Real-Time Syst (2008) 38: 223–235
DOI 10.1007/s11241-007-9047-9

Schedulability analysis of global EDF

Sanjoy Baruah · Theodore Baker

Published online: 18 December 2007

Abstract The multiprocessor EDF scheduling of sporadic task systems is studied. A new sufficient schedulability test is presented and proved correct. It is shown that this test generalizes the previously-known exact uniprocessor EDF-schedulability test, and that it offers non-trivial quantitative guarantees (including a resource augmentation bound) on multiprocessors.

Keywords Multiprocessor scheduling · Sporadic task systems · EDF · Global scheduling · Schedulability analysis

1 Introduction

A real-time system is often modelled as a finite collection of independent recurring tasks, each of which generates a potentially infinite sequence of *jobs*. Every job is characterized by an arrival time, an execution requirement, and a deadline, and it is required that a job completes execution between its arrival time and its deadline. Different formal models for recurring tasks place different restrictions on the values of the parameters of jobs generated by each task. One of the more commonly used formal models is the *sporadic task model* (Mok 1983; Baruah et al. 1990).

In this paper, we consider real-time systems that are modeled by the sporadic task model and implemented upon a platform comprised of several identical processors. We assume that the platform is fully *preemptive*—an executing job may be interrupted at any instant in time and have its execution resumed later with no cost or

S. Baruah (✉)
The University of North Carolina, Chapel Hill 27599, USA
e-mail: baruah@cs.unc.edu

T. Baker
Floride State University, Tallahassee, FL, USA

penalty. We study the behavior of the well-known *Earliest Deadline First* scheduling algorithm (Liu and Layland 1973; Dertouzos 1974) when scheduling systems of sporadic tasks upon such preemptive platforms. We assume that the platform allows for *global* inter-processor migration—a job may begin execution on any processor and a preempted job may resume execution on the same processor as, or a different processor from, the one it had been executing on prior to preemption. (However, each job may execute on at most one processor at each instant in time.) We will refer to Earliest Deadline First scheduling with global inter-processor migration as *global* EDF (or simply EDF).

Our results We derive a new test for determining whether a given sporadic task system is guaranteed to be scheduled to meet all deadlines upon a specified computing platform, when scheduled using EDF. We prove that this test has several desirable properties that are not possessed by earlier tests: it generalizes known optimal uniprocessor tests, and it offers quantitative performance guarantees upon multiprocessors.

Organization The remainder of this paper is organized as follows. In Sect. 2 we formally define the sporadic task model and provide some additional useful definitions. In Sect. 3, we describe some related research, and present previously known schedulability tests for global EDF. In Sect. 4 we derive, and prove the correctness of, a new global-EDF schedulability test. In Sect. 5 we provide a quantitative characterization of the efficacy of this new schedulability test in terms of the resource augmentation metric.

2 Model

A *sporadic task* $\tau_i = (C_i, D_i, T_i)$ is characterized by a *worst-case execution requirement* C_i, a *(relative) deadline* D_i, and a *minimum inter-arrival separation* parameter T_i, also referred to as the *period* of the task. Such a sporadic task generates a potentially infinite sequence of jobs, with successive job-arrivals separated by at least T_i time units. Each job has a worst-case execution requirement equal to C_i and a deadline that occurs D_i time units after its arrival time. We refer to the interval, of size D_i, between such a job's arrival instant and deadline as its *scheduling window*. We assume a fully *preemptive* execution model: any executing job may be interrupted at any instant in time, and its execution resumed later with no cost or penalty. A *sporadic task system* is comprised of a finite number of such sporadic tasks. Let τ denote a system of such sporadic tasks: $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, with $\tau_i = (C_i, D_i, T_i)$ for all i, $1 \le i \le n$. Task system τ is said to be a *constrained* sporadic task system if it is guaranteed that each task $\tau_i \in \tau$ has its relative deadline parameter no larger than its period: $D_i \le T_i$ for all $\tau_i \in \tau$. *We restrict our attention here to constrained task systems.*

Priority-driven scheduling algorithms operate as follows: at each instant in time they assign a priority to each job that is awaiting execution, and choose for execution the jobs with the greatest priority. The Earliest Deadline First (EDF) scheduling algorithm (Liu and Layland 1973; Dertouzos 1974) is a priority-driven scheduling algorithm that assigns priority to jobs according to their (absolute) deadlines: the earlier the deadline, the greater the priority.

