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

Vincenzo Bonifaci[1,2], Alberto Marchetti-Spaccamela[1], and Sebastian Stiller[3]

[1] Sapienza Università di Roma, Italy
[2] Università degli Studi dell'Aquila, Italy
[3] Technische Universität Berlin, Germany

Abstract. We devise the first constant-approximate feasibility test for sporadic multiprocessor real-time scheduling. We give an algorithm that, given a task system and $\varepsilon > 0$, correctly decides either that the task system can be scheduled using the earliest deadline first algorithm on m speed-$(2 - 1/m + \varepsilon)$ machines, or that the system is infeasible for m speed-1 machines. The running time of the algorithm is polynomial in the size of the task system and $1/\varepsilon$. We also provide an improved bound trading off speed for additional machines.
Our analysis relies on a new concept for counting the workload of an interval, that might also turn useful for analyzing other types of task systems.

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 I of tasks, which need to be executed by the system; each task generates a possibly infinite sequence of jobs.

In the *periodic* version of the problem, a task τ, $\tau \in I$, is characterized by a quadruple of positive numbers: an offset o_τ that represents the time instant when the first job generated by the task is released, a processing time c_τ, a relative deadline D_τ and a period T_τ. Each invocation of task τ is represented by a job: the kth occurrence of task τ is released at time $o_\tau + (k-1)T_\tau$, requires at most c_τ units of processor time and must complete its execution before time $o_\tau + (k-1)T_\tau + D_\tau$. Note that a task defines an infinite sequence of jobs, but a given set of tasks generates exactly one sequence of jobs. In the following we denote by n the cardinality of I.

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* time interval between successive invocations of the task. Note that in a sporadic task system there is no a priori knowledge of when the next invocation of a task will be released after the minimal separation time has elapsed. Therefore, in analyzing sporadic task systems, every conceivable sequence of jobs consistent with the task system specification must be considered.

In hard-real-time systems it is imperative for the correctness of the system that all jobs complete by their deadlines. A periodic (sporadic) task system is *feasible* if there is a feasible schedule for every set of jobs that is consistent with the period, deadline,

and worst-case execution time constraints of the task system, and it is *schedulable* by a given algorithm if the algorithm finds a feasible schedule for every such set of jobs. In the sequel we focus on preemptive scheduling algorithms that are allowed to interrupt the execution of a job and resuming it later.

Given a scheduling algorithm A, a *schedulability test* for A is an algorithm that takes as input a description of a task system and outputs an answer to whether the system is schedulable by A or not. A schedulability test is *exact* if it correctly identifies all schedulable and unschedulable task systems and it is *sufficient* if it correctly identifies all unschedulable task systems, but may give a wrong answer for schedulable task systems. For any scheduling algorithm to be useful for hard-deadline real-time applications it must have at least a sufficient schedulability test, that can verify that a given job set is schedulable. The quality of the scheduling algorithm and the schedulability test are inseparable, since there is no practical difference between a job set that is not schedulable and one that cannot be proven to be schedulable.

In the case of a single machine, the problem has been widely studied and effective scheduling algorithms are well understood [5, 11]. In this paper we study scheduling algorithm for sporadic task systems on parallel machines. The problem is not only interesting from a theoretical point of view but it is also relevant in practice. In fact, real-time multiprocessor systems are becoming common: there are single-chip architectures, characterized by a small number of processors and large-scale signal-processing systems with many processing units.

Related work

There is an extensive literature on real-time scheduling. We limit the following discussion to the results that are more relevant to our work.

Single machine scheduling. In the case of a single machine it is known [5, 7] that the earliest deadline first scheduling algorithm (EDF), which at each instant in time schedules the available job with the smallest deadline (with ties broken arbitrarily), is an optimal scheduling algorithm for scheduling a periodic (sporadic) task system in the following sense: if it is possible to preemptively schedule a given collection of independent 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 the previous positive result, the feasibility test for periodic task systems is strongly co-NP-hard even in special cases and it is solvable in exponential time [5, 10].

Approximate feasibility tests have been proposed that allow the design of efficient feasibility tests (e.g. running in polynomial time) while introducing a small error in the decision process, that is 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 set is guaranteed to be feasible. If the test returns "infeasible", the task set is guaranteed to be infeasible on a slower processor, of computing capacity $(1-\varepsilon)$, where ε denotes the approximation guaranteed.

If an optimistic test returns "feasible", then the task set is guaranteed to be feasible on a $(1+\varepsilon)$-speed processor. If the test returns "infeasible", the task set is guaranteed to be infeasible on a unit-speed processor [6].

Fully polynomial-time approximation schemes (FPTAS) are known for a single processor; in fact for any $\varepsilon > 0$ there exists a feasibility test that returns an ε-approximation; the running time of the algorithm is polynomial in the number of tasks and in $1/\varepsilon$ (see for example [1, 2, 6, 8] and references therein).

