A Constant-Approximate Feasibility Test for Multiprocessor Real-Time Scheduling*

Vincenzo Bonifaci† Alberto Marchetti-Spaccamela‡ Sebastian Stiller§

Abstract

We devise an approximate feasibility test for sporadic multiprocessor real-time scheduling. We give an algorithm that, given a task system and $\epsilon > 0$, correctly decides either that the task system can be scheduled using the Earliest Deadline First algorithm on m speed-$(2 - 1/m + \epsilon)$ machines, or that the system is not schedulable by any algorithm on m unit speed machines. This error bound is known to be best possible for EDF. The running time of the algorithm is polynomial in the size of the task system and $1/\epsilon$. We also provide a generalized tight bound that trades off speed with additional machines.

Keywords: real-time scheduling, sporadic task system, feasibility test, Earliest Deadline First, approximation algorithm

1 Introduction

We study the problem of scheduling recurring processes, or tasks, on a multiprocessor platform. An instance of the problem is given by a finite set $\mathbb{T}$ of tasks which need to be executed by the system; each task can generate an arbitrarily long or even infinite sequence of jobs.

In the *periodic* version of the problem, a task $\tau \in \mathbb{T}$ is characterized by a quadruple of positive integers: an offset o_τ that represents the time instant when the first job generated by the task is released, an execution time C_τ, a relative deadline D_τ and a period T_τ. Each occurrence of task τ is represented by a job: the k-th occurrence of the task is released at time $o_\tau + (k-1)T_\tau$, requires C_τ units of processor time and must complete its execution before time $o_\tau + (k-1)T_\tau + D_\tau$. Note that a periodic task system defines a single, infinite sequence of jobs.

In the *sporadic* case, each task is characterized by a triple (C_τ, D_τ, T_τ) where C_τ, D_τ have the same meaning as in the periodic case, while T_τ denotes the *minimum* separation between successive occurrences of the task. Note that in a sporadic task system the time instant when the next invocation of a task will be released after the minimal separation time has elapsed is unknown. Thus, in contrast with the periodic case, there are infinitely many job sequences that conform to the system's specification.

The correctness of a hard real-time system requires that all jobs complete by their deadlines. A task system is *feasible* if every sequence of jobs that is consistent with the parameters of the task system admits a schedule meeting all the deadlines, and it is *schedulable* by a given algorithm if the algorithm constructs a feasible schedule for every such sequence of jobs. In the sequel we focus on preemptive scheduling algorithms that are allowed to interrupt the execution of a job and resume it later; job migration is also allowed.

A *feasibility test* for a task system is an algorithm that takes as input a description of the task system and answers whether the system is feasible or not. A feasibility test is *exact* if it correctly identifies all feasible and infeasible task systems and *sufficient* if it correctly identifies all infeasible task systems, but may give a wrong answer for feasible task systems. A sufficient feasibility test is a natural requirement in hard-deadline real-time applications. In fact, from a practical point of view, there is no difference between a task system that is not feasible and one that cannot be proven to be feasible.

In the case of a single machine, the problem has been widely studied and effective scheduling algorithms are quite well understood [7, 11, 16]. In this paper we study the feasibility problem for sporadic

*A preliminary version of this work appeared in *Proceedings of the 16th European Symposium on Algorithms*, Lecture Notes in Computer Science, Springer, Berlin, 2008, pp. 210–221.

†Max-Planck-Institut für Informatik, Saarbrücken, Germany, email bonifaci@mpi-inf.mpg.de.

‡Sapienza Università di Roma, Italy, email alberto@dis.uniroma1.it.

§Technische Universität Berlin, Germany, email stiller@math.tu-berlin.de

task systems on identical parallel machines. The problem is not only interesting from a theoretical point of view but is also relevant in practice. In fact, real-time multiprocessor systems are becoming common, both in the form of single-chip architectures characterized by a small number of processors, and of large-scale signal-processing systems with many processing units.

Single machine scheduling. In the case of a single machine it is known [7, 10, 16] that the Earliest Deadline First scheduling algorithm (EDF), which at each instant in time schedules the available job with the earliest absolute deadline (with ties broken arbitrarily), is an *optimal* scheduling algorithm for scheduling a sporadic (or periodic) task system, in the following sense: if it is possible to schedule a given collection of jobs such that all the jobs meet their deadlines, then the schedule generated by EDF for this collection of jobs will meet all deadlines as well. Despite this positive result, the feasibility problem for periodic or sporadic task systems on one processor is coNP-hard [7, 12, 15].

For this reason, approximate feasibility tests have been proposed that run efficiently (say, in time polynomial in the description of the system) but introduce a small error in the decision process, controlled by an accuracy parameter. Such approaches have been developed for EDF scheduling and for other scheduling algorithms. Two different paradigms can be used to define approximate feasibility tests: pessimistic and optimistic. If a pessimistic feasibility test returns "feasible", then the task system is guaranteed to be feasible. If the test returns "infeasible", the task system is guaranteed to be infeasible on a slower processor, of computing capacity $(1-\epsilon)$, where ϵ denotes the error parameter. Conversely, if an optimistic test returns "feasible", then the task system is guaranteed to be feasible on a $(1+\epsilon)$-speed processor. If the test returns "infeasible", the task system is guaranteed to be infeasible on a unit speed processor [9]. These two approaches are in fact equivalent, as by scaling the input parameters it is possible to turn an optimistic test into a pessimistic one, or vice versa.

Fully polynomial-time approximation schemes (FPTAS) are known for a single processor. That is, for any $\epsilon > 0$ there exists a feasibility test that returns an ϵ-approximated answer, and the running time of the algorithm is polynomial in the input size of the task system and in $1/\epsilon$ (see for example [1, 2, 9, 13] and references therein).

Finally, we remark that in the uniprocessor case the sporadic feasibility problem is known to reduce to a special case of the periodic feasibility problem in which all tasks have zero offset (i.e. each task releases its first job at time zero) [7]. The fact that this special case remains coNP-hard was established recently [12].