With respect to a given platform, a given sporadic task system is said to be *feasible* if there exists a schedule meeting all deadlines for every collection of jobs that may be generated by the task system. A given sporadic task system is said to be *(global)* EDF *schedulable* if EDF meets all deadlines for every collection of jobs that may be generated by the task system. While every EDF-schedulable task system is trivially feasible, it is known that not all feasible task systems are EDF-schedulable. (This is in contrast to the preemptive *uni*processor case, where it is known that EDF is an *optimal* algorithm: every feasible system is guaranteed to be EDF-schedulable as well.)

We find it convenient to define some properties and parameters for individual sporadic tasks, and for sporadic task systems.

Utilization The utilization u_i of a task τ_i is the ratio C_i/T_i of its execution requirement to its period. The total utilization $u_{\text{sum}}(\tau)$ and the largest utilization $u_{\max}(\tau)$ of a task system τ are defined as follows:

$$u_{\text{sum}}(\tau) \overset{\text{def}}{=} \sum_{\tau_i \in \tau} u_i; \qquad u_{\max}(\tau) \overset{\text{def}}{=} \max_{\tau_i \in \tau}(u_i).$$

Density The density δ_i of a task τ_i is the ratio C_i/D_i of its execution requirement to its relative deadline. The total density $\delta_{\text{sum}}(\tau)$ and the largest density $\delta_{\max}(\tau)$ of a task system τ are defined as follows:

$$\delta_{\text{sum}}(\tau) \overset{\text{def}}{=} \sum_{\tau_i \in \tau} \delta_i; \qquad \delta_{\max}(\tau) \overset{\text{def}}{=} \max_{\tau_i \in \tau}(\delta_i).$$

DBF For any interval length t, the demand bound function DBF(τ_i, t) of a sporadic task τ_i bounds the maximum cumulative execution requirement by jobs of τ_i that both arrive in, and have deadlines within, any interval of length t. It has been shown (Baruah et al. 1990) that

$$\text{DBF}(\tau_i, t) = \max\left(0, \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right) C_i\right).$$

Load A load parameter, based upon the DBF function, may be defined for any sporadic task system τ as follows:

$$\text{LOAD}(\tau) \overset{\text{def}}{=} \max_{t>0}\left(\frac{\sum_{\tau_i \in \tau} \text{DBF}(\tau_i, t)}{t}\right).$$

Computing DBF (and thereby, LOAD) will turn out to be a critical component of the schedulability analysis test proposed in this paper; hence, it is important that DBF be efficiently computable if this schedulability test is to be efficiently implementable as claimed. Fortunately, computing DBF is a well-studied subject, and algorithms are known for computing DBF *exactly* (Baruah et al. 1990; Ripoll et al. 1996), or *approximately* to any arbitrary degree of accuracy (Baker et al. 2005; Fisher et al. 2006a, 2006b) in pseudo-polynomial or polynomial time respectively.

3 Related Work

Phillips et al. (1997) studied the global EDF scheduling of specified collections of *independent jobs*. They obtained the following resource-augmentation characterization of global EDF:

Theorem 1 (from (Phillips et al. 1997)) *Any collection of independent jobs that can be scheduled to meet all deadlines upon m processors of a given speed will meet all deadlines if scheduled using global* EDF *upon m processors each of which is* $(2 - \frac{1}{m})$ *times as fast.*

This is an interesting and important result, but in order to apply it to sporadic task systems we would first need to determine whether a given sporadic task system is feasible. Unfortunately, no exact (i.e., necessary and sufficient), or non-trivial sufficient, feasibility tests are currently known for arbitrary (or even constrained) sporadic task systems.

Implicit-deadline or *Liu and Layland* task systems are sporadic task systems in which all tasks' relative deadline parameters are equal to their inter-arrival separation parameters (i.e., $D_i = T_i (\forall i)$). Feasibility-analysis for such systems is trivial: implicit-deadline sporadic task system τ is feasible upon m unit-capacity processors if and only if $u_{\max}(\tau) \le 1$ and $u_{\text{sum}}(\tau) \le m$. Based on this observation, the techniques of Phillips et al. (1997) were extended (Srinivasan and Baruah 2002; Goossens et al. 2003; Baruah 2004) to show that

$$u_{\text{sum}}(\tau) \le m - (m-1)u_{\max}(\tau)$$

is a sufficient condition for implicit-deadline sporadic task system τ to be global EDF-schedulable upon m unit-capacity processors. (This condition was also shown to be tight in the sense that there are implicit-deadline task systems τ with $u_{\text{sum}}(\tau)$ exceeding the right-hand side (RHS) of the above condition by an arbitrarily small amount, upon which global EDF misses deadlines.) Minor extensions to the proofs in Goossens et al. (2003) can be used to obtain the following sufficient global-EDF schedulability test for constrained sporadic task systems:

$$\delta_{\text{sum}}(\tau) \le m - (m-1)\delta_{\max}(\tau). \tag{1}$$

This test is often referred to in the literature as the *density* test for global EDF-schedulability of constrained sporadic task systems.