Finally we observe that, in the case of one processor, the sporadic feasibility problem is known to reduce to a special case of the periodic problem, where all tasks have offset 0 (i.e. each task releases its first job at time zero).

Multiple machine scheduling. We first observe that in the multiprocessor case the previous analogy between sporadic and periodic problems is not true.

Regarding the analysis of EDF, it has been proven [12] that any *feasible* task system on m machines of unit capacity is EDF-schedulable on m machines of speed $2 - 1/m$. This result holds for EDF and other scheduling policies and has not been improved since then. Subsequent work has analyzed the advantage of trading speed for machines [9], while further work on conditions for the schedulability of EDF has been done by Baker [3].

Note that the result of [12] *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. [12] is the lack of a feasibility test.

The problem has attracted a lot of attention in recent years (see e.g. [4] and references therein). 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 has been shown that

$$\sum_{\tau \in I} \frac{c_\tau}{T_\tau} \leq m \text{ and } \max_{\tau \in I} \frac{c_\tau}{T_\tau} \leq 1$$

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

However, not much was known regarding the feasibility of an arbitrary-deadline task system. A sufficient test in this case is given by

$$\sum_{\tau \in I} \frac{c_\tau}{\min(D_\tau, T_\tau)} \leq m \text{ and } \max_{\tau \in I} \frac{c_\tau}{\min(D_\tau, T_\tau)} \leq 1,$$

but this test is far from approximating a necessary condition, i.e., it does not provide a good approximate feasibility test in general (it is not hard to see that there exist feasible task systems for which $\sum_{\tau \in I} c_\tau / \min(D_\tau, T_\tau)$ can be $\Omega(m \log m)$).

To the best of our knowledge, no better bound is known. We refer the reader to the survey [4] for feasibility tests that are known for other special cases.

Our Contribution

We give the first constant-approximate feasibility test for sporadic multiprocessor real-time scheduling. Namely, we show for any sporadic multiprocessor instance I that it can either be scheduled by EDF on m speed-$(2 - 1/m + \varepsilon)$ machines, or that the instance violates some basic conditions, which are necessary for being scheduled by any algorithm on m speed-1 machines. In fact we give a slightly stronger result, allowing to trade some extra speed for extra machines. Note, that in general extra machines are less powerful than extra speed. Two of the basic conditions can be checked trivially. For the

third we give an algorithm checking the condition in time polynomial in the input size of I and $1/\varepsilon$, for any desired $\varepsilon > 0$.

In the proof we devise a completely new concept of counting demand, the so-called *forward forced demand*. This concept is strong enough to approximately capture the possibilities of scheduling on multiprocessors. But it is still simple enough to be approximated in polynomial time upto an arbitrarily small $\varepsilon > 0$.

2 The model

An instance is a finite set of tasks I. Each task $\tau \in I$ is a triple of positive numbers, namely, a processing time c_τ, a relative deadline D_τ and a period or minimal separation time T_τ. Every job j belongs to a task τ_j, and has a release date $r_j \geq 0$. We write $c_j := c_{j_\tau}$, and $D_j := D_{j_\tau}$, and $T_j := T_{j_\tau}$, and we call $r_j + D_j = d_j$ the (absolute) deadline of j. We assume D_τ, c_τ and $T_\tau \in \mathbb{N}$.

A (sporadic) realization R of an instance I is an arbitrary, countable set of jobs, all belonging to tasks in I, with the following property: Any pair of distinct jobs j and k belonging to the same task τ, satisfies $|r_j - r_k| \geq T_\tau$.

A feasible schedule for a realization R on m machines is a set of measurable functions $S_j : \mathbb{R}^+ \to \{0, \ldots, m\}$, one function for each job $j \in R$, satisfying:

- Everything is scheduled: $\forall j \in R : c_j = \sum_{p=1}^{m} |S_j^{-1}(p)|$.
- Deadlines and release dates are respected: $\forall j \in R : \bigcup_{p=1}^{m} S_j^{-1}(p) \subseteq [r_j, d_j]$.
- Each machine processes at most one job at a time: $\forall p \in \{1, \ldots, m\} : \forall j \neq g \in R : S_j^{-1}(p) \cap S_g^{-1}(p) = \emptyset$.
- Jobs of the same task are not scheduled in parallel:

$$\forall j \neq g \in R : \tau_j = \tau_g \Rightarrow \bigcup_{p=1}^{m} S_j^{-1}(p) \cap \bigcup_{p=1}^{m} S_g^{-1}(p) = \emptyset.$$

- No is processed job at two machines at the same time:

$$\forall j \in R, \forall p \neq q \in \{1, \ldots, m\} : S_j^{-1}(p) \cap S_j^{-1}(q) = \emptyset.$$

Preemption and migration of jobs are explicitly allowed.