Multiple machine scheduling. We first observe that in the multiprocessor case the previously mentioned analogy between sporadic and periodic problems does not hold; that is, the sporadic feasibility problem cannot be reduced to the periodic feasibility problem by setting the tasks' offsets to zero (Figure 1).

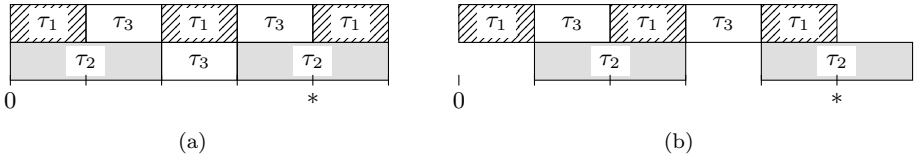

Figure 1: In this example, $(C_1, D_1, T_1) = (1, 1, 2)$, $(C_2, D_2, T_2) = (2, 2, 3)$, $(C_3, D_3, T_3) = (3, 4, 6)$. There are two processors. In case (a), all three tasks are activated simultaneously at time zero and the periodic sequence is schedulable. In case (b), tasks 2 and 3 are activated one time unit later and task 3 cannot be completed within its deadline (marked with $*$).

Moreover, in the multiprocessor case EDF is no longer an optimal algorithm and in fact no optimal scheduling algorithm is known. However, it is known that any feasible task system on m unit speed machines is EDF-schedulable on m machines of speed $2 - 1/m$ [17]. This result holds for EDF as well as for other scheduling algorithms. For EDF this result is tight, while for other algorithms it has not been improved since then. Subsequent work has analyzed the advantage of trading speed for machines [14], while further work on conditions for EDF-schedulability has been done by Baker [3].

Note that the result of [17] *does not* imply an efficient test for deciding when EDF (possibly with extra speed) can schedule a sporadic task system. Thus, the main open problem in order to apply the result of Phillips et al. [17] is the lack of an efficient feasibility test.

This problem attracted attention in recent years (see e.g. [4] and references therein for a thorough presentation). A number of special cases have also been studied; for example, when for each task the

deadline is equal to the period (*implicit-deadline* task systems), it is known that the condition

$$\sum_{\tau \in \mathbb{T}} \frac{C_\tau}{T_\tau} \leq m \text{ and } \max_{\tau \in \mathbb{T}} \frac{C_\tau}{T_\tau} \leq 1$$

gives a necessary and sufficient test for the feasibility of the system.

However, not much was known regarding the feasibility of an arbitrary-deadline task system. In a recent work, Baruah and Baker [6] gave a 2.62-approximate feasibility test in the case of constrained-deadline sporadic task systems, that is, task systems in which for each task the deadline is at most equal to the period. The same guarantee was later extended to arbitrary-deadline sporadic task systems [5]. For periodic task systems, it turns out that no efficient approximate feasibility test is possible unless P = NP [8]. We refer the reader to the survey [4] for feasibility tests that are known for other special cases.

Our contribution. We give an efficient approximate feasibility test for arbitrary-deadline sporadic task systems with an approximation guarantee of $2 - 1/m + \epsilon$, for any fixed $\epsilon > 0$. More precisely, we give a test that, given a sporadic task system $\mathbb{T}$ and $\epsilon > 0$, decides either that the system can be scheduled by EDF on m speed-$(2 - 1/m + \epsilon)$ machines, or that the instance violates at least one of two basic conditions which are both necessary for feasibility on m unit speed machines. The running time is polynomial in the input size of $\mathbb{T}$ and $1/\epsilon$. In fact we give a slightly more general result, allowing to trade some extra speed for extra machines. Note that extra machines are in general less powerful than extra speed.

One of the two basic conditions of our test is trivial. The other condition is the crucial one; it provides a lower bound on the processing requirement of an arbitrary time interval. We call this processing requirement the *forward forced demand*. This concept is strong enough to approximately capture the feasibility of scheduling a sporadic task system on a multiprocessor platform, but also simple enough to be approximated in polynomial time up to an arbitrarily small $\epsilon > 0$: we give an algorithm that checks this condition in time polynomial in the input size of $\mathbb{T}$ and $1/\epsilon$, for any desired error bound $\epsilon > 0$.

Apart from improving the bound in [5], our result is tight from the point of view of EDF-schedulability. It might be possible to improve the bound by considering schedulability with respect to an algorithm different from EDF, but notice that such an improvement would have to break the long standing bound of Phillips et al. [17]. As a byproduct of our analysis, we also obtain an alternative proof of the optimality of EDF for uniprocessor systems.

Outline of the paper. The rest of the paper is structured as follows. In Section 2 we give the necessary definitions and introduce the concept of forced forward demand. Section 3 provides the main connection between forced forward demand and EDF-schedulability. In Section 4 we complete the description of the feasibility test by providing a fully polynomial time approximation scheme to estimate the worst-case forced forward demand of a sporadic task system.

2 Preliminaries

An instance of the feasibility problem for sporadic task systems is a finite set of tasks $\mathbb{T}$. Each task $\tau \in \mathbb{T}$ is defined by three positive integers: an *execution time* C_τ, a *relative deadline* D_τ and a *minimum separation time* T_τ. Every task may generate jobs from time to time. A job j is defined by the task $\tau(j)$ it belongs to and by a release date r_j, a nonnegative integer. We abbreviate $C_j := C_{\tau(j)}$, $D_j := D_{\tau(j)}$ and $T_j := T_{\tau(j)}$, and we call $d_j := r_j + D_j$ the *absolute deadline* of job j. Each job j requires a processor to be allocated to it for C_j time units during the interval $[r_j, r_j + D_j)$. Jobs can be preempted and migrated without penalty, but each job may execute on at most one processor at any given instant in time, and jobs from the same task cannot be processed in parallel.

A sporadic *job sequence* R is any countable set of jobs from tasks in $\mathbb{T}$ with the following property: any distinct jobs j and k from the same task τ satisfy $|r_j - r_k| \geq T_\tau$. Job sequence R is *schedulable* when there exists a schedule satisfying the execution requirements of all jobs in R. A task system $\mathbb{T}$ is *feasible* if all job sequences from $\mathbb{T}$ are schedulable.