Baker (2003, 2005), Bertogna et al. (2005) designed a test from first principles for global-EDF schedulability analysis of sporadic task systems. In essence, Baker's test—henceforth referred to here as the [BAK] test—is obtained by assuming that a task τ_k's job misses its deadline, and then determining necessary conditions on the parameters of all the tasks that must be satisfied in order for such a deadline miss to occur. Then *negating* these conditions for each τ_k yields a sufficient test for global-EDF schedulability.

In 2005, Bertogna et al. (2005) used ideas very similar to Baker's to come up with a simpler test that occasionally outperformed both the density and the [BAK] tests. The test from Bertogna et al. (2005) will be referred to here as the [BCL] test.

Overview of the [BAK] *and* [BCL] *tests* Both tests are built around the following general strategy, first introduced by Baker (2003). Suppose that τ is not global EDF-schedulable, and consider a legal sequence of job requests of τ on which global EDF misses one or more deadlines. Suppose that a job of τ_k is the first job that misses its deadline, at some time-instant t_d. Both tests analyze the situation over some interval $[t_o, t_d)$, where t_o is different in the [BAK] and [BCL] tests (and is precisely defined for each test). Both tests compute upper bounds on the amount of work that EDF can be required to execute over the interval $[t_o, t_d)$, and obtain an unschedulability condition by setting this bound to be large enough to deny τ_k's job C_k units of execution over its scheduling window. The parameters of the tasks in the task system must satisfy a certain inequality (which is different for the two tests) in order that the bound be large enough; the sufficient schedulability conditions for both tests are obtained by negating this inequality.

In bounding the total amount of work that each task τ_i needs to have executed over $[t_o, t_d)$ in the EDF schedule, one must consider (i) jobs of τ_i that arrive within the interval, *and* (ii) possibly one additional job that arrives prior to t_o but has not completed execution by time-instant t_o and hence "carries in" some execution into the interval. The contribution of jobs arriving within the interval may be computed using the demand bound abstraction (Sect. 2 above), but new techniques are needed for bounding the carry-in. Both the [BAK] and [BCL] tests use different bounds on the amount of such work for each τ_i, and then sum over all i to obtain an upper bound on the total amount of work to be completed over $[t_o, t_d)$.

Differences between the [BAK] *and* [BCL] *tests* Despite the similarities in overall approach, the two tests differ quite significantly in the details:

1. In [BCL], t_o is set equal to $(t_d - D_k)$; i.e., the arrival time of the job that misses its deadline. In [BAK], it is set equal to the earliest time-instant prior to this job's arrival time, at which a certain condition[1] is satisfied.
2. In [BCL], fairly primitive techniques are used to bound the amount of work that τ_i may contribute to this interval, while [BAK] uses a far more sophisticated analysis. Specifically, while [BAK] is able to show that parts of "carry-in" jobs—jobs whose scheduling windows span t_o—must have completed execution before t_o, [BCL] pessimistically assumes that each carry-in job executes as late as possible, immediately before their deadlines and thereby carries in as much work as possible.

   Unfortunately, the more sophisticated analysis of [BAK] does not allow one to use some simpler observations that [BCL] were able to exploit. It seems that these simpler observations are often more useful in practice; so that in many cases the [BCL] test is able to provide a tighter bound than the [BAK] test on the amount of work carried in into the interval $[t_o, t_d)$.
3. The length of the interval $[t_o, t_d)$ could be larger in the [BAK] tests than in the [BCL] test. This is desirable since any additive inaccuracy introduced in computing bounds on the amount of work that must be executed over $[t_o, t_d)$ gets

[1] It is not essential for us to know what this condition is, in order to understand the remainder of this paper. Please consult Baker (2005) for details.

amortized over a larger interval. (However, a closer examination of the [BAK] test indicates that this potential advantage is not exploited—a final step in the derivation of the [BAK] test lower-bounds the size of the interval $[t_o, t_d)$ by D_k, which is the value used by the [BCL] test as well.)

Comparison Although the [BAK] test employs considerably more sophisticated analysis than the [BCL] test, it does not unequivocally outperform the [BCL] test. Baker has conducted extensive simulation experiments comparing the density, [BAK] and [BCL] tests. These simulations have demonstrated that all three tests are incomparable: there are task systems deemed EDF-schedulable by each test but not by the other two tests. They also seem to demonstrate that the density and [BCL] tests, taken together, are able to cover most of the task systems that are handled by the [BAK] test.

