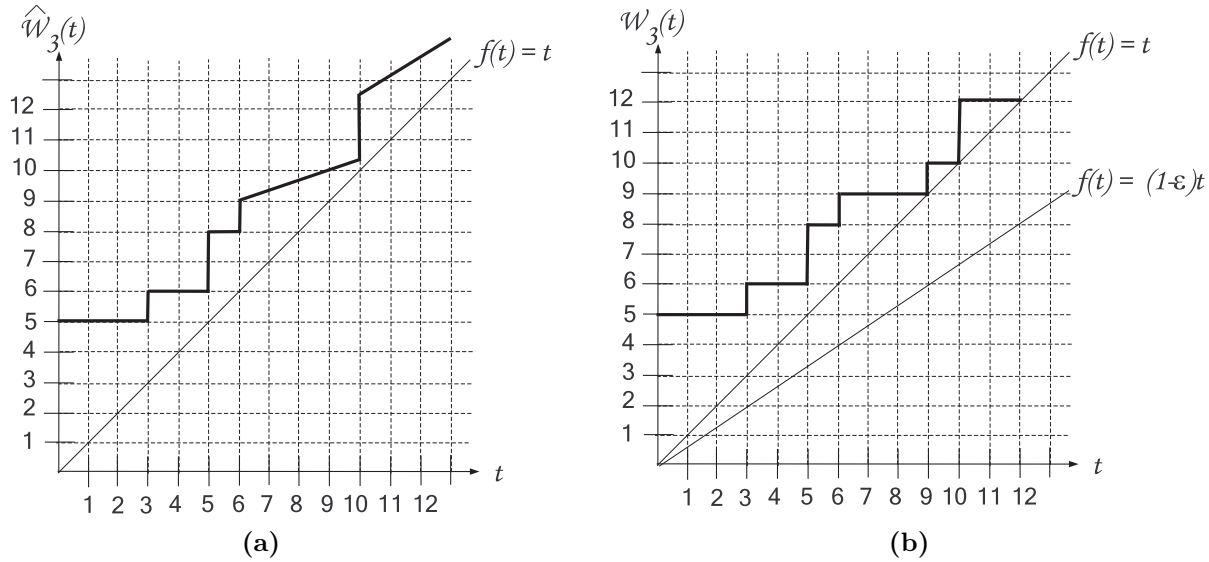

Figure 2: The two graphs above illustrate the *cumulative request-bound functions* of task τ_3 for task system τ where $\tau_1 = (1, 3, 3)$, $\tau_2 = (2, 5, 5)$, and $\tau_3 = (2, 12, 12)$. Graph **(a)** shows the approximate cumulative request-bound function, $\widehat{W}_3(t)$ where $\epsilon = \frac{1}{3}$ (i.e. $k = 2$) . Since there does not exist a $t \in (0, 12)$ such that $\widehat{W}_3(t) \leq t$, the approximate feasibility test declares τ *is infeasible*. Graph **(b)** shows the exact cumulative request-bound function, $W_3(t)$. Notice, that τ is, in fact, infeasible on a uniprocessor of $(1 - \frac{1}{3})$ capacity; however, τ is feasible on the unit capacity uniprocessor.

4 Correctness of Approximation

In this section, we will prove the correctness of Theorems 2 and 3, thereby proving the correctness of the approximate feasibility test. However, before proving the theorems, it will be useful to prove properties of the request-bound functions. Using the request-bound properties, we will be able to prove lemmas about the approximate cumulative request-bound function. From these lemmas and properties, Theorems 2 and 3 will follow.

We prove several properties about $\delta(\tau_i, t)$, which together show that $\delta(\tau_i, t)$ closely tracks $\text{RBF}(\tau_i, t)$. The first property states that our approximation always exceeds or equals RBF.

Property 1 $\forall t \geq 0$, $\delta(\tau_i, t) \geq \text{RBF}(\tau_i, t)$.