3 A feasibility test

Definition 1. *Consider a job j with release date r_j, absolute deadline d_j, and processing time c_j satisfying $d_j \geq r_j + c_j$ (i.e., for its task we have $D_{\tau_j} \geq c_{\tau_j}$). For a non-empty interval $\Delta = [t, t + |\Delta|)$ with $d_j \in \Delta$, we call*

$$f(j, \Delta) := [c_j - (t - r_j)^+]^+$$

the forward forced demand *of j in Δ.*

Note, for a job j and an interval Δ, where both deadline and release date lie in the interval, $r_j, d_j \in \Delta$, the forward forced demand equals the processing time of the job, $f(j,\Delta) = c_j$. If $c_\tau \leq T_\tau$ for all tasks τ, then each pair of an interval Δ and a task τ can have at most one job j_τ with release date outside the interval $r_{j_\tau} \notin \Delta$ that has positive forward forced demand $f(j_\tau,\Delta) > 0$ in the interval.

Definition 2. *For a realization R of an instance I the* necessary demand $W_R(\Delta)$ *of a non-empty interval Δ is the sum of the forward forced demands of all jobs with absolute deadline in Δ. We write $W(\Delta)$ when the realization R is unmistakable. We use $W_\tau(\Delta)$ to denote the part of the necessary demand stemming only from jobs of task τ.*

Observe, that any algorithm working on any number of speed-1 machines must schedule in an interval at least the necessary demand of that interval.

We use the notation $\text{EDF}_{(m+\mu,\sigma)}$ to express the scheduling according to EDF on $(m+\mu)$ speed-σ machines, breaking ties arbitrarily.

Definition 3. *Given an instance I and a realization R. For a point in time t, a task τ, and a scheduling algorithm A an interval $\Delta = [t',t)$ is called* τ-A-busy before t, *if executing the algorithm A on the realization R yields for every point in Δ a positive remaining processing time for at least one of the jobs of task τ.*

Observe that the maximal τ-A-busy interval before t is unique, well defined, and starts with the release date of some job of τ, unless it is empty. Moreover, all demand from τ-jobs released before some maximal τ-A-busy interval Δ is processed by A strictly before Δ.

Theorem 1. *Let $\sigma \geq 2 - \frac{1+\mu}{m+\mu}$. Given an instance I which satisfies for all tasks τ that $c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$. If there is some realization R which cannot be scheduled by $EDF_{(m+\mu,\sigma)}$, then there is an interval Δ, in which R generates a necessary demand $W_R(\Delta) > m|\Delta|$.*

Before we give the formal and a bit involved proof we convey the main intuitions. Knowing that $\text{EDF}_{(m+\mu,\sigma)}$ fails, we will inductively construct an interval with load greater than m. The interval is comprised of several subintervals. To each subinterval we associate a task such that the subinterval is busy for that task. Whenever EDF does not process a job of that task in the subinterval, it must have all machines busy. In order to conclude that the load of the whole interval is big, we must establish two things: First, the fraction of a subinterval, in which its associated task is processed, is small, i.e., the part, when all machines must be busy, is large. Second, everything processed in those busy parts is part of the necessary demand of the whole interval. To establish these tokens, we need to use that the forced forward demand is necessary in an interval.

Proof. From now on we assume that R is a realization which cannot be scheduled by $\text{EDF}_{(m+\mu,\sigma)}$, and that t_0 is the first point in time, when $\text{EDF}_{(m+\mu,\sigma)}$ fails a deadline.

We define inductively a finite sequence of pairs, comprised of a time t_i and a job j_i, for $1 \leq i \leq z$. For convenience define $\Delta_i := [t_i,t_0)$ and $\overline{\Delta_i} := [t_i,t_{i-1})$. Also the following

notation for the work that $\mathrm{EDF}_{(m+\mu,\sigma)}$ does for a job j in a certain measurable subset S of $\mathbb{R}^+$ will be helpful: $\mathrm{EDF}_{(m+\mu,\sigma)}(j,S)$.

For each pair (t_i, j_i) we define two subsets of the interval $\overline{\Delta_i}$, namely X_i and Y_i. The first subset X_i is the set of points in time between t_i and t_{i-1} when a job of task τ_{j_i} is processed. Due to the way $\mathrm{EDF}_{(m+\mu,\sigma)}$ schedules, X_i is a finite union of intervals. The other subset is its complement in the interval: $Y_i := \overline{\Delta_i} \setminus X_i$. Further, we set $x_i := |X_i|$ and $y_i := |Y_i|$.

Next, we define three values for each pair. They will be interpreted later as certain parts of the work that $\mathrm{EDF}_{(m+\mu,\sigma)}$ does or has to do.

$$\widetilde{W}^{\overline{\Delta_i}} := (m+\mu)\sigma y_i + \sigma x_i$$
$$\widetilde{W}^{\Delta_i} := \sum_{s=1}^{i} \widetilde{W}^{\overline{\Delta_s}}$$
$$W^{\Delta_i} := \widetilde{W}^{\Delta_i} + \{\text{The work } \mathrm{EDF}_{(m+\mu,\sigma)} \text{ failed to do for jobs of task } \tau_{j_1} \text{ until } t_0.\}$$

We will show the following properties for our sequence:

1. $t_0 > t_1 > \ldots > t_z$.
2. During each Y_i all machines are busy.
3. All jobs $\mathrm{EDF}_{(m+\mu,\sigma)}$ schedules during Y_i have a deadline in Δ_i.
4. $W^{\Delta_i} > m|\Delta_i|$.
5. $W(\Delta_z) \geq W^{\Delta_z}$.

