Improved multiprocessor global schedulability analysis

Sanjoy Baruah*
The University of North Carolina at Chapel Hill

Vincenzo Bonifaci
Max-Planck Institut für Informatik

Alberto Marchetti-Spaccamela†
Sapienza Università di Roma

Sebastian Stiller‡
Technische Universität Berlin

Abstract

A new technique was recently introduced by Bonifaci et al. for the analysis of real-time systems scheduled on multiprocessor platforms by the global Earliest Deadline First (EDF) scheduling algorithm. In this paper, this technique is generalized so that it is applicable to the schedulability analysis of real-time systems scheduled on multiprocessor platforms by any work-conserving algorithm. The resulting analysis technique is applied to obtain a new sufficient global Deadline Monotonic (DM) schedulability test. It is shown that this new test is quantitatively superior to pre-existing DM schedulability analysis tests; in addition, the degree of its deviation from any hypothetical optimal scheduler (that may be clairvoyant) is quantitatively bounded. A new global EDF schedulability test is also proposed here that builds on the results of Bonifaci et al. This new test is shown to be less pessimistic and more widely applicable than the earlier result was, while retaining the strong theoretical properties of the earlier result.

Keywords. Global multiprocessor scheduling; sporadic tasks; Deadline Monotonic; EDF; processor speedup factor

*Supported in part by NSF Grant Nos. CNS-0834270, CNS-0834132, and CNS-0615197, ARO Grant No. W911NF-06-1-0425, and funding from IBM, Sun Microsystems, and the Intel Corporation.

†Supported in part by grants EU FET project FRONTS n.215270 and by MIUR FIRB project Italy-Israel RBIN047MH9.

‡Supported by the DFG-Research Center MATHEON.

1 Introduction

Real-time systems comprised of recurrent processes or tasks are often modeled using the *sporadic* tasks model. In this model, each sporadic task τ_i is characterized by three parameters — a worst-case execution time C_i, a relative deadline D_i, and a period T_i. Such a task generates a potentially infinite sequence of jobs. Successive jobs of τ_i arrive at least T_i time units apart, with each job having an execution requirement $\leq C_i$ and a deadline D_i time units after its arrival time.

We consider here real-time systems that can be modeled in this manner as collections of independent sporadic tasks, and that are implemented upon a platform comprised of several identical processors. We assume that the platform is fully *preemptive*, and that it allows for *global* inter-processor migration. (However, each job may execute on at most one processor at each instant in time.) We study the behavior of two well-known algorithms when scheduling systems of sporadic tasks upon such preemptive platforms: *Earliest Deadline First* [16, 20] and *Deadline-Monotonic* (DM) [19].

Schedulability tests for sporadic task systems. It is evident from the definition that a sporadic task system may generate infinitely many different collections of jobs during different executions. In hard-real-time systems, it must be guaranteed prior to system run time that all deadlines will be met. Such guarantees are made by *schedulability tests*. Let A denote a scheduling algorithm. A sporadic task system is said to be A-schedulable upon a specified platform if A meets all deadlines when scheduling each of the potentially infinite different collections of jobs that could be generated by the sporadic task system, upon the specified platform. An A-schedulability test accepts as input the specifications of a sporadic task system and a multiprocessor platform, and determines whether the task system is A-schedulable. An A-schedulability test is said to be *exact* if it identifies all A-schedulable systems, and *sufficient* if it identifies only some A-schedulable systems. (Of course, an A-schedulability test may never incorrectly mis-identify some system that is not A-schedulable, as being A-schedulable.) A sufficient schedulability test that is not exact is pessimistic, but for many situations an exact schedulability test is unknown or is computationally intractable. From an engineering point of view, a tractable schedulability test that is exact is ideal, while a tractable sufficient test with low pessimism is acceptable.

No exact schedulability tests are known, of any computational complexity, for the global EDF and DM scheduling of sporadic task systems (although an exact test of unacceptably high computational complexity, based on brute force state-space search, has been proposed [3] for EDF, for the special case when all task parameters are restricted to be integers). A number of sufficient schedulability tests have been proposed, including [1, 4, 8–10, 14, 18] – see [12] for a fairly comprehensive survey. It has been observed that none of these sufficient tests dominates all the others — there are schedulable systems deemed to be so by each test, which the other tests fail to identify as being schedulable. In the absence of such domination, extensive simulation experiments have been performed [5, 6, 12, 13] comparing the efficacy of the different tests in scheduling large collections of randomly-generated task systems. Although such simulations are very useful particularly in that they provide insight into the kinds of scenarios under which the different schedulability tests are superior to each other, the applicability of such simulation-based analysis is limited by the doubts raised regarding how realistic the random task-system generators are. In essence, it is difficult to generalize the validity of these results beyond the kinds of workloads that are modeled by the random task-system generator.