Given a real number x we denote by x^+ its positive part, that is $x^+ := \max(x, 0)$.

Definition 2.1. Let $\mathbb{T}$ be a task system, R a job sequence, $\Delta = [t, t')$ an interval, τ a task and j a job. We define the following quantities.

$$\mathrm{FFD}(j, \Delta) := \begin{cases} (C_j - (t - r_j)^+)^+ & \text{if } d_j \in \Delta \text{ or } d_j = t', \\ 0 & \text{otherwise.} \end{cases}$$

$$\mathrm{FFD}_R(\tau, \Delta) := \sum_{j \in R:\, \tau(j) = \tau} \mathrm{FFD}(j, \Delta).$$

$$\mathrm{FFD}_R(\Delta) := \sum_{\tau \in \mathbb{T}} \mathrm{FFD}_R(\tau, \Delta).$$

Quantity $\mathrm{FFD}(j, \Delta)$ is called the *forward forced demand* of j in Δ. Similarly, the quantity $\mathrm{FFD}_R(\Delta)$ is called the forward forced demand of the interval. We write $\mathrm{FFD}(\tau, \Delta)$ and $\mathrm{FFD}(\Delta)$, respectively, when the task system $\mathbb{T}$ and the sequence R are implicitly fixed. The quantity $\mathrm{FFD}(\Delta)/\left\|\Delta\right\|$ is called the *load* of interval Δ.

Note that when $r_j \in \Delta$ the forward forced demand equals the execution time of the job. If $C_\tau \leq T_\tau$ for all tasks τ, then each task τ can have at most one job with release date outside the interval that has strictly positive forward forced demand in the interval.

The following proposition easily follows from the fact that the forward forced demand of a job in an interval is a lower bound on the amount of work that has to be performed on the job during the interval, if the job's deadline has to be met.

Proposition 2.1. *Let R, Δ and m be such that $\mathrm{FFD}_R(\Delta) > m\left\|\Delta\right\|$. Then R is not schedulable on m unit speed machines.*

Proposition 2.2. *Let $\tau \in \mathbb{T}$ be such that $C_\tau > \min(D_\tau, T_\tau)$. Then $\mathbb{T}$ is not feasible on any number of unit speed machines.*

Proof. If $C_\tau > D_\tau$, then the sequence consisting of a single τ-job is clearly not schedulable (recall that a job can be processed on at most one processor at any given time). If $C_\tau > T_\tau$, then any sufficiently long periodic sequence of τ-jobs with period T_τ is not schedulable (jobs from the same task have to be processed sequentially). □

We use the notation $\mathrm{EDF}[M, \sigma]$ to denote the Earliest Deadline First scheduling algorithm executed on M speed-σ machines. We do not assume any particular tie-breaking rule for the algorithm. A job sequence R is EDF-*schedulable* if, on input R, EDF produces a schedule that meets the requirements of all jobs in R. A task system $\mathbb{T}$ is EDF-schedulable if every job sequence from $\mathbb{T}$ is EDF-schedulable.

Definition 2.2. Let $\mathbb{T}$ be a task system and R a job sequence. For a task $\tau \in \mathbb{T}$, an interval $\Delta = [t', t)$ is called *τ-busy before t* if executing algorithm $\mathrm{EDF}[M, \sigma]$ on the sequence R yields at any time in Δ a positive remaining execution time for at least one of the jobs of task τ.

Observe that when at time t there is some pending job from task τ, the maximal τ-busy interval before t is well-defined, unique, and starts with the release date of some job of τ. Moreover, all the execution requirements from τ-jobs released before a maximal τ-busy interval Δ are satisfied by EDF strictly before Δ.

3 A feasibility test

In this section we present and discuss the conditions that will be used for testing feasibility of a sporadic task system. The main result of this section is the following.

Theorem 3.1. *Let M be a positive integer and $\sigma \geq 1$. Consider a task system $\mathbb{T}$ satisfying $C_\tau \leq \min(D_\tau, T_\tau)$ for all $\tau \in \mathbb{T}$. If R is a job sequence which cannot be scheduled by* EDF *on M speed-σ machines, then there is an interval Δ such that $\mathrm{FFD}_R(\Delta)/\left\|\Delta\right\| > M(\sigma - 1) + 1$.*

Before giving the proof we explain the main intuition. Given a job sequence on which EDF fails, we will inductively construct an interval with high load; this interval will be a witness of the non-schedulability of R. The interval will be composed of several subintervals, each of which will be τ-busy for some appropriate task τ. Whenever EDF does not process a job of τ in the subinterval, it must have all machines busy.

Algorithm 1 Construction of an Infeasibility Certificate

$j_1 :=$ job that missed its deadline at t_0
$\overline{\Delta}_1 := [t_1, t_0) :=$ maximal $\tau(j_1)$-busy interval before t_0
$\Delta_1 := \overline{\Delta}_1$
$X_1 :=$ time intervals in $\overline{\Delta}_1$ when a job of task $\tau(j_1)$ is being processed
$Y_1 := \overline{\Delta}_1 \setminus X_1$
$i := 2$
repeat
Let j_i denote a job such that:

(i) the release date of j_i is strictly before t_{i-1};

(ii) $\mathrm{EDF}\left(j_i, \bigcup_{s=1}^{i-1} Y_s\right) > \mathrm{FFD}(j_i, \Delta_{i-1})$.

$\overline{\Delta}_i := [t_i, t_{i-1}) :=$ maximal $\tau(j_i)$-busy interval before t_{i-1}
$\Delta_i := [t_i, t_0)$
$X_i :=$ time intervals in $\overline{\Delta}_i$ when a job of task $\tau(j_i)$ is being processed
$Y_i := \overline{\Delta}_i \setminus X_i$
$i := i + 1$
until no such job exists

In order to conclude that the load of the whole interval is large, we establish two facts: first, that the fraction of a subinterval in which its associated task is processed is small, i.e., in a large part of the subinterval all machines must be busy; second, that what is processed in those busy subintervals is part of the forward forced demand of the witness interval.