Shortcomings common to both tests In analyzing a possible deadline miss for τ_k both the [BAK] and [BCL] tests consider a "worst-case" scenario in which it is assumed that *every* one of the n tasks in the system carries work into the interval that must be considered. In systems with a large number of tasks ($n \gg m$), this results in severe over-estimation of the cumulative carry-in. This is further aggravated by the feature of both tests—outlined above—of amortizing this carry-in over an interval of size D_k.

4 A global-EDF schedulability test

We now derive (Theorem 2 below) a new schedulability test for constrained sporadic task systems. Our approach towards developing this test follows the same general framework as the derivations of the [BAK] and [BCL] tests: we obtain necessary conditions for global EDF to miss a deadline when scheduling sporadic task system τ upon m unit-capacity processors. By taking the contrapositive of these conditions, we obtain a sufficient schedulability condition.

Consider any legal sequence of job requests of task system τ, on which EDF misses a deadline. Suppose that a job of task τ_k is the one to first miss a deadline, and that this deadline miss occurs at time-instant t_d (see Fig. 1). Let t_a denote this job's arrival time: $t_a = t_d - D_k$.

Discard from the legal sequence of job requests all jobs with deadline $> t_d$, and consider the EDF schedule of the remaining (legal) sequence of job requests. Since later-deadline jobs have no effect on the scheduling of earlier-deadline ones under

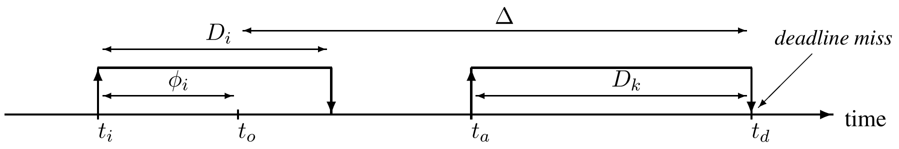

Fig. 1 Notation. A job of task τ_k arrives at t_a and misses its deadline at time-instant t_d

preemptive EDF, it follows that a deadline miss of τ_k occurs at time-instant t_d (and this is the earliest deadline miss), in this new EDF schedule.

We introduce some notation now. For any time-instant $t \le t_d$:

- Let $W(t)$ denote the cumulative execution requirement of all the jobs in this legal sequence of job requests, *minus* the total amount of execution completed by the EDF schedule prior to time-instant t.
- Let $\Omega(t)$ denote $W(t)$ normalized by the interval-length: $\Omega(t) \stackrel{\text{def}}{=} W(t)/(t_d - t)$.

Since τ_k's job receives strictly less than C_k units of execution over $[t_a, t_d)$, all m processors must be executing tasks other than τ_k's job for a total duration greater than $(D_k - C_k)$ over this interval. Hence it must be the case that $W(t_a) > (D_k - C_k)m + C_k$; equivalently,

$$\begin{aligned}\Omega(t_a) &> \frac{(D_k - C_k)m + C_k}{D_k} \\ &= m - (m-1)\frac{C_k}{D_k} \\ &\ge m - (m-1)\delta_{\max}(\tau).\end{aligned}$$

Let

$$\mu \stackrel{\text{def}}{=} m - (m-1)\delta_{\max}(\tau). \tag{2}$$

Let t_o denote the smallest value of $t \le t_a$ such that $\Omega(t) \ge \mu$. Let $\Delta \stackrel{\text{def}}{=} t_d - t_o$.

By definition, $W(t_o)$ denotes the amount of work that the EDF schedule needs (but fails) to execute over $[t_o, t_d)$. This work in $W(t_o)$ arises from two sources: those jobs that arrived at or after t_o, and those that arrived prior to t_o but have not completed execution in the EDF schedule by time-instant t_o. We will refer to jobs arriving prior to t_o that need execution over $[t_o, t_d)$ as ***carry-in jobs.***

Lemma 1 *Each carry-in job has* $<\Delta \times \delta_{\max}(\tau)$ *remaining execution requirement at time-instant* t_o.

Proof Let us consider a carry-in job of some task τ_i, that arrives at time-instant $t_i < t_o$ and has (by definition of carry-in job) not completed execution by time-instant t_o. Let $\phi_i \stackrel{\text{def}}{=} t_o - t_i$ (see Fig. 1).