Property 2 implies that $(m+\mu)\sigma y_i = \sum_{j\in J} \mathrm{EDF}_{(m+\mu,\sigma)}(j, Y_i)$ for some set of jobs J.

Start of the induction. As job j_1 we pick one of the jobs $\mathrm{EDF}_{(m+\mu,\sigma)}$ failed to finish at t_0, though they were due. Among these jobs the job j_1 is one of those jobs j with largest maximal τ_j-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval before t_0. We choose the maximal τ_{j_1}-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval before t_0 as $\overline{\Delta_1} = \Delta_1$, which indirectly defines t_1.

We have to verify property 2, 3 and 4 for (t_1, j_1). If at a certain time t in the τ_{j_1}-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval $\overline{\Delta_1}$ no job of τ_{j_1} is processed by $\mathrm{EDF}_{(m+\mu,\sigma)}$, then all machines must be busy with jobs that have deadlines not later than t_0. This gives the first two properties. For property 4 we use that $\mathrm{EDF}_{(m+\mu,\sigma)}$ failed at t_0 for j_1:

$$\begin{aligned} W^{\Delta_1} &= \widetilde{W}^{\Delta_1} + \{\text{The work } \mathrm{EDF}_{(m+\mu,\sigma)} \text{ failed to do for jobs of task } \tau_{j_1} \text{ until } t_0\} \\ &= \widetilde{W}^{\overline{\Delta_1}} + \{\text{The work } \mathrm{EDF}_{(m+\mu,\sigma)} \text{ failed to do for jobs of task } \tau_{j_1} \text{ until } t_0\} \\ &> (m+\mu)\sigma y_1 + \sigma x_1 = (m+\mu)\sigma\left(|\Delta_1| - x_1\right) + \sigma x_1 \end{aligned}$$

So we get

$$\frac{W^{\Delta_1}}{|\Delta_1|} > (m+\mu)\sigma - (m+\mu-1)\frac{\sigma x_1}{|\Delta_1|}.$$

In Δ_1 the $\text{EDF}_{(m+\mu,\sigma)}$ schedule devotes x_1 units of time on jobs of task τ_{j_i} processing with speed-σ. Because, the interval Δ_1 is maximally τ_{j_1}-$\text{EDF}_{(m+\mu,\sigma)}$-busy before t_0 and j_i is not finished until t_0, we know that all those jobs must be released in the interval, and have their deadline in the interval.

The busy interval Δ_1 starts with the release date of some job of task τ_{j_1}. Therefore the number of jobs of that task with release date and deadline in Δ_1 is $\left(\left\lfloor \frac{|\Delta_1|-D_{j_1}+T_{j_1}}{T_{j_1}} \right\rfloor\right)$ on the number of jobs of τ_{j_1}. Therefore we can bound:

$$\frac{\sigma x_1}{|\Delta_1|} < \frac{c_{j_1}}{|\Delta_1|} \cdot \frac{|\Delta_1|-D_{j_1}+T_{j_1}}{T_{j_1}} \leq \max\left(\frac{c_{j_1}}{D_{j_1}}, \frac{c_{j_1}}{T_{j_1}}\right) \leq 1.$$

To verify the middle inequality one should distinguish the cases $(D_{j_1} \leq T_{j_1})$ and $(D_{j_1} > T_{j_1})$ using $|\Delta_1| \geq D_{j_1}$ for the former.

Combining the two bounds we get property 4 from the choice of σ:

$$\frac{W^{\Delta_1}}{|\Delta_1|} > (m+\mu)(\sigma-1)+1 \geq m.$$

The step from $i-1$ to i. Given the sequence of pairs satisfying the properties until $(i-1)$. We choose the job j_i as one having the following two properties:

1. The release date of j_i is strictly before t_{i-1}.
2. $$\text{EDF}_{(m+\mu,\sigma)}\left(j_i, \bigcup_{s=1}^{i-1} Y_s\right) > f(j_i, \Delta_{i-1}).$$

If no such job can be found, set $z = i-1$ and return the interval Δ_{i-1} as one justifying the claim of the theorem. We will show later, why this holds true. So, assume j_i exists as required. Take $\overline{\Delta_i}$ as the maximal τ_{j_i}-$\text{EDF}_{(m+\mu,\sigma)}$-busy interval before t_{i-1}, and accordingly set t_i as its beginning.

Let us show the properties. As the release date of j_i is strictly before t_{i-1}, also $t_i < t_{i-1}$, and we have property 1. The next two properties again follow from the fact that $\overline{\Delta_i}$ is τ_{j_i}-$\text{EDF}_{(m+\mu,\sigma)}$-busy. Here, take into account for property 3 that j_i has a deadline in Δ_{i-1} by induction.