Processor speedup factor. Processor speedup factors represent a quantitative approach towards comparing different sufficient schedulability tests. A schedulability test is defined to have a processor speedup factor f, $f \geq 1$, if any task system not deemed to be schedulable by this test upon a particular platform is guaranteed to not be *feasible* – schedulable by an optimal clairvoyant scheduler – upon a platform in which each processor is at most $1/f$ times as fast. More formally,

Definition 1 (Processor speedup factor) *A schedulability test has a processor speedup factor f, $f \geq 1$, if it is guaranteed that any task system that is feasible upon a specified platform is deemed to be schedulable by the test upon a platform in which each processor is at least f times as fast.*

Intuitively, the processor speedup factor of a schedulability test quantifies both the pessimism of the test and the non-optimality of the scheduling algorithm. According to this metric, schedulability tests with smaller processor speedup factors are superior to ones with larger processor speedup factors, with a processor speedup factor equal to one implying both that the scheduling algorithm is optimal and that the test is exact.

It is reasonable to ask whether it makes sense to bundle both the non-optimality of the scheduling algorithm and the pessimism of the test into a single metric. From a pragmatic perspective, we believe that the answer is "yes" for our domain of interest, which is hard-real-time systems. Since it must be a priori guaranteed in such hard-real-time systems that all deadlines will be met during run-time, a scheduling algorithm is only as good as its associated schedulability test. In other words, a scheduling algorithm, no matter how close to optimal, cannot be used in the absence of an associated schedulability test able to guarantee that all deadlines will be met; what matters is that the *combination* of scheduling algorithm and schedulability test together have desirable properties.

This research. Some novel techniques were introduced in [15], for global EDF schedulability analysis of sporadic task systems. These techniques were applied to design an EDF schedulability test that is *optimal* from the perspective of processor speedup factor (this will be explained in greater detail in Section 2). In this paper, we generalize the analysis technique of [15] to render it applicable to algorithms other than EDF; in particular, so that it is applicable to DM [1]. Also, we present some pragmatic improvements that reduce some the pessimism of the [15] test for EDF, and indicate how such pragmatic improvements can be done for DM scheduling as well.

Organization. The remainder of this paper is organized as follows. In Section 2, we briefly review some related work that is most relevant to the research presented here, and that helps place our results in context. In Section 3, we present the task and system models used in this work, and describe techniques — *demand bound function* and its refinement the *maxmin* or the *forced-forward demand bound function* — that are used for quantifying the cumulative execution requirement of a task system upon a platform. In Section 4 we generalize the novel analysis technique from [15] to extend its applicability to both EDF and DM scheduling (in fact, to any work-conserving scheduling discipline), and thereby obtain necessary un-schedulability conditions for both EDF and DM. In Section 5 we use these unschedulability conditions to obtain processor speedup bounds for both EDF and DM (while the derivation for

[1] However unlike the results in [15] which are applicable for arbitrary sporadic task systems, we restrict our attention in this paper to systems in which each task's deadline is no larger than its period: $D_i \leq T_i$ for all tasks τ_i. Such systems are called *constrained sporadic task systems*.

EDF repeats the one from [15], the bound for DM is new). In Section 6 we present an algorithm implementing an EDF schedulability test that strictly dominates the one from [15], and indicate how this implementation can be modified to come up with a DM schedulability test as well.

2 Prior results

It has been shown [21] that *any collection of independent jobs that are feasible upon a platform comprised of m speed-$(\frac{m}{2m-1})$ processors is EDF-schedulable upon a platform comprised of m speed-1 processors.* It was also shown that this bound is tight: there are task systems feasible on m speed-$(\frac{m}{2m-1}+\epsilon)$ processors that are not EDF-schedulable on m speed-1 processors, for arbitrary small positive ϵ.

Although the result in [21] is very interesting and important from a theoretical perspective, it does not yield an EDF-schedulability test for sporadic task systems, since it does not tell us how to determine whether a given sporadic task system is feasible (upon the slower platform).