By definition of t_o, it must be the case that $\Omega(t_i) < \mu$. That is,

$$W(t_i) < \mu(\Delta + \phi_i). \tag{3}$$

On the other hand, $\Omega(t_o) \ge \mu$, meaning that

$$W(t_o) \ge \mu\Delta. \tag{4}$$

From inequalities (3) and (4) and the definition of $W(\cdot)$, it follows that the amount of work done in the EDF schedule over $[t_i, t_o)$ is *less than* $\mu\phi_i$. Let y_i denote the amount of time over this interval $[t_i, t_o)$, during which τ_i's job is executing. All m processors must be executing other jobs for the remaining $(\phi_i - y_i)$ time units, implying that the total amount of work done in the EDF schedule over $[t_i, t_o)$ is *at least* $m(\phi_i - y_i) + y_i$. From these upper and lower bounds, we have

$$\begin{aligned} & m(\phi_i - y_i) + y_i < \mu\phi_i \\ & \quad \equiv m\phi_i - (m-1)y_i < (m - (m-1)\delta_{\max}(\tau))\phi_i \\ & \quad \equiv y_i > \phi_i \delta_{\max}(\tau). \end{aligned} \tag{5}$$

Inequality (5) is important—it tells us that task τ_i must have already completed a significant amount of its execution *before* time-instant t_o. More specifically, the remaining work of this job of τ_i, which contributes to $W(t_o)$, is given by

$$(C_i - y_i) < (D_i\delta_{\max}(\tau) - \phi_i\delta_{\max}(\tau)) = \delta_{\max}(\tau)(D_i - \phi_i).$$

Moreover, since the (absolute) deadline of this job of τ_i—at $t_i + D_i$—is, by construction, $\le t_d$, it follows that $(D_i - \phi_i) \le \Delta$; i.e., each carry-in job contributes at most $\Delta \times \delta_{\max}(\tau)$ to $W(t_o)$. □

And how many carry-in jobs can there be? This question is addressed in the following lemma.

Lemma 2 *The number of carry-in jobs is strictly bounded from above by* $\lceil\mu\rceil - 1$.

Proof Let ϵ denote an arbitrarily small positive number. By definition of the instant t_o, $\Omega(t_o - \epsilon) < \mu$ while $\Omega(t_o) \ge \mu$; consequently, strictly fewer than μ jobs must have been executing at time-instant t_o. And since $\mu \le m$ (as can be seen from (2) above), it follows that some processor was idled over $[t_o - \epsilon, t_o)$, implying that all jobs active at this time would have been executing. This allows us to conclude that there are strictly fewer than μ—equivalently, at most $(\lceil\mu\rceil - 1)$—carry-in jobs. □

Based upon Lemmas 1 and 2, we obtain the following schedulability condition for global EDF:

Theorem 2 *Sporadic task system τ is global-*EDF *schedulable upon a platform comprised of m unit-capacity processors, provided*

$$\text{LOAD}(\tau) \le \mu - (\lceil\mu\rceil - 1)\delta_{\max}(\tau), \tag{6}$$

where μ is as defined in (2).

Proof The proof is by contradiction: we obtain necessary conditions for the scenario above—when τ_k's job misses its deadline at t_d—to occur. Negating these conditions yields a sufficient condition for global-EDF schedulability.

Let us bound the total amount of execution that contributes to $W(t_o)$.

- First, there are the carry-in jobs: by Lemmas 1 and 2, there are at most $(\lceil\mu\rceil - 1)$ of them, each contributing at most $\Delta\delta_{\max}(\tau)$ units of work. Therefore their total contribution to $W(t_o)$ is bounded from above by $(\lceil\mu\rceil - 1)\Delta\delta_{\max}(\tau)$.

- All other jobs that contribute to $W(t_o)$ arrive in, and have deadlines within, the Δ-sized interval $[t_o, t_d)$. Their total execution requirement is therefore bounded from above by $\Delta \times \textsc{load}(\tau)$.

We consequently obtain the following bound on $W(t_o)$:

$$W(t_o) < \Delta\textsc{load}(\tau) + (\lceil\mu\rceil - 1)\Delta\ \delta_{\max}(\tau). \tag{7}$$

Since $\Omega(t_o) \overset{\text{def}}{=} W(t_o)/(t_d - t_o)$ and $(t_d - t_o) = \Delta$, we derive

$$\Omega(t_o) < \textsc{load}(\tau) + (\lceil\mu\rceil - 1)\delta_{\max}(\tau)$$

from inequality (7) above.