Proof: For all $t \in [0, (k-1)p_i]$, $\delta(\tau_i, t) = \text{RBF}(\tau_i, t)$, by definition. For $t > (k-1)p_i$, $\delta(\tau_i, t) = e_i + \frac{te_i}{p_i} = (\frac{t}{p_i} + 1)e_i \geq \lceil \frac{t}{p_i} \rceil e_i = \text{RBF}(\tau_i, t)$

■

The second property states that if the approximation strictly exceeds the RBF at time t, then we have calculated at least $k - 1$ steps of the RBF. We may observe this visually in Figure 1, because the linear approximation does not begin until after the $k - 1$ step. Formally,

Property 2 *If* $\delta(\tau_i, t) > \text{RBF}(\tau_i, t)$, *then* $\text{RBF}(\tau_i, t) \geq ke_i$.

Proof: $\delta(\tau_i, t) > \text{RBF}(\tau_i, t)$ implies that $t > (k-1)p_i$. Thus, $\text{RBF}(\tau_i, t) = \lceil \frac{t}{p_i} \rceil e_i > \left\lceil \frac{(k-1)p_i}{p_i} \right\rceil e_i = (k-1)e_i$. Since RBF increases by e_i at each step, $\text{RBF}(\tau_i, t) \geq ke_i$.
■

The third property reflects the fact that the approximation never exceeds the RBF by more than one step size, e_i. Again, visually this is easy to see, since after $k-1$ steps, $\delta(\tau_i, t)$ is a linear interpolation of the "edges" of the steps.

Property 3 $\forall t \geq 0$, $\delta(\tau_i, t) - \text{RBF}(\tau_i, t) \leq e_i$.

Proof: For all $t \in [0, (k-1)p_i]$, $\delta(\tau_i, t) - \text{RBF}(\tau_i, t) = 0 \leq e_i$. For $t > (k-1)p_i$, $\delta(\tau_i, t) - \text{RBF}(\tau_i, t) = \frac{te_i}{p_i} + e_i - \left\lceil \frac{t}{p_i} \right\rceil e_i \leq \frac{te_i}{p_i} + e_i - \frac{te_i}{p_i} = e_i$.
■

Property 4, below, bounds the ratio of the value of the approximate function at time t to the value of the exact function:

Property 4 $\forall t \geq 0$, $\text{RBF}(\tau_i, t) \leq \delta(\tau_i, t) \leq (1 + \frac{1}{k})\text{RBF}(\tau_i, t)$.

Proof: $\text{RBF}(\tau_i, t) \leq \delta(\tau_i, t)$ follows from Property 1. By Property 3, $\delta(\tau_i, t) \leq \text{RBF}(\tau_i, t) + e_i$. Then by Property 2, $\text{RBF}(\tau_i, t) + e_i \leq \text{RBF}(\tau_i, t) + \frac{\text{RBF}(\tau_i, t)}{k}$. This implies, $\forall t \geq 0$, $\delta(\tau_i, t) \leq (1 + \frac{1}{k})\text{RBF}(\tau_i, t)$.
■

The following lemmas describe the implications of using the approximation function δ in the cumulative request-bound function. Informally, Lemma 1 states that if the approximate cumulative request-bound function is below line $f(t) = t$, then the exact cumulative request-bound function must be below as well.

Lemma 1 *If* $\widehat{W}_i(t) \leq t$, *then* $W_i(t) \leq t$.

Proof:

By Property 1, $\sum_{\tau_j \in \mathbf{T}_{-i}} \delta(\tau_j, t) \geq \sum_{\tau_j \in \mathbf{T}_{-i}} \text{RBF}(\tau_j, t)$. Thus, $e_i + \sum_{\tau_j \in \mathbf{T}_{-i}} \delta(\tau_j, t) \leq t \Rightarrow e_i + \sum_{\tau_j \in \mathbf{T}_{-i}} \text{RBF}(\tau_j, t) \leq t$. The Lemma follows from the definitions of $W_i(t)$ and $\widehat{W}_i(t)$.
■

Lemma 2 states that if the approximate cumulative request-bound function lies above $f(t) = t$, then the exact cumulative request-bound function must lie above the line $f(t) = \frac{k}{1+k}(t)$. Formally stated:

Lemma 2 *If* $\widehat{W}_i(t) > t$, *then* $W_i(t) > \frac{k}{1+k}(t)$.