Recently [15], a new sufficient schedulability test for sporadic task systems was derived, which was shown to possess a processor speedup factor of $(2-\frac{1}{m})$ on an m-processor platform[2]. It follows from the result from [21] cited above that this EDF-schedulability test is in fact speedup-optimal. (By *speedup-optimal*, we mean that no EDF-schedulability test for sporadic task systems can possibly have a processor speedup factor less than this test's.)

With regards to sufficient schedulability tests for global DM scheduling[3], a test was presented in [17] for determining whether a given sporadic task system is global-DM schedulable upon a preemptive multiprocessor platform comprised comprised of m unit-capacity processors. It was shown that the processor speedup factor for this global-DM schedulability test is at most $(4-\frac{1}{m})$, when implemented upon m-processor platforms. This result was subsequently improved [7], and extended to sporadic task systems [11] which are not constrained (i.e., in which individual tasks' relative deadlines may exceed their periods). These improved results yield a processor speedup factor equal to

$$\frac{2(m-1)}{(4m-1)-\sqrt{12m^2-8m+1}}.$$

This approaches $2\sqrt{3}$ (which is ≈ 3.73) as $m \to \infty$.

The global EDF schedulability test of [15] is, despite its theoretical significance as a speedup-optimal test, of limited applicability in the analysis of actual real-time systems. First, it fails to identify as being EDF-schedulable any system τ for which $\max_{\tau_i \in \tau}\Big(C_i/D_i, C_i/T_i\Big)$ is larger than $m/(2m-1)$ (where m denotes the number of processors), and hence, for example, cannot be used to determine the EDF-schedulability of any task system containing even a single task with C_i/D_i or $C_i/T_i > 0.5$. Even for task systems τ satisfying $\max_{\tau_i \in \tau}\Big(C_i/D_i, C_i/T_i\Big) \leq m/(2m-1)$, we will see in Section 6 that the test of [15] is overly pessimistic. One of the contributions of this paper is to flesh out the details of the test

[2] [15] also presents a fully polynomial-time approximation scheme that trades off accuracy for computational efficiency: given any task system and an $\epsilon > 0$, it correctly decides, in polynomial time, that either the system is EDF-schedulable on m speed-$(2-1/m+\epsilon)$ processors, or it is infeasible on m speed-1 machines.

[3] We will focus on those tests for which quantitative performance bounds, in the form of processor speedup factors, have been determined.

in [15] and thereby derive a sufficient schedulability test of wider applicability and lower pessimism, while retaining the strong theoretical properties – in particular, the optimal processor speedup factor. This new schedulability test can be implemented to have a run-time that is pseudo-polynomial in the representation of the task system being analyzed, and is thus efficient enough to be used in practice.

3 Model and Notation

In the remainder of this paper, we will let τ denote a system of n sporadic tasks: $\tau = \{\tau_1, \tau_2, \ldots \tau_n\}$, with $\tau_i = (C_i, D_i, T_i)$ for all i, $1 \leq i \leq n$. 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: $D_i \leq T_i$ for all $\tau_i \in \tau$. *We restrict our attention here to constrained task systems.*

In order to devise schedulability tests, it is necessary to quantify the cumulative execution requirement that sequences of jobs may place on a computing platform. Given a sequence of jobs J and a specified time-interval $[t_a, t_f)$, the *demand* of J over $[t_a, t_f)$ is defined to be the sum of the execution requirements of all jobs in J with arrival times $\geq t_a$ and deadlines $\leq t_f$. The *demand bound function* $\text{DBF}(\tau, t)$ of a sporadic task system τ for an interval length t is then defined to be the largest demand by any legal sequence of jobs that may be generated by τ over any interval of length t. The *load* $\text{LOAD}(\tau)$ of a sporadic task system τ is defined as $\max_{t>o} \text{DBF}(\tau, t)/t$.

Baker and Cirinei [2] observed that some jobs arriving and/or having deadlines outside an interval could also contribute to the cumulative execution requirement placed on the computing platform within the interval. They introduced a notion that they call *minimum demand*. The minimum demand of a given collection of jobs in any specific time interval is the minimum amount of execution that the sequence of jobs could require within that interval in order to meet all its deadlines[4]. We illustrate the difference between *demand* and *minimum demand* by a simple example.

Example 1 Consider a sequence of jobs comprised of a single job that arrives at time-instant zero, has an execution requirement equal to 5, and a deadline at time-instant 10. The demand of this sequence of jobs over the time-interval $[0, t)$ is 0 for all $t < 10$, and 5 for all $t \geq 10$. The minimum demand of this sequence of jobs over the time-interval $[0, t)$ is equal to

- zero, for values of $t \leq 5$;
- $t - 5$, for $t \in (5, 10]$, since the sole job must execute for at least $t - 5$ units over the interval if it is to meet its deadline; and
- five, for $t > 10$.