Since, by the definition of t_o, it is required that $\Omega(t_o)$ be at least as large as μ, we must have

$$\textsc{load}(\tau) + (\lceil\mu\rceil - 1)\ \delta_{\max}(\tau) > \mu$$

as a necessary condition for EDF to miss a deadline; equivalently, the negation of this condition is sufficient to ensure EDF-schedulability:

$$\begin{aligned} &\textsc{load}(\tau) + (\lceil\mu\rceil - 1)\delta_{\max}(\tau) \le \mu \\ &\equiv \textsc{load}(\tau) \le \mu - (\lceil\mu\rceil - 1)\delta_{\max}(\tau) \end{aligned}$$

which is as claimed in the theorem. □

Observe that Theorem 2 uses the parameters $\textsc{load}(\tau)$ and $\delta_{\max}(\tau)$ of sporadic task system τ in order to determine whether the system is EDF-schedulable. In other words, the parameters of the individual tasks comprising the task system are distilled into these two composite parameters, and determination regarding EDF-schedulability is made based upon the values of these composite parameters. The guarantee is that all systems τ whose $(\textsc{load}(\tau), \delta_{\max}(\tau))$ parameters satisfy the condition of the theorem are guaranteed EDF-schedulable while systems whose parameters do not satisfy this condition may or may not be EDF-schedulable.

This load bound of Theorem 2 above is depicted graphically in Fig. 2, when the number of processors m is very large.

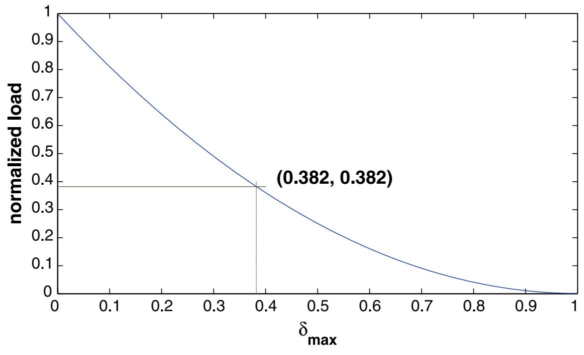

Fig. 2 Graphical depiction of Theorem 2. The bound on $\textsc{load}(\tau)/m$ is represented as a function of $\delta_{\max}(\tau)$, for $m \to \infty$

Optimality for uniprocessors The earlier tests—the density, [BAK] and [BCL] tests—are not exact even for uniprocessor systems (the special case when $m = 1$). (This is also indicated by the fact that all these prior tests have polynomial run-time, while EDF-schedulability analysis of sporadic task systems on uniprocessors is not known to be in polynomial time.) The following corollary asserts that the test of Theorem 2 is superior to earlier tests in this regard:

Corollary 1 *The test of Theorem 2 is exact for uniprocessors.*

Proof For $m = 1$, it is easily seen (2) that μ always evaluates to one. Consequently, the RHS of the condition in Theorem 2 evaluates to one, and the test becomes

$$\textsc{load}(\tau) \le 1.$$

And this is exactly the uniprocessor preemptive EDF-schedulability test from, e.g., Baruah et al. (1990). □

5 A resource augmentation result

As stated in Sect. 3 (Theorem 1) above, Phillips et al. (1997) related *feasibility* and EDF-*schedulability* of collections of independent jobs by means of a resource-augmentation result: any collection of independent jobs that can be scheduled to meet all deadlines upon m processors of a given speed will meet all deadlines if scheduled using global EDF upon m processors each of which is $(2 - \frac{1}{m})$ times as fast. In this section, we obtain a similar resource-augmentation result for our EDF-schedulability test of Theorem 2.

First, a technical lemma.

Lemma 3 *Any sporadic task system τ that is feasible upon a multiprocessor platform comprised of m speed-x processors must satisfy*

$$\delta_{\max}(\tau) \le x \quad \text{and} \quad \textsc{load}(\tau) \le mx. \tag{8}$$

Proof Suppose that task system τ is feasible upon m speed-x processors. Observe that a job of any task τ_i, with execution-requirement C_i and relative-deadline D_i, can receive at most $D_i \times x$ units of execution by its deadline on such a platform; hence, we must have $C_i \le D_i \times x$. Taken over all tasks in τ, this observation yields the first condition.

For the second condition, recall the definition of $\textsc{load}(\tau)$ from Sect. 2. Let t' denote some value of t which defines $\textsc{load}(\tau)$:

$$t' \overset{\text{def}}{=} \operatorname{argmax}\left(\frac{\sum_{\tau_i \in \tau} \text{DBF}(\tau_i, t)}{t}\right).$$

Suppose that all tasks in τ generate a job at time-instant zero, and each task τ_i generates subsequent jobs exactly T_i time-units apart. The total amount of execution that is available over the interval $[0, t')$ on this platform is equal to mxt'; hence, it is necessary that $\text{LOAD}(\tau) \le mx$ if all deadlines are to be met. $\square$

Lemma 3 may be interpreted graphically: in Fig. 2, this lemma (with $x \leftarrow 1$) asserts that all task systems that lie outside the outer rectangle—i.e., with normalized load > 1 or with maximum density >1—are guaranteed to not be schedulable upon a platform comprised of unit-capacity processors.