Proof:

$$\begin{aligned}
&\widehat{W}_i(t) = e_i + \textstyle\sum_{\tau_j \in \mathbf{T}_{-i}} \delta(\tau_j, t) > t \\
&\Rightarrow e_i + \textstyle\sum_{\tau_j \in \mathbf{T}_{-i}} (1 + \frac{1}{k})\text{RBF}(\tau_j, t) > t \qquad \text{(by Property 4)} \\
&\Rightarrow (1 + \tfrac{1}{k})(e_i + \textstyle\sum_{\tau_j \in \mathbf{T}_{-i}} \text{RBF}(\tau_j, t)) > t \\
&\Rightarrow e_i + \textstyle\sum_{\tau_j \in \mathbf{T}_{-i}} \text{RBF}(\tau_j, t) > \frac{k}{1+k}(t)
\end{aligned}$$

The Lemma follows from definition of $W_i(t)$.

■

We are now prepared to prove Theorems 2 and 3, originally stated in Section 3.

Proof of Theorem 2 Assume there exists a $t_0 \in (0, d_i]$ such that $\widehat{W}_i(t_0) \leq t_0$. Then, by Lemma 1, $W_i(t_0) \leq t_0$. From Theorem 1, τ_i is schedulable using DM.

■

Proof of Theorem 3 Assume that for all $t \in (0, d_i]$, $\widehat{W}_i(t) > t$, but τ_i is still feasible on a processor of $(1-\epsilon)$ capacity. Notice that $1 - \frac{k}{k+1} \leq \epsilon$. So, by Theorem 1, $\exists t_0 \in (0, d_i]$ such that $W_i(t) \leq (1-\epsilon)t \leq (\frac{k}{k+1})t$. But, $\widehat{W}_i(t) > t \Rightarrow W_i(t) > (\frac{k}{k+1})t$ by Lemma 2. This is a contradiction; therefore, τ_i is infeasible on a processor of capacity $(1-\epsilon)$.

■

5 Testing Set

We now turn our attention to the most significant benefit of the approximate feasibility test: its polynomial time complexity. In order to prove that the test runs in polynomial time, we show that the number of points at which equation (5) must be evaluated is bounded by a polynomial in terms of n and ϵ.

As a basis of comparison, let us first consider the testing set for the exact feasibility test. It is known [2] that the time-instants at which the condition of Theorem 1 must be evaluated is:

$$S_i \stackrel{\text{def}}{=} \left\{ t = bp_a : a = 1, \ldots, i; b = 1, \ldots, \left\lfloor \frac{d_i}{p_a} \right\rfloor \right\} \tag{6}$$

Informally, these are the only values of t where $W_j(t)$ changes value, for some j, $1 \leq j \leq i$. The number of points in this set may be as large as:

$$\sum_{j=1}^{i} \left\lfloor \frac{d_i}{p_j} \right\rfloor \tag{7}$$

Since the number of points that must be checked is dependent on the task periods, this represents a pseudo-polynomial feasibility test.

In the remainder of this section, we will show that if we were to use the approximation $\widehat{W}_i(t)$ in place of $W_i(t)$, we would only need test the condition of Theorem 2 at *polynomially* many points. In particular, we will show that the set of points that must be checked is:

$$\widehat{S}_i \stackrel{\text{def}}{=} \{t = bp_a : a = 1, \ldots, i-1; b = 1, \ldots, k-1\} \cup \{t = d_i\} \tag{8}$$

Necessity and Sufficiency of Testing Set

In the following argument, we will show that $\widehat{S}_i$ is a sufficient number of points to determine the existence (or absence) of a time t such that $\widehat{W}_i(t) \leq t$. Our proof obligation is to show: if for all $t \in \widehat{S}_i$, $\widehat{W}_i(t) > t$, then for all $t \in (0, d_i]$, $\widehat{W}_i(t) > t$.

We call two elements t_1 and t_2 ($t_1 < t_2$) in set $\widehat{S}_i$ *adjacent* if no t satisfying $t_1 < t < t_2$ is in $\widehat{S}_i$. In order to satisfy our proof obligation, we will show that for any pair of adjacent points in $\widehat{S}_i$, if the value of $\widehat{W}_i$ at the adjacent points lies above $f(t) = t$, then the value of $\widehat{W}_i$ at all points in between the adjacent points are above $f(t) = t$.