From now on we assume that R is a job sequence which cannot be scheduled by $\mathrm{EDF}[M, \sigma]$, and that t_0 is the first point in time when EDF fails a deadline. We can without loss of generality assume that no job has absolute deadline after t_0: after removing such jobs, EDF still fails some deadline at t_0 and the forward forced demand has not increased.

We define inductively a finite sequence of pairs (t_i, j_i), for $1 \leq i \leq k$, where t_i is a time and j_i is a job. The sequence is defined in Algorithm 1 and illustrated in Figure 2. We use the notation $\mathrm{EDF}(j, S)$ for the total work that $\mathrm{EDF}[M, \sigma]$ allocates to a job j in a given subset S of $\mathbb{R}_+$.

The sequence defines time intervals $\Delta_i := [t_i, t_0)$ and $\overline{\Delta}_i := [t_i, t_{i-1})$. Each interval $\overline{\Delta}_i$ is partitioned into two subsets X_i and $Y_i := \overline{\Delta}_i \setminus X_i$. The subset X_i is the set of time instants in $\overline{\Delta}_i$ for which a job of task $\tau(j_i)$ is being processed by EDF. Due to the way EDF schedules, X_i is a finite union of intervals. Further, we set $x_i := \|X_i\|$ and $y_i := \|Y_i\|$.

Let ξ be the amount of work that $\mathrm{EDF}[M, \sigma]$ failed to complete before t_0 for jobs of task $\tau(j_1)$. By assumption, $\xi > 0$. The last definitions we need are $\overline{W}_i := M\sigma y_i + \sigma x_i$ and $W_i := \sum_{s=1}^{i} \overline{W}_s + \xi$.

Lemma 3.2. *Assume that Algorithm 1 produces the sequence of pairs $((t_1, j_1), (t_2, j_2), \ldots, (t_k, j_k))$. Then the following hold for all $i = 1, \ldots, k$.*

1. *$t_i < t_{i-1}$;*
2. *During interval Y_i all M machines are busy;*
3. *All jobs scheduled by $\mathrm{EDF}[M, \sigma]$ during Y_i have absolute deadline in Δ_i;*
4. *$W_i > m' \|\Delta_i\|$, where $m' := M(\sigma - 1) + 1$.*

Proof. The proof is by induction on i.

Basis of the induction. Job j_1 is defined as one of the jobs that $\mathrm{EDF}[M, \sigma]$ failed to complete within t_0, though they were due. Then $\overline{\Delta}_1$ $(= \Delta_1)$ is the maximal $\tau(j_1)$-busy interval before t_0. This also defines t_1 as the lower endpoint of this interval. Clearly, $t_1 < t_0$ since relative deadlines are strictly positive; thus property 1 holds.

Now, if at a certain time in $\overline{\Delta}_1$ no job of $\tau(j_1)$ is processed by EDF, then at that time all machines must be busy with jobs that have deadlines not later than t_0. This yields properties 2 and 3. For

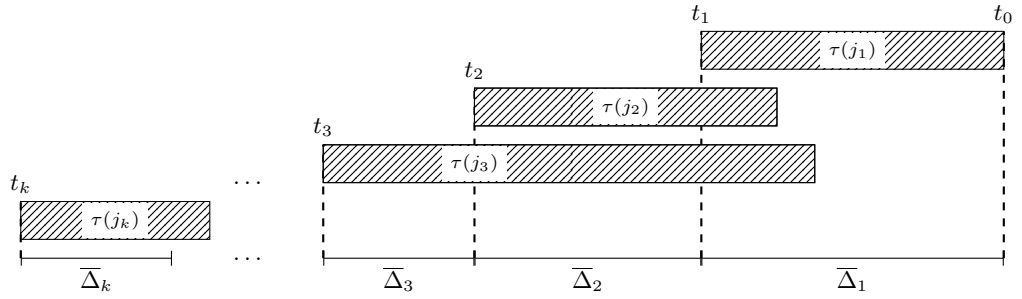

Figure 2: Illustration of the construction in Lemma 3.2. Shaded rectangles represent busy intervals. The interval's label is the task τ for which the interval is τ-busy.

property 4 we use the fact that EDF failed at t_0 for j_1:

$$\begin{aligned} W_1 &= \overline{W}_1 + \xi \\ &> M\sigma y_1 + \sigma x_1 = M\sigma\left(\left\|\overline{\Delta}_1\right\| - x_1\right) + \sigma x_1. \end{aligned}$$

We obtain

$$\frac{W_1}{\left\|\overline{\Delta}_1\right\|} > M\sigma - (M-1)\frac{\sigma x_1}{\left\|\overline{\Delta}_1\right\|}. \tag{1}$$

In $\overline{\Delta}_1$, EDF's schedule devotes x_1 time units to jobs of task $\tau(j_1)$ at processing speed σ. Since the interval $\overline{\Delta}_1$ is maximally $\tau(j_1)$-busy before t_0 and j_1 is not completed within t_0, all these jobs must be released in the interval, and all except j_1 have their deadline in the interval. The interval $\overline{\Delta}_1$ starts with the release date of some job of task $\tau(j_1)$ (possibly j_1 itself). Therefore the number of $\tau(j_1)$-jobs processed by EDF in $\overline{\Delta}_1$ is at most $\left\lfloor(\left\|\overline{\Delta}_1\right\| - D_{j_1} + T_{j_1})/T_{j_1}\right\rfloor$, and we can bound:

$$\frac{\sigma x_1}{\left\|\overline{\Delta}_1\right\|} \leq \frac{C_{j_1}}{\left\|\overline{\Delta}_1\right\|} \cdot \frac{\left\|\overline{\Delta}_1\right\| - D_{j_1} + T_{j_1}}{T_{j_1}} \leq \max\left(\frac{C_{j_1}}{D_{j_1}}, \frac{C_{j_1}}{T_{j_1}}\right) \leq 1. \tag{2}$$