To proof property 4 it suffices to show $W^{\overline{\Delta_i}} \geq m\left|\overline{\Delta_i}\right|$, because we have strict inequality in $W^{\Delta_{i-1}} > m\left|\Delta_{i-1}\right|$ from induction. By definition

$$\frac{W^{\overline{\Delta_i}}}{\left|\overline{\Delta_i}\right|} = (m+\mu)\sigma - (m+\mu-1)\frac{\sigma x_i}{\left|\overline{\Delta_i}\right|}.$$

We want to establish $\sigma x_i \leq \left|\overline{\Delta_i}\right|$. Having this, property 4 follows as above.

For this part we simplify notation $\tau := \tau_{j_i}$, $T := T_{\tau_{j_i}}$, $c := c_{\tau_{j_i}}$ and $D := D_{\tau_{j_i}}$. Distinguish the cases $\left|\overline{\Delta_i}\right| < T$ and $\left|\overline{\Delta_i}\right| \geq T$:

Case $\left(\left|\overline{\Delta_i}\right| \geq T\right)$**.** We can bound σx_i by the amount of work released for τ during the maximal τ-EDF$_{(m+\mu,\sigma)}$-busy interval $\overline{\Delta_i}$.

$$\sigma x_i \leq \left\lfloor \frac{\left|\overline{\Delta_i}\right|}{T} \right\rfloor \cdot c + \mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \overline{\Delta_i}\right).$$

W.l.o.g. $\left|\overline{\Delta_i}\right|$ is not an integer multiple of T. Otherwise, the last released job could not contribute to the work done in X_i. But, then a slightly smaller value replacing $\left|\overline{\Delta_i}\right|$ would also give a valid bound on what is processed during $\overline{\Delta_i}$.

Recall that $f(j_i, \Delta_{i-1}) := \left[c_{j_i} - \left[t_{i-1} - r_{j_i}\right]^+\right]^+$. By choice of j_i we know that more than its forced forward demand is done by EDF$_{(m+\mu,\sigma)}$ in Δ_{i-1}. Therefore

$$\mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \overline{\Delta_i}\right) \leq c - f(j_i, \Delta_{i-1}) \leq \left(t_{i-1} - r_{j_i}\right) \leq \left(\left|\overline{\Delta_i}\right| - T \cdot \left\lfloor \frac{\left|\overline{\Delta_i}\right|}{T} \right\rfloor\right).$$

Note, that the middle inequality is also true for $f(j_i, \Delta_{i-1}) = 0$. To verify the last inequality, assume first that j_i is the last job of task τ released in $\overline{\Delta_i}$. Then between the release of j_i and the end of $\overline{\Delta_i}$ at most $\left(\left|\overline{\Delta_i}\right| - T \cdot \left\lfloor \frac{\left|\overline{\Delta_i}\right|}{T} \right\rfloor\right)$ units of time may pass.

Now, say j_i is not the last job of task τ released in $\overline{\Delta_i}$. Remember that j_i is not finished by EDF$_{(m+\mu,\sigma)}$ within $\overline{\Delta_i}$. Therefore all jobs of τ released later are not processed within $\overline{\Delta_i}$ at all, because EDF implies FIFO for the jobs of a common task. If there is such a job released but not started in $\overline{\Delta_i}$, we can subtract its entire processing time from the upper bound on σx_i. This means to subtract at least as much as when we subtract $\mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \overline{\Delta_i}\right)$. Thus, we have

$$\sigma x_i \leq \left\lfloor \frac{\left|\overline{\Delta_i}\right|}{T} \right\rfloor \cdot c \leq \frac{\left|\overline{\Delta_i}\right|}{T} \cdot c \leq \left|\overline{\Delta_i}\right|.$$

To finish the case $\left(\left|\overline{\Delta_i}\right| \geq T\right)$ plug everything together:

$$\frac{\sigma x_i}{\left|\overline{\Delta_i}\right|} \leq \frac{\left\lfloor \frac{\left|\overline{\Delta_i}\right|}{T} \right\rfloor \cdot c + \left|\overline{\Delta_i}\right| - \left\lfloor \frac{\left|\overline{\Delta_i}\right|}{T} \right\rfloor \cdot T}{\left|\overline{\Delta_i}\right|} = 1 - \frac{(T-c)\left\lfloor \frac{\left|\overline{\Delta_i}\right|}{T} \right\rfloor}{\left|\overline{\Delta_i}\right|}.$$

As $T \geq c$ we have $\sigma x_i \leq \left|\overline{\Delta_i}\right|$.