Define a linear function, $g_{(t_1,t_2)}(t)$ over this interval as follows,

$$g_{(t_1,t_2)}(t) \stackrel{\text{def}}{=} \frac{\widehat{W}_i(t_2) - \widehat{W}_i(t_1)}{t_2 - t_1}(t - t_1) + \widehat{W}_i(t_1), \forall t \in (t_1, t_2) \tag{9}$$

Intuitively, $g_{(t_1,t_2)}(t)$ is a linear interpolation of points $(t_1, \widehat{W}_i(t_1))$ and $(t_2, \widehat{W}_i(t_2))$. For example, in Figure 2a consider the two adjacent points $t_1 = 6$ and $t_2 = 10$. $g_{(t_1,t_2)}(t)$ is the line defined by points $(6, 8)$ and $(10, 10\frac{1}{3})$.

The following claim shows that between any two adjacent points, the linear function, $g_{(t_1,t_2)}(t)$ is less than the approximation function. So, $g_{(t_1,t_2)}(t)$ bounds the approximation function from below over the interval (t_1, t_2). Let $r_i(t)$ be the total execution of all tasks of priority τ_i or greater that release jobs at time t. Formally,

$$r_i(t) \stackrel{\text{def}}{=} \sum_{\{\tau_j \in \mathbf{T}_{-i} | p_j \text{ divides } t\}} e_j \tag{10}$$

Claim 1 *For adjacent elements, t_1, $t_2 \in \widehat{S}_i$ $(t_1 < t_2)$::*$\Big(g_{(t_1,t_2)}(t) < \widehat{W}_i(t),\ \forall t \in (t_1, t_2)\Big)$.

Proof: We know that $g_{(t_1,t_2)}(t_1) = \widehat{W}_i(t_1)$ and $g_{(t_1,t_2)}(t_2) = \widehat{W}_i(t_2)$ by definition of $g_{(t_1,t_2)}(t)$. Observe that the "height" of the step for $\widehat{W}_i(t)$ is $r_i(t_S) > 0$ at every point t_S in $\widehat{S}_i$. Also, note that $\widehat{W}_i(t)$ is linear in interval (t_1, t_2). Then, for $t \in (t_1, t_2)$, $\widehat{W}_i(t)$ is a line that passes through $(t_1, \widehat{W}_i(t_1) + r_i(t_1))$ and $(t_2, \widehat{W}_i(t_2))$. Thus, $\widehat{W}_i(t) = \frac{\widehat{W}_i(t_2) - \widehat{W}_i(t_1) - r_i(t_1)}{t_2 - t_1}(t - t_1) + \widehat{W}_i(t_1) + r_i(t_1)$. Notice that $\widehat{W}_i(t) - g_{(t_1,t_2)}(t) = r_i(t_1)\left(\frac{t_2 - t}{t_2 - t_1}\right) > 0$. This implies, $\widehat{W}_i(t) > g_{(t_1,t_2)}(t)$.
■

The next lemma shows that for any adjacent elements of $\widehat{S}_i$, if the value of $\widehat{W}_i$ lies above the line $f(t) = t$, then the value of $\widehat{W}_i$ of all points in between the adjacent elements must lie above $f(t) = t$.

Lemma 3 *For adjacent elements, t_1, $t_2 \in \widehat{S}_i$, if $\widehat{W}_i(t_1) > t_1$ and $\widehat{W}_i(t_2) > t_2$, then $\widehat{W}_i(t) > t$, $\forall t \in (t_1, t_2)$.*

Proof: Notice that $g_{(t_1,t_2)}(t_1) = \widehat{W}_i(t_1) > t_1$ and $g_{(t_1,t_2)}(t_2) = \widehat{W}_i(t_2) > t_2$. Now consider function $g_{(t_1,t_2)}(t)$ over the interval (t_1, t_2). Assume that there exist $t' \in (t_1, t_2)$, such that $g_{(t_1,t_2)}(t') \leq t'$. Since, $g_{(t_1,t_2)}(t)$ is linear the slope from t_1 to t' must equal the slope from t' to t_2. But,

$$\begin{aligned} & \widehat{W}_i(t') \leq t' \\ \Rightarrow\ & \tfrac{\widehat{W}_i(t') - t_1}{t' - t_1} \leq 1 \\ \Rightarrow\ & \tfrac{\widehat{W}_i(t') - \widehat{W}_i(t_1)}{t' - t_1} < 1, \end{aligned}$$

and

$$\begin{aligned} & -\widehat{W}_i(t') \geq -t' \\ \Rightarrow & \frac{t_2 - \widehat{W}_i(t')}{t_2 - t'} \geq 1 \\ \Rightarrow & \frac{\widehat{W}_i(t_2) - \widehat{W}_i(t')}{t_2 - t'} > 1. \end{aligned}$$