■

The minimum demand concept is extended to sporadic tasks (and task systems) as follows. By definition, a sporadic task τ_i may generate infinitely many different collections of jobs. For a given interval-length t, τ_i's *maxmin demand* is defined to be the largest minimum demand over an interval of length t, by any collection of jobs that could legally be generated by τ_i. The maxmin demand of a sporadic task system τ for interval-length t is defined as the sum of the maxmin demands of the tasks in τ, each for an interval-length t. The *maxmin load* of τ is defined as the maximum value of the maxmin demand of τ for t, normalized by the interval length.

[4] An essentially identical concept was independently introduced in [15], and called the *necessary demand*; a related

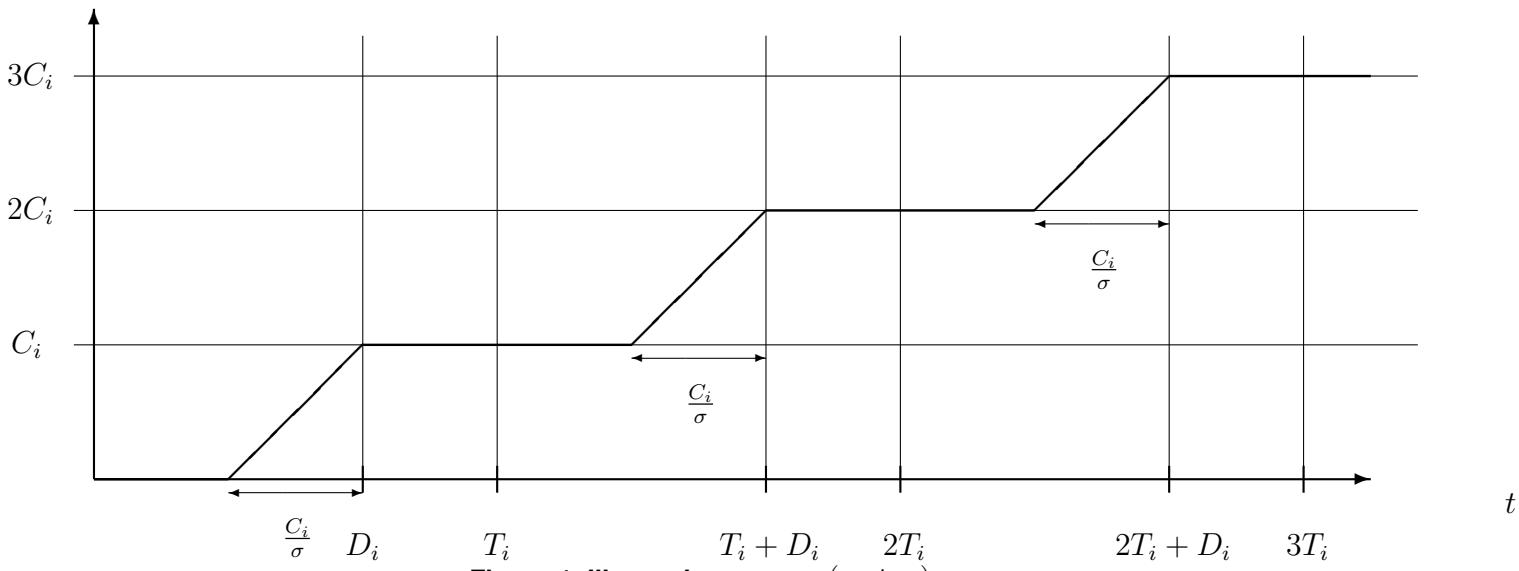

Figure 1. Illustrating FF-DBF(τ_i, t, σ)**.**

Forced-forward demand bound function. In [15], these concepts of maxmin demand and maxmin load were generalized to be applicable to execution on speed-σ processors, for arbitrary $\sigma > 0$. We now discuss a modified form of these ideas from [15].