Case $\left(\left|\overline{\Delta_i}\right| < T\right)$**.** Assume $\left|\overline{\Delta_i}\right| < T$. Then only one job of task τ can be released during $\left|\overline{\Delta_i}\right|$, namely j_i. The choice of j_i gives

$$c = \sigma x_i + \mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \Delta_{i-1}\right) \geq \sigma x_i + \mathrm{EDF}_{(m+\mu,\sigma)}\left(j_i, \bigcup_{s=1}^{i-1} Y_s\right) > \sigma x_i + f(j_i, \Delta_{i-1}).$$

As the release date of j_i is in $\overline{\Delta_i}$ we can use $r_{j_i} - t_{i-1} \leq \left|\overline{\Delta_i}\right|$ (indeed we have equality here) to conclude that

$$c > \sigma x_i + f(j_i, \Delta_{i-1}) = \sigma x_i + c - \left(r_{j_i} - t_{i-1}\right) \geq \sigma x_i + c - \left|\overline{\Delta_i}\right| \ ,$$

which shows $0 > \sigma x_i - \left|\overline{\Delta_i}\right|$ for the case $f(j_i, \Delta_{i-1}) > 0$. Yet, if $f(j_i, \Delta_{i-1}) = 0$ we immediately have $\left|\overline{\Delta_i}\right| \geq c \geq \sigma x_i$.

So, for the case $\left(\left|\overline{\Delta_i}\right| < T\right)$ we also get $\frac{\sigma x_i}{\left|\overline{\Delta_i}\right|} \leq 1$, yielding property 4.

The breaking condition. In each step from $i-1$ to i the interval is strictly extended backwards to the release date of at least one job which is released before t_0. As there are finitely many task, and all have positive minimum separation time T, there are finitely many such jobs, and we can make only finitely many steps. So at some point the breaking condition, namely that there is no job j_i with the two required properties, must hold.

If this holds we claim property 5 to be true, i.e., $W(\Delta_z) \geq W^{\Delta_z}$. In the value W^{Δ_z} we count σx_i for each X_i, because the whole τ-demand processed in a τ-$\mathrm{EDF}_{(m+\mu,\sigma)}$-busy interval is part of the necessary demand of that interval. Also, the demand $\mathrm{EDF}_{(m+\mu,\sigma)}$ failed to process until t_0 is part of the necessary demand of Δ_z. For each Y_i part we count $(m+\mu)\sigma y_i$, which is by property 2 exactly what is processed in those times by $\mathrm{EDF}_{(m+\mu,\sigma)}$. By property 3 all jobs processed in some Y_i have their deadline in the interval Δ_i and therefore also in Δ_z. Finally, there is no job among those processed in some section Y_i with release date before t_z, which has been counted in the term $(m+\mu)\sigma y_i$ with more than its forced forward demand in Δ_i. The forced forward demand in the greater interval Δ_z can only be greater, and thus we count for no job more in W^{Δ_z} than in $W(\Delta_z)$.

We required $c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$. Both are easy to test in linear time. In fact, the later condition is necessary for scheduling any realization on any number of machines with speed 1. The first condition is necessary for scheduling all realization on any number of speed-1 machines.

If an instance I allows for a realization R with an interval Δ generating a necessary demand $W_R(\Delta) > m|\Delta|$ as in the theorem, it can of course not be scheduled by any algorithm on m speed-1 machines. So, all three conditions of the theorem, $c_\tau \leq T_\tau$, $c_\tau \leq D_\tau$, and $W_R(\Delta) \leq m|\Delta|$, are necessary for scheduling on m speed-1 machines. By the theorem they are sufficient for scheduling on $(m+\mu)$ speed-σ machines. Therefore, all that is missing for a $(\mu, 2-\frac{1+\mu}{m+\mu})$-approximative feasibility test, is a test whether an instance I can have a realization R with an interval Δ generating a necessary demand $W_R(\Delta) > m|\Delta|$. For this we will provide an FPTAS in the remainder.

4 An FPTAS for load estimation

The following observation facilitates the test:

Lemma 1. *Assume $c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$ for all tasks τ of an instance I. Then, over all intervals $\Delta = [t, t+\ell)$ of a fixed length ℓ and all realizations R of I, the maximal*

necessary demand from a certain task τ is

$$W_{R^*,\tau}(\Delta^*) = c_\tau k + [c_\tau - (T_\tau - (\ell - D_\tau - (k-1)T_\tau))]^+ = c_\tau k + [c_\tau + \ell - D_\tau - kT_\tau]^+$$

where $k = \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor$.

Proof. Make $t^* + \ell$ the deadline of some job j from τ and $t^* \geq T_\tau$. Further choose R^* such that all jobs of task τ released in $[t - T_\tau, t + \ell)$ preceed their follower at the minimum distance T_τ. Then the necessary demand $W_{R^*,\tau}(\Delta^*)$ is as claimed.