We can use Lemma 3 to derive a resource-augmentation bound for the EDF-schedulability test of Theorem 2 above.

Corollary 2 *Any sporadic task system that is feasible upon a multiprocessor platform comprised of m speed-$\frac{3-\sqrt{5}}{2}$ processors is determined to be global-*EDF *schedulable on m unit-capacity processors by the* EDF-*schedulability test of Theorem 2.*

Proof Suppose that τ is feasible upon a platform comprised of m speed-x processors. We will prove that τ is determined to be EDF-schedulable upon m unit-capacity processors by the test of Theorem 2 for all $x \le \frac{3-\sqrt{5}}{2}$.

From Lemma 3, it must be the case that $\text{LOAD}(\tau) \le mx$ and $\delta_{\max}(\tau) \le x$. For τ to be determined to be EDF-schedulable upon m unit-capacity processors by the test of Theorem 2, it is sufficient that:

$$\begin{aligned}
&\text{LOAD}(\tau) \le \mu - (\lceil \mu \rceil - 1)\delta_{\max}(\tau) \quad \Leftarrow \quad (\text{Since } \lceil \alpha \rceil - 1 \le \alpha \text{ for all } \alpha)\\
&\qquad \text{LOAD}(\tau) \le \mu - \mu\, \delta_{\max}(\tau)\\
&\equiv \text{LOAD}(\tau) \le \mu(1 - \delta_{\max}(\tau))\\
&\equiv \text{LOAD}(\tau) \le (m - (m-1)\delta_{\max}(\tau))(1 - \delta_{\max}(\tau))\\
&\quad \Leftarrow \quad mx \le (m - (m-1)x)(1-x)\\
&\equiv mx \le (m(1-x) + x)(1-x) \quad \Leftarrow \quad mx \le m(1-x)(1-x)\\
&\equiv x^2 - 3x + 1 \ge 0 \quad \Leftarrow \quad x \le \frac{3-\sqrt{5}}{2}.
\end{aligned}$$

That is, τ being feasible upon m speed-$\frac{3-\sqrt{5}}{2}$ processors is sufficient to ensure that τ satisfies the sufficient schedulability test of Theorem 2. $\square$

It may be verified that $\frac{3-\sqrt{5}}{2} \approx 0.382$; hence, Corollary 2 is stating that the point $(0.382, 0.382)$ lies on the normalized load versus density bound curve depicted in Fig. 2. This point is also depicted in the figure.

We already know that global EDF is not an optimal scheduling algorithm. We have also seen that the schedulability test of Theorem 2 is not optimal (except for the special case $m = 1$). The significance of this resource-augmentation result lies in what it tells us about the “goodness” of both global EDF and of our schedulability test: in essence, it is asserting that a processor speedup of $(2/(3-\sqrt{5}) \approx 2.62$ compensates for *both* the non-optimality of global EDF *and* the inexactness of our schedulability test.

6 Conclusions

We have obtained a new schedulability test for the global EDF scheduling of constrained sporadic task systems upon preemptive multiprocessor platforms. This test characterizes a task system by its LOAD and $\delta_{\max}$ parameters; for each task system, it is able to guarantee that either the task system *is* EDF-schedulable upon the specified platform, or it *is not* feasible upon a platform approximately 0.382 times as fast. To our knowledge, this is the first global schedulability test for constrained-deadline sporadic task systems (with respect to any scheduling algorithm) that offers such a non-trivial performance guarantee.

The major insight upon which our test is based, and which serves to distinguish it from the [BAK] and [BCL] tests, is this. Consider a system of n tasks, to be scheduled upon m processors. In deriving sufficient conditions for schedulability of a particular task, prior tests must allow for the possibility that all remaining $(n-1)$ tasks contribute carry-in load. By contrast, our test only allows for $(m-1)$ tasks to contribute carry-in load. The benefit of this is likely to be particularly pronounced in systems with $n \gg m$; as n becomes smaller, this advantage is no longer quite as pronounced.

Another major difference between our test and the [BAK] and [BCL] tests in that these prior tests tend to be over-pessimistic when different tasks' parameters are of different orders of magnitude (for example, these tests would reject as unschedulable systems where one task's relative deadline parameter is the same as the execution-requirement parameter of m other tasks). However, our test is able to handle task systems in which different tasks have parameters with the same or different orders of magnitude.

We believe that the analysis techniques introduced here are novel and important, and that they will prove useful in studying multiprocessor global schedulability for many more general task and machine models, and for different scheduling algorithms. Indeed, extensions to more general models of recurring real-time tasks that satisfy the *frame separation property*—successive jobs of the same task do not have overlapping scheduling windows—is straightforward.