Let τ_i denote a sporadic task, t any positive real number, and σ any positive real number ≤ 1. The *forced-forward demand bound function* FF-DBF(τ_i, t, σ) is defined as follows:

$$\text{FF-DBF}(\tau_i, t, \sigma) \stackrel{\text{def}}{=} q_i C_i + \begin{cases} C_i & \text{if } r_i \geq D_i \\ C_i - (D_i - r_i)\sigma & \text{if } D_i > r_i \geq D_i - \frac{C_i}{\sigma} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where

$$q_i \stackrel{\text{def}}{=} \left\lfloor \frac{t}{T_i} \right\rfloor \quad \text{and} \quad r_i \stackrel{\text{def}}{=} t \bmod T_i \ .$$

Informally speaking, FF-DBF(τ_i, t, σ) can be thought of as denoting the maxmin demand of τ_i for interval-length t, when execution *outside* the interval occurs on a speed-σ processor — see Figure 1. Similarly, it is easy to see that the "traditional" demand bound function is a special case of FF-DBF, in which it is assumed that execution outside the interval occurs on an infinite-speed processor:

$$\text{DBF}(\tau_i, t) = \text{FF-DBF}(\tau_i, t, \infty) \ .$$

Some additional notation: for a task system τ

$$\textit{Utilization } U(\tau) \stackrel{\text{def}}{=} \sum_{\tau_\ell \in \tau} C_\ell / T_\ell \ .$$

$$\textit{Maximum density } \delta_{\max}(\tau) \stackrel{\text{def}}{=} \max_{\tau_\ell \in \tau} C_\ell / D_\ell \ .$$

concept called *forced-forward demand* was also introduced.

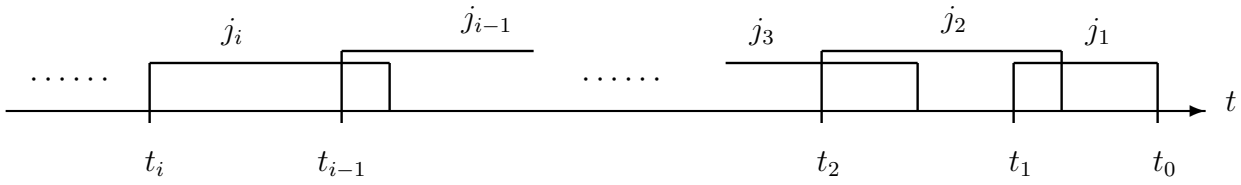

Figure 2. Notation used in Section 4.

$$\textit{Hyperperiod } P(\tau) \stackrel{\text{def}}{=} \mathsf{lcm}_{\tau_\ell \in \tau}\{T_\ell\} \ .$$

$$\text{FF-DBF}(\tau, t, \sigma) \stackrel{\text{def}}{=} \sum_{\tau_\ell \in \tau} \text{FF-DBF}(\tau_\ell, t, \sigma) \ . \tag{2}$$

$$\text{FF-LOAD}(\tau, \sigma) \stackrel{\text{def}}{=} \max_{t>0} \left(\frac{\text{FF-DBF}(\tau, t, \sigma)}{t} \right) \ . \tag{3}$$

4 Sufficient schedulability conditions for EDF and DM

Both EDF and DM are *work-conserving* algorithms: they never idle any processor while there are jobs awaiting execution. As we will see below, this work-conserving property implies that a deadline miss can only follow an interval during which a considerable amount of execution must have occurred. This property is formalized in the following discussion.

Let A denote some work-conserving algorithm that misses a deadline while scheduling some legal collection of jobs generated by τ. Let us now examine A's behavior on some minimal[5] legal collection of jobs of τ on which it misses a deadline. Let t_0 denote the instant at which the deadline miss occurs. Let j_1 denote a job that misses its deadline at t_0, and let t_1 denote j_1's arrival-time. Let s denote any constant satisfying $\delta_{\max}(\tau) \leq s \leq 1$. (Observe that, since $s \geq \delta_{\max}(\tau)$ and j_1 has not completed execution by t_0, it has executed for strictly less than $(t_0 - t_1) \times s$ units over the interval $[t_1, t_0)$.) We define a sequence of jobs j_i, time-instants t_i, and an index k, according to the following pseudo-code (also see Figure 2):

for $i \leftarrow 2, 3, \ldots$ **do**
  let j_i denote a job that
    – arrives at some time-instant $t_i < t_{i-1}$;
    – has a deadline after t_{i-1};
    – has not completed execution by t_{i-1}; and
    – has executed for strictly less than $(t_{i-1} - t_i) \times s$
      units over the interval $[t_i, t_{i-1})$.
  if there is no such job **then**
    $k \leftarrow (i-1)$
    break (out of the for loop)
end if
end for

[5]By *minimal* we mean that A is able to successfully schedule every proper subset of this collection of jobs.