The middle inequality can be verified by distinguishing the cases $D_{j_1} \leq T_{j_1}$ and $D_{j_1} > T_{j_1}$: in the case $D_{j_1} \leq T_{j_1}$, using $\left\|\overline{\Delta}_1\right\| \geq D_{j_1}$ one has

$$\begin{aligned} \frac{C_{j_1}}{\left\|\overline{\Delta}_1\right\|} \cdot \frac{\left\|\overline{\Delta}_1\right\| - D_{j_1} + T_{j_1}}{T_{j_1}} &= \frac{C_{j_1}}{T_{j_1}}\left(1 + \frac{T_{j_1} - D_{j_1}}{\left\|\overline{\Delta}_1\right\|}\right) \\ &\leq \frac{C_{j_1}}{T_{j_1}}\left(1 + \frac{T_{j_1} - D_{j_1}}{D_{j_1}}\right) = \frac{C_{j_1}}{D_{j_1}}. \end{aligned}$$

In the case $D_{j_1} > T_{j_1}$,

$$\frac{C_{j_1}}{\left\|\overline{\Delta}_1\right\|} \cdot \frac{\left\|\overline{\Delta}_1\right\| - D_{j_1} + T_{j_1}}{T_{j_1}} = \frac{C_{j_1}}{T_{j_1}}\left(1 - \frac{D_{j_1} - T_{j_1}}{\left\|\overline{\Delta}_1\right\|}\right) \leq \frac{C_{j_1}}{T_{j_1}}.$$

Combining (1) and (2) we get property 4:

$$\frac{W_1}{\left\|\overline{\Delta}_1\right\|} > M(\sigma - 1) + 1 = m'.$$

The inductive step. As the release date of j_i is strictly before t_{i-1}, we have $t_i < t_{i-1}$ so that property 1 holds. Properties 2 and 3 again follow from the fact that $\overline{\Delta}_i$ is $\tau(j_i)$-busy. Here, take into account for property 3 that j_i has a deadline in Δ_{i-1} by induction.

To prove property 4 it suffices to show $\overline{W}_i \geq m' \left\|\overline{\Delta}_i\right\|$, because by induction $W_{i-1} > m' \left\|\Delta_{i-1}\right\|$. By definition

$$\frac{\overline{W}_i}{\left\|\overline{\Delta}_i\right\|} = M\sigma - (M-1)\frac{\sigma x_i}{\left\|\overline{\Delta}_i\right\|}.$$

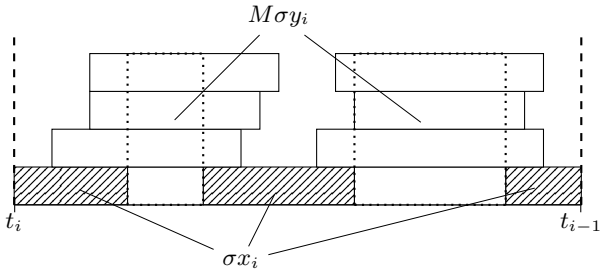

Figure 3: Illustration of the analysis in Theorem 3.1. The shaded area corresponds to the contribution σx_i, the dotted area to the contribution $M\sigma y_i$. When $\tau(j_i)$ is not executing, all M processors are busy.

We want to establish $\sigma x_i \leq \left\|\overline{\Delta}_i\right\|$. Having that, property 4 follows as it did in the base case of the induction. For this part we use a simplified notation by setting $\tau := \tau(j_i)$, $c := C_{\tau(j_i)}$, $T := T_{\tau(j_i)}$, and $D := D_{\tau(j_i)}$.

We will bound σx_i by the work performed by EDF on job j_i during $\overline{\Delta}_i$, plus the amount of work done for other jobs of task τ that are released during $\overline{\Delta}_i$ and before r_{j_i} (jobs of τ released later than j_i are not processed in $\overline{\Delta}_i$). Assume there are $q \geq 0$ such jobs. Then

$$\sigma x_i \leq q \cdot c + \mathrm{EDF}(j_i, \overline{\Delta}_i). \tag{3}$$

As the deadline of j_i is in Δ_{i-1}, by definition of forward forced demand and by choice of j_i,

$$\begin{aligned} \mathrm{EDF}(j_i, \overline{\Delta}_i) &\leq c - \mathrm{EDF}(j_i, \Delta_{i-1}) \\ &\leq c - \mathrm{EDF}\left(j_i, \bigcup_{s=1}^{i-1} Y_s\right) \\ &\leq c - \mathrm{FFD}(j_i, \Delta_{i-1}) \\ &\leq t_{i-1} - r_{j_i}. \end{aligned} \tag{4}$$

As q jobs of τ have been released in $[t_i, r_{j_i})$, we also have

$$t_{i-1} - r_{j_i} \leq \left\|\overline{\Delta}_i\right\| - q \cdot T. \tag{5}$$

Combining (3),(4) and (5) gives

$$\frac{\sigma x_i}{\left\|\overline{\Delta}_i\right\|} \leq 1 - \frac{q \cdot (T - c)}{\left\|\overline{\Delta}_i\right\|} \leq 1,$$

where the last inequality holds since $q \geq 0$ and $c \leq T$ by assumption. Property 4 now follows as in the base case of the induction. □

Proof of Theorem 3.1. At each step of Algorithm 1 the interval Δ_i is strictly extended backwards to the release date of at least one job which is released before t_0. As there are finitely many tasks, all with a positive minimum separation time, there are finitely many such jobs. So at some point the breaking condition, namely that there is no job j_i with the required properties, must hold.