The slopes are unequal and we have reached a contradiction. Our assumption that there exists a t' such that $g_{(t_1,t_2)}(t') \leq t'$ is incorrect. Therefore, $\forall t \in (t_1, t_2)$, $g_{(t_1,t_2)}(t) > t$. By Claim 1, $\widehat{W}_i(t) > t$, $\forall t \in (t_1, t_2)$.

■

The following theorem proves $\widehat{S}_i$ is a necessary and sufficient testing set for our approximation.

Theorem 4 *There exists* $t \in (0, d_i]$ *such that* $\widehat{W}_i(t) \leq t$ if and only if *there exists* $t' \in \widehat{S}_i$ *such that* $\widehat{W}_i(t') \leq t'$.

Proof: The "if" direction is obvious since $\widehat{S}_i \subset (0, d_i]$. Therefore, we will concentrate on the "only if" direction of the proof. If there exists a $t \in (0, p_{min})$ such that $\widehat{W}_i(t) \leq t$ (where p_{min} is the smallest period), then $\widehat{W}_i(p_{min}) \leq p_{min}$ and $p_{min} \in \widehat{S}_i$. So, assume there exists a $t \in [p_{min}, d_i]$ such that $\widehat{W}_i(t) \leq t$, but $\nexists t' \in \widehat{S}_i$ such that $\widehat{W}_i(t') \leq t'$. Since $\widehat{S}_i \subset [p_{min}, d_i]$, there exists adjacent elements t_1 and t_2 of $\widehat{S}_i$ such that $t \in (t_1, t_2)$. But, by Lemma 3, for all $t \in (t_1, t_2)$, $\widehat{W}_i(t) > t$. This a contradiction; therefore, our assumption that there did not exist a $t' \in \widehat{S}_i$ such that $\widehat{W}_i(t') \leq t'$ is incorrect.

■

Complexity

It is easy to see that the number of items in the testing set for τ_i is at most:

$$1 + (i-1)(k-1) \tag{11}$$

In a naive implementation of approximate feasibility-analysis, the condition in Theorem 2 would be evaluated at most $\Sigma_{i=1}^{n}(1 + (i-1)(k-1))$ times, which is $\mathcal{O}(n^2 k)$; a smarter implementation (see, e.g, [8]) would take $\mathcal{O}(nk \log n)$ time.

For a given accuracy, ϵ, the running time of the approximation algorithm is $\mathcal{O}(n^2/\epsilon)$. Thus, the algorithm is a member of a family of algorithms that collectively represent a polynomial-time approximation scheme for uniprocessor feasibility analysis for both synchronous periodic and sporadic task systems in a static-priority system. The following theorem states this formally.

Theorem 5 *For any* ϵ *in the range* $(0, 1)$*, there is an algorithm* A_ϵ *that has run-time* $\mathcal{O}(n^2/\epsilon)$ *and exhibits the following behavior: On any synchronous periodic or sporadic task system* τ*,*

- if τ *is* DM*-infeasible on a unit-capacity processor* then *Algorithm* A_ϵ *correctly identifies it as being* DM*-infeasible;*
- if τ *is* DM*-feasible on a processor of computing capacity* $(1 - \epsilon)$ then *Algorithm* A_ϵ *correctly identifies it as being* DM*-feasible;*

- else *Algorithm A_ϵ may identify τ as being either feasible or infeasible.*

■

6 Future Work: Arbitrary Relative Deadlines

When deadlines can exceed periods, Lehoczky [5] shows that it is no longer sufficient to check the response-times of only the first job of each task. Instead, it is potentially necessary to check the response-time of all jobs in the *level-i busy interval* for each task τ_i . A level-i busy interval is a time interval $[a, b]$ where only jobs of $\mathbf{T}_i = \mathbf{T}_{-i} \cup \{\tau_i\}$ are executing continuously and the following is true:

1. A job of $\mathbf{T}_i$ is released at time a.
2. All jobs of $\mathbf{T}_i$ released prior to a have completed by time a.
3. b is the first time instant where all jobs of $\mathbf{T}_i$ released after a have completed.