Let W denote the amount of execution that occurs in this schedule over the interval $[t_k, t_0)$. Observation 1 derives a lower bound on W for any work-conserving algorithm A.

Observation 1

$$W > \big(m - (m-1)s\big) \times (t_0 - t_k) \; .$$

Proof: For each i, $1 \leq i \leq k$, let W_i denote the total amount of execution that occurs over the interval $[t_i, t_{i-1})$. Hence, $W = \sum_{i=1}^{k} W_i$.

Let x_i denote the total length of the time-intervals over $[t_i, t_{i-1})$ during which job j_i executes. By choice of job j_i, it is the case that

$$x_i < (t_{i-1} - t_i) \cdot s \; .$$

By choice of job j_i, it has not completed execution by time-instant t_{i-1}. Hence over $[t_i, t_{i-1})$, all m processors must be executing whenever j_i is not; it follows that

$$\begin{aligned} W_i &\geq m(t_{i-1} - t_i - x_i) + x_i \\ &= m(t_{i-1} - t_i) - (m-1)x_i \\ &> m(t_{i-1} - t_i) - (m-1)(t_{i-1} - t_i)s \\ &= (m - (m-1)s) \times (t_{i-1} - t_i) \; . \end{aligned}$$

The observation, follows, by summing $\sum_{i=1}^{k} W_i$. ■

Observation 1 above derived a lower bound on the amount of work W that is executed over $[t_k, t_0)$. In the following two observations, we will derive *upper* bounds on W when the scheduling algorithm is EDF and DM respectively; necessary conditions for non-schedulability under EDF and DM will follow, by requiring that the respective upper bounds be at least as large as the lower bound.

Observation 2 (EDF) *If the work-conserving algorithm A is EDF, then*

$$W \leq \text{FF-DBF}(\tau, (t_0 - t_k), s) \; .$$

Proof: Recall our assumption that we are analyzing a *minimal* unschedulable collection of jobs. If EDF is the scheduling algorithm, then such a minimal unschedulable collection will not contain any job that has its deadline $> t_0$ (since the presence of such jobs cannot effect the scheduling of jobs with deadline $\leq t_0$, the collection of jobs obtained by removing all such jobs is also unschedulable by EDF.) Thus all jobs that execute in $[t_k, t_0)$ (and thereby contribute to W) have their deadlines within the interval $[t_k, t_0)$. Some of them will also have arrived within this interval, while others may not.

Now it may be verified that the amount of execution that jobs of any task τ_ℓ contribute to W is bounded from above by the scenario in which a job of τ_ℓ has its deadline coincident with the end of the interval t_0, and prior jobs have arrived exactly T_ℓ time-units apart. Under this scenario, the jobs of τ_ℓ that may contribute to W include

- at least $q_\ell \stackrel{\text{def}}{=} \lfloor (t_0 - t_k)/T_\ell \rfloor$ jobs of τ_ℓ that lie entirely within the interval $[t_k, t_0)$; and
- (perhaps) an additional job that has its deadline at time-instant $t_k + r_\ell$, where $r_\ell \stackrel{\text{def}}{=} (t_0 - t_k) \bmod T_\ell$.

We now consider two separate cases:

1. $r_\ell \geq D_\ell$; i.e., the additional job with deadline at $t_k + r_\ell$ arrives at or after t_k. In this case, its contribution is C_ℓ.

2. $r_\ell < D_\ell$; i.e., the additional job with deadline at $t_k + r_\ell$ arrives prior to t_k. From the exit condition of the for-loop, it must be the case that this job has completed at least $(D_\ell - r_\ell) \times s$ units of execution prior to time-instant t_k; hence, its remaining execution is at most $\max(0, C_\ell - (D_\ell - r_\ell) \times s)$.

In either case, it may be seen that the upper bound on the total contribution of τ_ℓ to W is equal to FF-DBF$(\tau_\ell, t_0 - t_k, s)$ (see Equation 1). Summing over all ℓ, we conclude that the total contribution of all the tasks to W is bounded from above by $\sum_{\ell=1}^{n}(\text{FF-DBF}(\tau_\ell, t_0 - t_k, s))$, which is, by definition, FF-DBF(τ, L, s). ■

Observation 3 (DM) *If the work-conserving algorithm A is DM, then*

$$W \leq \text{FF-DBF}(\tau, 2(t_0 - t_k), s) \ .$$

Proof: Recall once again our assumption that we are analyzing a minimal unschedulable collection of jobs. If DM is the scheduling algorithm, then such a minimal unschedulable collection will not contain any job with *relative* deadline greater than the relative deadline of j_1, the job that misses its deadline at time-instant t_0 (since such jobs are assigned lower priority than j_1 under DM, and hence cannot effect the ability or otherwise of j_1 to meet its deadline.)