To see that this is maximal, assume any interval Δ with $|\Delta| = \ell$ and any realization R of I with higher necessary demand than the one in the above construction. As $c_\tau \leq T_\tau$, at most $k+1$ jobs can contribute to $W_{R,\tau}(\Delta)$. Compressing the distances between all contributing jobs cannot diminish the forced forward demand in the interval for any of those jobs. Now push the compressed sequence of contributing jobs towards the right until the deadline of the last job coincides with the right boundary of Δ. This will not diminish the forced forward demand of any contributing job. Thus, we arrive at a realization and an interval as in the above construction which generate at least as much forced forward demand as the pair (R, Δ) with which we started. This contradicts that $W_{R,\tau}(\Delta) > c_\tau k + [c_\tau + \ell - D_\tau - kT_\tau]^+$.

The construction of the lemma also shows, that the maximal forced forward demand can be achieved for each task independently. As a consequence we only have to find the optimal length of an interval. Then we know how much forced forward demand a maximal pair of interval and realization has. We define for any instance I satisfying for all $\tau : c_\tau \leq T_\tau$ and $c_\tau \leq D_\tau$:

$$w := w_I : \mathbb{R}^+ \to \mathbb{R}^+, \ell \mapsto w(\ell) := w_I(\ell) := \sum_{\tau \in I} c_\tau k + [c_\tau + \ell - D_\tau - kT_\tau]^+$$

Lemma 1 states that $w_I(\ell)$ is the maximum forced forward demand of any realization of I in any interval of length ℓ.

The following algorithm finds a length ℓ' which approximates the maximum of $\frac{w(\ell)}{\ell}$ by a factor of ε in time polynomial in the input size of I and $1/\varepsilon$. In fact, we devise a function ϕ which pointwise approximates the load, i.e., $\forall \ell \in \mathbb{R}^+ : (1-\varepsilon)\frac{w(\ell)}{\ell} \leq \phi(\ell) \leq \frac{w(\ell)}{\ell}$. There is a polynomial size subset of $\mathbb{R}^+$, a priori determinable, in which the function ϕ must achieve its maximum. So, the approximation algorithm is straightforward.

Algorithm 1 Load Estimation(ε)

For $\tau \in I$, compute threshold$(\tau) := D_\tau + T_\tau/\varepsilon$.
Compute points$(\tau) := \{\ell \in (0, \text{threshold}(\tau)] : \ell = q \cdot T_\tau + D_\tau \text{ for some } q \in \mathbb{N}\}$,
points$'(\tau) := \{\ell \in (0, \text{threshold}(\tau)] : \ell = q \cdot T_\tau + D_\tau - c_\tau \text{ for some } q \in \mathbb{N}\}$,
POINTS $:= \cup_{\tau \in I} \text{points}(\tau) \cup \text{points}'(\tau) \cup \{\text{threshold}(\tau)\}$.
Output $\lambda = \max\left(\max_{\ell \in \text{POINTS}} \frac{w(\ell)}{\ell}, \sum_{\tau=1}^{n} \frac{c_\tau}{T_\tau}\right)$.

Lemma 2. *For any instance I Algorithm 1 outputs a λ such that $(1-\varepsilon)\lambda^* \leq \lambda \leq \lambda^*$ where $\lambda^* = \sup_{\Delta,R} \frac{W_R(\Delta)}{|\Delta|}$, and has running time polynomial in n and $1/\varepsilon$.*

Proof. We know that $\lambda^* = \sup_\ell \frac{w(\ell)}{\ell}$. We show that for all $\ell \geq 0$ the function

$$\phi(\ell) := \sum_{\tau:\text{threshold}(\tau)\geq\ell} \frac{w_\tau(\ell)}{\ell} + \sum_{\tau:\text{threshold}(\tau)<\ell} \left(1 - \frac{D_\tau}{\ell}\right) \frac{c_\tau}{T_\tau}$$

approximates the load $w(\ell)/\ell$ in the following sense:

$$(1-\varepsilon)\frac{w(\ell)}{\ell} \leq \phi(\ell) \leq \frac{w(\ell)}{\ell}.$$

Secondly, we will show that we can find the maximum of ϕ by only considering points in POINTS. The number of points in POINTS is obviously polynomial in the input, and so is the evaluation of ϕ for each point. This completes the proof.

Recall that

$$w_\tau(\ell) = c_\tau \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor + \left[c_\tau + \ell - D_\tau - \left\lfloor \frac{\ell + T_\tau - D_\tau}{T_\tau} \right\rfloor \cdot T_\tau \right]^+.$$

Therefore $w_\tau(\ell)T_\tau \geq c_\tau(\ell - D_\tau)$ and

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

which summed over all tasks τ yields the upper bound on ϕ.

Concerning the lower bound, for $\ell > \text{threshold}(\tau)$ we have $\ell > D_\tau + \frac{T_\tau}{\varepsilon}$ implying $\varepsilon > \frac{T_\tau}{\ell - D_\tau}$. Using again $w_\tau(\ell)T_\tau \geq c_\tau(l - D_\tau)$ gives $\varepsilon > \frac{c_\tau}{w_\tau(\ell)}$.