Let k be the last index for which an appropriate job j_k was found. We claim that $\mathrm{FFD}(\Delta_k) \geq W_k$. In the value W_k we count σx_i for each X_i, because the whole τ-demand processed in a τ-busy interval is part of the forward forced demand of that interval. Also, the demand that EDF failed to process before t_0 is part of the forward forced demand of Δ_k. For each Y_i part we count $M\sigma y_i$, which is by Lemma 3.2(2) exactly what is processed in those times by EDF; see Figure 3 for an illustration. By Lemma 3.2(3) all jobs processed in some Y_i have their deadline in the interval Δ_i and therefore also in Δ_k. Finally, there is no job among those processed in some Y_i with release date before t_k, which has been counted in the term $M\sigma y_i$ with more than its forward forced demand in Δ_i. The forward forced demand in the greater interval Δ_k can only be greater, and thus we count for no job more in W_k than in $\mathrm{FFD}(\Delta_k)$. We conclude

$$\mathrm{FFD}(\Delta_k) \geq W_k.$$

On the other hand, by Lemma 3.2(4) applied to $i = k$, $W_k > m' \left\|\Delta_k\right\|$. The theorem follows. □

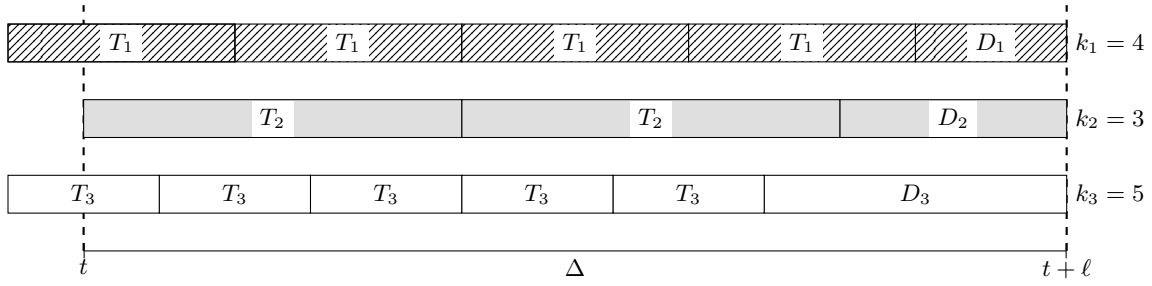

Figure 4: Example illustrating the construction in Lemma 4.1. Each interval is labeled with the corresponding length.

We required $C_\tau \leq \min(D_\tau, T_\tau)$ for all $\tau \in \mathbb{T}$. This condition can be trivially tested, and as we saw in Proposition 2.2 it is necessary for feasibility.

Now, assume $\sigma \geq 1 + \frac{m-1}{M}$. We get $m' = M(\sigma - 1) + 1 \geq m$. Then, if a task system $\mathbb{T}$ allows for a job sequence R with an interval Δ generating a forward forced demand $\mathrm{FFD}_R(\Delta) > m \|\Delta\|$ as in the theorem, by Proposition 2.1 it cannot be scheduled by any algorithm on m unit speed machines. So both the conditions of the theorem,

(1) $C_\tau \leq \min(D_\tau, T_\tau)$ for each task $\tau \in \mathbb{T}$, and

(2) $\mathrm{FFD}_R(\Delta) \leq m \|\Delta\|$ for each job sequence R and interval Δ

are necessary for feasibility on m unit speed machines. By Theorem 3.1 they are also sufficient for EDF-schedulability on M speed-σ machines. Therefore, all that is missing for an approximate feasibility test is an efficient procedure testing whether a task system $\mathbb{T}$ admits a job sequence R with an interval Δ generating a forward forced demand $\mathrm{FFD}_R(\Delta) > m \|\Delta\|$. For this we will provide an FPTAS in the following section. As this procedure determines the maximal load only up to an arbitrary positive error term, we will need to choose σ slightly larger than $1 + \frac{m-1}{M}$ to obtain the efficient test.

We remark that the minimum speed-up factor of $1 + \frac{m-1}{M}$ implicit in Theorem 3.1 cannot be improved, as it is known to be tight for EDF [14]. In fact a corollary of Theorem 3.1 is the well-known fact that EDF is an optimal scheduling algorithm for a single processor [16]. To see this, consider Theorem 3.1 when $\sigma = 1$ and $M = m = 1$. The theorem then states that if EDF fails to schedule a sequence, there is an interval Δ such that $\mathrm{FFD}(\Delta) > \|\Delta\|$. But then by Proposition 2.1 no algorithm can successfully schedule the same sequence of jobs on a unit speed processor.

4 A fully polynomial-time approximation scheme for load estimation

In this section we show how to efficiently compute an arbitrarily good estimate of the worst-case load of a sporadic task system. Together with Theorem 3.1, this will yield an efficient feasibility test. We remark that one cannot expect to design a polynomial-time algorithm for computing the worst-case load exactly: for one processor, by Theorem 3.1 such an algorithm would decide feasibility of a sporadic task system in polynomial time, thus solving a coNP-hard problem [12].

The following observation will facilitate the computation.

Lemma 4.1. *Assume $C_\tau \leq \min(D_\tau, T_\tau)$ for all tasks τ of a sporadic task system $\mathbb{T}$, and let $\ell \in \mathbb{N}$. Then*

$$\sup_{R,\, \Delta:\, \|\Delta\| = \ell} \mathrm{FFD}_R(\Delta) = \sum_{\tau \in \mathbb{T}} k_\tau \cdot C_\tau + (C_\tau + \ell - D_\tau - k_\tau \cdot T_\tau)^+, \text{ where } k_\tau := \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor.$$

Proof. To see that the supremum is at least as claimed, consider the interval $\Delta = [t, t + \ell)$ and the sequence R where, for each task τ, $t + \ell$ is the deadline of some τ-job, and jobs are separated according to the minimum separation time T_τ; more precisely, there is a τ-job released at time $t + \ell - D_\tau - i \cdot T_\tau$, for all $i \geq 0$ such that the absolute deadline of the job is in Δ (see Figure 4 for an example). There

are $k_\tau + 1$ jobs that can contribute to $\mathrm{FFD}_R(\tau, \Delta)$. Of these, the last k_τ have both release date and absolute deadline within Δ, and thus contribute $k \cdot C_\tau$ to the forward forced demand; the remaining job has release date before Δ and forward forced demand equal to $(C_\tau - (T_\tau - (\ell - D_\tau - (k-1)T_\tau)))^+$. After rearranging terms, the total forward forced demand is seen to be the same as in the claim.