References

Baker T (2003) Multiprocessor EDF and deadline monotonic schedulability analysis. In: Proceedings of the IEEE real-time systems symposium, December 2003. IEEE Computer Society Press, Los Alamitos, pp 120–129

Baker TP (2005) An analysis of EDF schedulability on a multiprocessor. IEEE Trans Parallel Distrib Syst 16(8):760–768

Baker TP, Fisher N, Baruah S (2005) Algorithms for determining the load of a sporadic task system. Technical Report TR-051201, Department of Computer Science, Florida State University

Baruah S (2004) Optimal utilization bounds for the fixed-priority scheduling of periodic task systems on identical multiprocessors. IEEE Trans Comput 53:6

Baruah S, Mok A, Rosier L (1990) Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Proceedings of the 11th real-time systems symposium, Orlando, Florida. IEEE Computer Society Press, Los Alamitos, pp 182–190

Bertogna M, Cirinei M, Lipari G (2005) Improved schedulability analysis of EDF on multiprocessor platforms. In: Proceedings of the EuroMicro conference on real-time systems, Palma de Mallorca, Balearic Islands, Spain, July 2005. IEEE Computer Society Press, Los Alamitos, pp 209–218

Dertouzos M (1974) Control robotics: The procedural control of physical processors. In: Proceedings of the IFIP congress, pp 807–813
Fisher N, Baker T, Baruah S (2006a) Algorithms for determining the demand-based load of a sporadic task system. In: Proceedings of the international conference on real-time computing systems and applications, Sydney, Australia, August 2006. IEEE Computer Society Press, Los Alamitos
Fisher N, Baruah S, Baker T (2006b) The partitioned scheduling of sporadic tasks according to static priorities. In: Proceedings of the EuroMicro conference on real-time systems, Dresden, Germany, July 2006. IEEE Computer Society Press, Los Alamitos
Goossens J, Funk S, Baruah S (2003) Priority-driven scheduling of periodic task systems on multiprocessors. Real-Time Syst 25(2–3):187–205
Liu C, Layland J (1973) Scheduling algorithms for multiprogramming in a hard real-time environment. J ACM 20(1):46–61
Mok AK (1983) Fundamental design problems of distributed systems for the hard-real-time environment. Ph.D. thesis, Laboratory for Computer Science, Massachusetts Institute of Technology. Available as Technical Report No. MIT/LCS/TR-297
Phillips CA, Stein C, Torng E, Wein J (1997) Optimal time-critical scheduling via resource augmentation. In: Proceedings of the twenty-ninth annual ACM symposium on theory of computing, El Paso, Texas, 4–6 May 1997, pp 140–149
Ripoll I, Crespo A, Mok AK (1996) Improvement in feasibility testing for real-time tasks. Real-Time Syst: Int J Time-Crit Comput 11:19–39
Srinivasan A, Baruah S (2002) Deadline-based scheduling of periodic task systems on multiprocessors. Inf Process Lett 84(2):93–98

Sanjoy Baruah is a professor in the Department of Computer Science at the University of North Carolina at Chapel Hill. He received his Ph.D. from the University of Texas at Austin in 1993. His research and teaching interests are in scheduling theory, real-time and safety-critical system design, and resourceallocation and sharing in distributed computing environments.

Theodore Baker received the Ph.D. in Computer Science from Cornell University in 1973. He is a Professor in the Department of Computer Science at the Florida State University, which he chaired from 1998 to 2005. After spending several years doing research in computational complexity theory, he moved on to more practical aspects of computing and has worked in the area of both Ada compilation and real-time systems for the last two decades. A group he organized at FSU in 1979 produced one of the first validated Ada cross-compilers for embedded systems. Since then, he has done research, development, and consulting related to real-time embedded computing, from basic research on scheduling and concurrency control through development of kernels and run-time system support for real-time programming languages. He has also been active in IEEE (POSIX) and ISO standards work related to real-time systems. Dr. Baker was a member of the SEI Rate Monotonic Analysis group, served as real-time area expert for the Ada 9X language mapping and revision team. He directed the FSU teams that developed several software products, including the FSU POSIX threads library, the Florist implementation of IEEE Std 1003.5b-c (the POSIX/Ada API), a set of validation tests for the 1003.5b standards, and the multitasking run-time system for the Gnu Ada (GNAT) compiler. He directed the porting of the latter to several environments, including the Java Virtual Machine and RT Linux. His current research interests are real-time multiprocessor scheduling and real-time device driver architecture.