As the difference between the necessary demand of one task τ and the approximate demand $(\ell - D_\tau) \cdot \frac{c_\tau}{T_\tau}$ can at most be the execution time of the task, c_τ, we can substitute

$$\frac{w_\tau(\ell) - (\ell - D_\tau) \cdot \frac{c_\tau}{T_\tau}}{w_\tau(\ell)} < \varepsilon$$

and by rewriting we get

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

Again, summing over all tasks gives the claimed lower bound on ϕ.

To finish, observe that between two consecutive points $\ell_1, \ell_2 \in \text{POINTS} \cup \{0\}$ we can write

$$\phi(\ell) = C_1 \frac{1}{\ell} + C_2 + \xi(\ell), \quad \forall \ell \in [\ell_1, \ell_2),$$

with

$$C_1 := \sum_{\tau:\text{threshold}(\tau)\geq\ell_1} w_\tau(\ell_1) - \sum_{\tau:\text{threshold}(\tau)<\ell_1} \frac{c_\tau D_\tau}{T_\tau}$$

$$C_2 := \sum_{\tau:\text{threshold}(\tau)<\ell_1} \frac{c_\tau}{T_\tau}$$

$$\xi(\ell) := \frac{\sum_{\tau:\text{threshold}(\tau)\geq\ell_1} \left(\left[c_\tau + \ell - D_\tau - k_1 T_\tau\right]^+ - \left[c_\tau + \ell_1 - D_\tau - k_1 T_\tau\right]^+ \right)}{\ell}$$

where $k_1 := \left\lfloor \frac{\ell_1 + T_\tau - D_\tau}{T_\tau} \right\rfloor$.

By definition of POINTS, the function ξ can be written as $C/\ell + C'$ for some constants C, C'; this implies that the same is true for the function ϕ inside each interval $[\ell_1, \ell_2)$. Thus, a maximum of ϕ is always attained at an extreme point of such an interval. Also, beyond the maximum of POINTS, the function ϕ equals $\sum_{\tau\in I} \left(1 - \frac{D_\tau}{\ell}\right) \cdot \frac{c_\tau}{T_\tau}$. Therefore, the overall maximum of ϕ is attained at one of the points in POINTS or equals $\sum_{\tau\in I} \frac{c_\tau}{T_\tau}$, and the algorithm is correct.

Theorem 2. *There exists a feasibility test that, given a task system I, $\mu \in \mathbb{N}$ and $\varepsilon > 0$, decides whether I can be scheduled by EDF on $(m+\mu)$ speed-$(1 + \frac{m}{(m+\mu)(1-\varepsilon)} - \frac{1}{m+\mu})$ machines, or I cannot be scheduled at all on m speed-1 machines. The running time is polynomial in n and $1/\varepsilon$.*

Proof. With the help of Algorithm 1 we can verify in polynomial time the following conditions:

(C1) For all tasks $\tau \in I : c_\tau \leq \min(D_\tau, T_\tau)$.

(C2) There is $\lambda \leq m$, where $(1-\varepsilon)\lambda^* \leq \lambda \leq \lambda^*$ and $\lambda^* = \sup_{R,\Delta} \frac{W_R(\Delta)}{|\Delta|}$.

Both are necessary for scheduling I on m speed-1 machines.

Condition (C2) implies that there is no realization R and interval Δ such $W_R(\Delta) > \frac{m}{1-\varepsilon}|\Delta|$. Choosing $\sigma \geq (1 + \frac{m}{(m+\mu)(1-\varepsilon)} - \frac{1}{m+\mu})$ gives $(m+\mu)(\sigma-1)+1 \geq \frac{m}{(1-\varepsilon)}$. Plugging this into the argument of Theorem 1 shows that I can be scheduled by EDF on $(m+\mu)$ speed-σ machines if conditions (C1) and (C2) are true.

Corollary 1. *There exists a feasibility test that, given a task system I and $\varepsilon > 0$, decides whether I can be scheduled by EDF on m speed-$(2 - 1/m + \varepsilon)$ machines, or I cannot be scheduled at all on m speed-1 machines. The running time is polynomial in n and $1/\varepsilon$.*

Acknowledgments. The authors acknowledge Enrico Bini 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 Conference 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 Trans. Parallel Distrib. Syst.*, 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*. CRC Press, 2007.
5. S. K. Baruah, R. R. Howell, and L. E. Rosier. Feasibility problems for recurring tasks on one processor. *Theor. Comput. Sci.*, 118(1):3–20, 1993.
6. S. Chakraborty, S. Künzli, and L. Thiele. Approximate schedulability analysis. In *Proc. 23rd IEEE Real-Time Systems Symposium*, pages 159–168, 2002.
7. M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *Proc. IFIP Congress*, pages 807–813, 1974.
8. 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 Conference on Real-Time Systems*, pages 117–126, 2005.
9. T. W. Lam and K.-K. To. Trade-offs between speed and processor in hard-deadline scheduling. In *Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 1999.
10. J. Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Inf. Process. Lett.*, 11(3):115–118, 1980.
11. C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.
12. C. A. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. *Algorithmica*, 32(2):163–200, 2002.