To see that this is maximal, consider any job sequence R and any interval Δ with $\|\Delta\| = \ell$. As $C_\tau \leq T_\tau$ for all τ, at most $k+1$ jobs can contribute to $\mathrm{FFD}_R(\tau, \Delta)$, and all such jobs have deadline in Δ. Now we modify the sequence in the following way: we first eliminate jobs that are not contributing to the forward forced demand; then, for each contributing job, starting with the one released last, we delay its release date as much as possible so that 1) the deadline of the job remains inside the interval; 2) the release date of the job does not violate the minimum separation time constraint. Notice that this modification does not decrease the total forward forced demand. On the other hand, after the modification we obtain a pair (R, Δ) that can be analyzed exactly as in the first part of the claim. □

In view of Lemma 4.1, we define for a given instance $\mathbb{T}$ the functions

$$w_\tau(\ell) := k_\tau \cdot C_\tau + (C_\tau + \ell - D_\tau - k_\tau \cdot T_\tau)^+$$
$$w(\ell) := \sum_{\tau \in \mathbb{T}} w_\tau(\ell).$$

Lemma 4.1 states that $w(\ell)$ is the maximum forward forced demand of any job sequence of $\mathbb{T}$ in any interval of length ℓ. The construction of the lemma also showed that the maximal forward forced demand is achieved for each task independently. As a consequence it only remains to find the optimal length of the interval: the value of ℓ that maximizes the load $w(\ell)/\ell$.

We use Algorithm 2 to approximate the maximum of $w(\ell)/\ell$ within a factor of ϵ in time polynomial in the input size of $\mathbb{T}$ and $1/\epsilon$. In fact, we devise a polynomial-time computable function ϕ which pointwise approximates the load. We also show that there is a polynomial size set of integers, a priori determinable, in which the function ϕ must achieve its maximum. The approximation algorithm then simply consists in evaluating ϕ for every point from this subset.

Algorithm 2 Load Estimation($\mathbb{T}, \epsilon$)

For each $\tau \in \mathbb{T}$, compute:

$$\begin{aligned}\text{threshold}(\tau) &:= D_\tau + T_\tau/\epsilon,\\ S'(\tau) &:= \{\ell \in (0, \text{threshold}(\tau)] : \ell = q \cdot T_\tau + D_\tau \text{ for some } q \in \mathbb{N}\},\\ S''(\tau) &:= \{\ell \in (0, \text{threshold}(\tau)] : \ell = q \cdot T_\tau + D_\tau - C_\tau \text{ for some } q \in \mathbb{N}\}.\end{aligned}$$

Let $S := \cup_{\tau \in \mathbb{T}} (S'(\tau) \cup S''(\tau) \cup \{\text{threshold}(\tau)\}) \cup \{1, \infty\}$.
Output

$$\lambda := \max_{\ell \in S} \left(\sum_{\tau : \ell \leq \text{threshold}(\tau)} \frac{w_\tau(\ell)}{\ell} + \sum_{\tau : \ell > \text{threshold}(\tau)} \left(1 - \frac{D_\tau}{\ell}\right) \frac{C_\tau}{T_\tau} \right).$$

Lemma 4.2. *For any task system $\mathbb{T}$ and $\epsilon \in (0,1)$ Algorithm 2 outputs $\lambda \in \mathbb{Q}$ such that $(1-\epsilon)\lambda^* \leq \lambda \leq \lambda^*$, where $\lambda^* := \sup_{R,\Delta} \frac{\mathrm{FFD}_R(\Delta)}{\|\Delta\|}$. The running time of the algorithm is polynomial in the size of $\mathbb{T}$ and $1/\epsilon$.*

Proof. We will show that for all $\ell \in \mathbb{N}$ the function

$$\phi(\ell) := \sum_{\tau : \ell \leq \text{threshold}(\tau)} \frac{w_\tau(\ell)}{\ell} + \sum_{\tau : \ell > \text{threshold}(\tau)} \left(1 - \frac{D_\tau}{\ell}\right) \frac{C_\tau}{T_\tau}$$

approximates the load $w(\ell)/\ell$ in the sense that for all $\ell \geq 1$,

$$(1-\epsilon)\frac{w(\ell)}{\ell} \leq \phi(\ell) \leq \frac{w(\ell)}{\ell}. \tag{6}$$

We will also show that we can find the maximum of ϕ by only considering points in S. This will complete the proof, since the number of points in S is polynomial in the input, and similarly ϕ can be evaluated in polynomial time for any given point.

Recall that by definition,

$$w_\tau(\ell) = \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor C_\tau + \left(C_\tau + \ell - D_\tau - \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor \cdot T_\tau \right)^+ . \tag{7}$$

As the second term is nonnegative, after multiplying both sides by T_τ we obtain

$$w_\tau(\ell) \cdot T_\tau \geq C_\tau \cdot (\ell - D_\tau) \tag{8}$$

which implies

$$\frac{w_\tau(\ell)}{\ell} \geq \frac{C_\tau}{T_\tau} \cdot \left(1 - \frac{D_\tau}{\ell}\right).$$

If we sum this inequality over all tasks τ we obtain the upper bound on ϕ in (6).

To obtain the lower bound in (6) it suffices to consider tasks τ for which threshold$(\tau) < \ell$ (for each other task, the term $w_\tau(\ell)/\ell$ in $\phi(\ell)$ exactly accounts for the contribution of the task). This condition is equivalent to $D_\tau + T_\tau/\epsilon < \ell$, implying $T_\tau < (\ell - D_\tau) \cdot \epsilon$. Using again (8) gives $C_\tau < w_\tau(\ell) \cdot \epsilon$.