It may be tempting to try extending our results to a task system with arbitrary deadlines by applying the approximate feasibility test presented in this paper to each job of τ_i in the level-i busy interval. Unfortunately, if this approach is used the approximate feasibility test is no longer polynomial in terms of n and ϵ. Applying the approximate feasibility test to each job of τ_i in the level-i busy interval results in a *pseudo-polynomial* time test. The reasons that the test is pseudo-polynomial are the following:

- The length of the level-i busy interval does not depend on n, but on the p_i and e_i terms; therefore, the level-i busy interval contains a pseudo-polynomial number of jobs of τ_i. Applying the approximate feasibility test in this paper would require running the test a pseudo-polynomial number of times.
- The number of jobs at each time instant t that are *active* (i.e. t lies between the job's release time and absolute deadline) could be $\Theta(d_i/p_i)$. Again, this is not polynomial in terms of n and ϵ. Therefore, at each point in the testing set, we may have to perform computation for a pseudo-polynomial number of active jobs to check if any of them have missed their deadline.

We are currently working on techniques that do not require these pseudo-polynomial time checks. We conjecture that there exists a PTAS for feasibility analysis in static-priority systems with arbitrary relative deadlines. We further conjecture that the PTAS for arbitrary relative deadlines can achieve the same asymptotic time complexity as the PTAS for bounded relative deadlines.

7 Conclusions

It has be shown [1] that there exists a polynomial-time approximation scheme (PTAS) for feasibility analysis of sporadic task sets in dynamic-priority systems. We have constructed a similar PTAS for static-priority feasibility analysis of uniprocessor synchronous periodic and sporadic task systems, in which all tasks have their relative deadlines no larger than their periods. Our PTAS has a run-time asymptotically identical to the test of [1]. We have, thus, shown that dynamic- and static-priority systems have equivalent approximate feasibility-analysis "tools" available. To further strengthen our claim that dynamic- and static-priority systems have similar feasibility-analysis tools available, we are currently extending our results to a model that allows arbitrary response times (i.e. the relation between deadlines and periods is arbitrary).

The polynomial-time approximation tests presented in this paper and in future work offer a reduction in complexity for feasibility tests. These approximate feasibility tests may be useful for quick estimates of task system feasibility in automatic system-synthesis tools.

References

[1] Albers, K., and Slomka, F. An event stream driven approximation for the analysis of real-time systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems* (Catania, Sicily, July 2004), IEEE Computer Society Press, pp. 187–195.

[2] Audsley, N., Burns, A., Richardson, M., Tindell, K., and Wellings, A. Applying new scheduling theory to static priority preemptive scheduling. *Software Engineering Journal 8*, 5 (1993), 285–292.

[3] Audsley, N. C., Burns, A., Richardson, M. F., and Wellings, A. J. Hard Real-Time Scheduling: The Deadline Monotonic Approach. In *Proceedings 8th IEEE Workshop on Real-Time Operating Systems and Software* (Atalanta, May 1991).

[4] Lehoczky, J., Sha, L., and Ding, Y. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the Real-Time Systems Symposium - 1989* (Santa Monica, California, USA, Dec. 1989), IEEE Computer Society Press, pp. 166–171.

[5] Lehoczky, J. P. Fixed priority scheduling of periodic tasks with arbitrary deadlines. In *IEEE Real-Time Systems Symposium* (Dec. 1990), pp. 201–209.

[6] Leung, J., and Whitehead, J. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation 2* (1982), 237–250.

[7] Liu, C., and Layland, J. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM 20*, 1 (1973), 46–61.

[8] Mok, A. Task management techniques for enforcing ED scheduling on a periodic task set. In *Proc. 5th IEEE Workshop on Real-Time Software and Operating Systems* (Washington D.C., May 1988), pp. 42–46.