By definition of t_1, the relative deadline of j_1 is $(t_0 - t_1)$. For any job with relative deadline $< (t_0 - t_1)$ to contribute to W (and hence execute prior to t_0), it must have a deadline $\leq (t_0 + (t_0 - t_1))$, i.e., $2t_0 - t_1$.

As in the proof of Observation 2 above, it may be verified that the amount of execution that jobs of any task τ_ℓ contribute to W is bounded from above by the scenario in which a job of τ_ℓ has its deadline coincident with the end of the interval (i.e., at $2t_0 - t_1$), and prior jobs have arrived exactly T_ℓ time-units apart. Under this scenario, it follows by an argument essentially identical to the one used in the proof of Observation 2 that the total contribution of all the tasks to W is bounded from above by FF-DBF$(\tau, 2t_0 - t_1 - t_k, s)$. But since $t_k \leq t_1$, $(2t_0 - t_1 - t_k) \geq 2(t_0 - t_k)$, and the observation is proved. ■

Lemma 1 (EDF) *Suppose that constrained-deadline sporadic task system τ is not schedulable by global EDF upon m unit-speed processors. For each s, $s \geq \delta_{\max}(\tau)$, there is an interval-length L such that*

$$\text{FF-DBF}(\tau, L, s) > (m - (m-1)s)L \ .$$

Proof: Follows by chaining the lower bound on W of Observation 1 with the upper bound of Observation 2, with $L \leftarrow (t_0 - t_k)$. ■

Lemma 2 (DM) *Suppose that constrained-deadline sporadic task system τ is not schedulable by global DM upon m unit-speed processors. For each s, $s \geq \delta_{\max}(\tau)$, there is an interval-length L such that*

$$\text{FF-DBF}(\tau, L, s) > (m - (m-1)s)\frac{L}{2} \ .$$

Proof: Follows by chaining the lower bound on W of Observation 1 with the upper bound of Observation 3, with $L \leftarrow 2(t_0 - t_k)$. ■

5 Determining processor speedup factor

The contrapositive of Lemma 1 above represents a global EDF schedulability condition: any task system τ satisfying

$$\Big(\exists \sigma \ : \sigma \geq \delta_{\max}(\tau) \ : \Big(\forall\, \Delta \ : \Delta \geq 0 : \Big(\text{FF-DBF}(\tau, \Delta, \sigma) \leq (m - (m-1)\sigma) \times \Delta \Big)\Big)\Big) \tag{4}$$

is EDF-schedulable upon m unit-speed processors.

Similarly, the contrapositive of Lemma 2 represents a global DM schedulability condition: any task system τ satisfying

$$\Big(\exists \sigma \ : \sigma \geq \delta_{\max}(\tau) \ : \Big(\forall\, \Delta \ : \Delta \geq 0 : \Big(\text{FF-DBF}(\tau, \Delta, \sigma) \leq (m - (m-1)\sigma) \times \frac{\Delta}{2} \Big)\Big)\Big) \tag{5}$$

is DM-schedulable upon m unit-speed processors.

We now determine, in Theorems 1 and 2 below, the processor speedup factors of schedulability test for EDF and DM respectively, based on checking these conditions. While the result in Theorem 1 is a rederivation of the one in [15], the one in Theorem 2 is new and represents a significant improvement over the previously best-known result in [11].

In [2, Theorem 2], necessary conditions for global multiprocessor schedulability were identified; these necessary conditions generalize to our context and notation in the following manner:

Lemma 3 *If* FF-LOAD$(\tau, \sigma) > m\sigma$ *then* τ *is not feasible on* m *speed-*σ *processors.*

Proof: Suppose that FF-LOAD$(\tau, \sigma) > m\sigma$. Let t_0 denote a value for t that maximizes the RHS of Equation 3, and hence determines the value of FF-LOAD(τ, σ):

$$\text{FF-LOAD}(\tau, \sigma) = \Big(\sum_{\tau_\ell \in \tau} \text{FF-DBF}(\tau_\ell, t_0, \sigma)\Big)/t_0$$