If we start again from (7) and use the shorthand

$$z_\tau := \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor \cdot T_\tau - (\ell - D_\tau),$$

after observing that $z_\tau \in [0, T_\tau]$ we can bound

$$w_\tau(\ell) - (\ell - D)\frac{C_\tau}{T_\tau} = z_\tau \cdot \frac{C_\tau}{T_\tau} + (c - z_\tau)^+ \leq C_\tau.$$

We thus obtain

$$\frac{w_\tau(\ell) - (\ell - D_\tau) \cdot C_\tau/T_\tau}{w_\tau(\ell)} \leq \frac{C_\tau}{w_\tau(\ell)} < \epsilon,$$

which can be equivalently expressed as

$$\frac{\ell - D_\tau}{\ell} \cdot \frac{C_\tau}{T_\tau} > (1 - \epsilon)\frac{w_\tau(\ell)}{\ell} .$$

Summing this last bound over all tasks completes the proof of (6).

To conclude the proof we observe that S has been defined so that between any two consecutive points $\ell_1, \ell_2 \in S$ the function $\ell \cdot \phi(\ell)$ is linear (compare with the definitions of $\phi(\ell)$ and $w(\ell)$). This implies that within any such interval $[\ell_1, \ell_2]$ the maximum of ϕ is achieved at an extreme point of the interval. Therefore, the global maximum of ϕ is attained at some point in S, and the lemma follows. □

Theorem 4.3. *Let $m, M \in \mathbb{N}$, $\epsilon \in (0,1)$, and $\sigma \geq 1 + M^{-1}(\frac{m}{1-\epsilon} - 1)$. There exists a feasibility test that, given a task system $\mathbb{T}$, ϵ and m, decides whether $\mathbb{T}$ is* EDF*-schedulable on M speed-σ machines, or $\mathbb{T}$ is not feasible on m unit speed machines. The running time of the test is polynomial in the size of $\mathbb{T}$, $1/\epsilon$ and $\log m$.*

Proof. By Lemma 4.2 we can verify within the claimed time bound the following conditions:

(C1) For all tasks $\tau \in \mathbb{T}$, $C_\tau \leq \min(D_\tau, T_\tau)$.

(C2) $\lambda \leq m$, where $(1-\epsilon)\lambda^* \leq \lambda \leq \lambda^*$ and $\lambda^* := \sup_{R,\Delta} \frac{\mathrm{FFD}_R(\Delta)}{\|\Delta\|}$.

Both conditions are necessary for scheduling $\mathbb{T}$ on m unit speed machines; this follows from Proposition 2.2 for (C1), and from Proposition 2.1 for (C2).

On the other hand, (C2) implies, by definition of λ^*, that there is no job sequence R and interval Δ such $\mathrm{FFD}_R(\Delta) > \frac{m}{1-\epsilon}\|\Delta\|$. By the assumption on σ, $M(\sigma - 1) + 1 \geq \frac{m}{(1-\epsilon)}$, and the claim follows from Theorem 3.1. □

Corollary 4.4. *Let $m \in \mathbb{N}$ and $\epsilon \in (0,1)$. There exists a feasibility test that, given a task system $\mathbb{T}$, ϵ and m, decides whether $\mathbb{T}$ is* EDF*-schedulable on m speed-$(2 - 1/m + \epsilon)$ machines, or $\mathbb{T}$ is not feasible on m unit speed machines. The running time of the test is polynomial in the size of $\mathbb{T}$, $1/\epsilon$ and $\log m$.*

Acknowledgments. The authors thank Enrico Bini and Sanjoy Baruah for helpful discussions.

References

[1] K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proc. 16th Euromicro Conf. on Real-Time Systems*, pages 187–195, 2004.

[2] K. Albers and F. Slomka. Efficient feasibility analysis for real-time systems with EDF scheduling. In *Proc. Conf. on Design, Automation and Test in Europe*, pages 492–497, 2005.

[3] T. P. Baker. An analysis of EDF schedulability on a multiprocessor. *IEEE Transactions on Parallel and Distributed Systems*, 16(8):760–768, 2005.

[4] T. P. Baker and S. K. Baruah. Schedulability analysis of multiprocessor sporadic task systems. In S. H. Son, I. Lee, and J. Y.-T. Leung, editors, *Handbook of Real-Time and Embedded Systems*, chapter 3. CRC Press, 2007.

[5] S. K. Baruah and T. P. Baker. Global EDF schedulability analysis of arbitrary sporadic task systems. In *Proc. 20th Euromicro Conf. on Real-Time Systems*, pages 3–12, 2008.

[6] S. K. Baruah and T. P. Baker. Schedulability analysis of global EDF. *Real-Time Systems*, 38(3):223–235, 2008.

[7] S. K. Baruah, R. R. Howell, and L. E. Rosier. Feasibility problems for recurring tasks on one processor. *Theoretical Computer Science*, 118(1):3–20, 1993.

[8] V. Bonifaci, H.-L. Chan, A. Marchetti-Spaccamela, and N. Megow. Algorithms and complexity for periodic real-time scheduling. In M. Charikar, editor, *Proc. 21st Symp. on Discrete Algorithms*, 2010.

[9] S. Chakraborty, S. Künzli, and L. Thiele. Approximate schedulability analysis. In *Proc. 23rd Real-Time Systems Symp.*, pages 159–168, 2002.

[10] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *Proc. IFIP Congress*, pages 807–813, 1974.

[11] F. Eisenbrand and T. Rothvoß. A PTAS for static priority real-time scheduling with resource augmentation. In *Proc. 35th Int. Colloquium on Automata, Languages and Programming*, pages 246–257, 2008.

[12] F. Eisenbrand and T. Rothvoß. EDF-schedulability of synchronous periodic task systems is coNP-hard. In *Proc. 21st Symp. on Discrete Algorithms*, pages 1029–1034, 2010.

[13] N. Fisher and S. K. Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines. In *Proc. 17th Euromicro Conf. on Real-Time Systems*, pages 117–126, 2005.

[14] T. W. Lam and K.-K. To. Trade-offs between speed and processor in hard-deadline scheduling. In *Proc. 10th Symp. on Discrete Algorithms*, pages 623–632, 1999.

[15] J. Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 11(3):115–118, 1980.

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

[17] C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.