By definition of FF-DBF, each task τ_ℓ can generate a sequence of jobs that together require $\geq$ FF-DBF(τ_ℓ, t_0, σ) units of execution over some interval of length t_0, when executing upon speed-σ processors. Since the different tasks of a sporadic task system are assumed to be independent of each other, such intervals for the different tasks can be aligned; the total execution requirement by all the tasks over the aligned interval is

$$\begin{aligned} &\geq \sum_{\tau_\ell \in \tau} \text{FF-DBF}(\tau_\ell, t_0, \sigma) \\ &= \text{FF-LOAD}(\tau, \sigma) \times t_0 \ \text{(By definition of } t_0) \\ &> m\sigma t_0 \ \text{(Since FF-LOAD}(\tau, \sigma) \text{ is assumed to be } > m\sigma) \end{aligned}$$

But $m\sigma t_0$ denotes the total computing capacity over an interval of size t_0 upon m speed-σ processors. We therefore conclude that the total execution requirement by all the tasks in τ over the interval cannot be met, and some deadline must necessarily be missed. ■

Lemma 4 *If task system* τ *does not satisfy Condition 4, then it is not feasible upon a platform comprised of* m *speed-*$\frac{m}{2m-1}$ *processors.*

Proof: First, any τ not satisfying Condition 4 that has $\delta_{\max}(\tau) > m/(2m-1)$ is trivially not feasible on speed-$\frac{m}{2m-1}$ processors.

Next, consider τ with $\delta_{\max}(\tau) \leq m/(2m-1)$. Suppose that τ does not satisfy Condition 4: for all values of $\sigma > \delta_{\max}(\tau)$, there is an interval-length $\Delta \geq 0$ such that FF-DBF$(\tau, \Delta, \sigma) > (m-(m-1)\sigma) \times \Delta$.

Let us instantiate this inequality for $\sigma \leftarrow m/(2m-1)$:

$$\begin{aligned}
\forall\, \Delta \geq 0 \ : \text{FF-DBF}(\tau, \Delta, \frac{m}{2m-1}) & \\
& > \left(m-(m-1)\frac{m}{2m-1}\right) \times \Delta \\
\Rightarrow \quad \text{FF-LOAD}(\tau, \frac{m}{2m-1}) & > m-(m-1)\frac{m}{2m-1} \\
\equiv \quad \text{FF-LOAD}(\tau, \frac{m}{2m-1}) & > \frac{2m^2-m-m^2+m}{2m-1} \\
\equiv \quad \text{FF-LOAD}(\tau, \frac{m}{2m-1}) & > m\frac{m}{2m-1}
\end{aligned}$$

It therefore follows, from Lemma 3, that τ is not feasible upon a platform comprised of m speed-$\frac{m}{2m-1}$ processors. ■

By taking the contrapositive of Lemma 4 above and observing that $1/(\frac{m}{2m-1})$ is equal to $(2-\frac{1}{m})$, we have

Theorem 1 *The processor speedup factor of an EDF schedulability test based on checking Condition 4 has a processor speedup factor of* $(2-\frac{1}{m})$ *on an* m*-processor platform.*

Reasoning very similar to that used in Lemma 4 and Theorem 1 above are now applied to DM scheduling:

Lemma 5 *If task system* τ *fails the DM schedulability test of Condition 5, then it is not feasible upon a platform comprised of* m *speed-*$\frac{m}{3m-1}$ *processors.*

Proof: Suppose that τ fails the schedulability test of Condition 5.

If $\delta_{\max}(\tau) > (m/(3m-1)$ then τ is trivially not feasible on a platform comprised of (any number of) speed-$(m/(3m-1)$ processors, and we are done.

Assume now that $\delta_{\max}(\tau) \leq (m/(3m-1)$. Suppose that τ does not satisfy Condition 5: for all values of $\sigma > \delta_{\max}(\tau)$, there is an interval-length $\Delta \geq 0$ such that FF-DBF$(\tau, \Delta, \sigma) > (m-(m-1)\sigma) \times \frac{\Delta}{2}$.

Let us instantiate this inequality for $\sigma \leftarrow m/(3m-1)$:

$$\begin{aligned}
\forall\, \Delta \geq 0 \ : \text{FF-DBF}(\tau, \Delta, \frac{m}{3m-1}) & \\
& > \left(m-(m-1)\frac{m}{3m-1}\right) \times \frac{\Delta}{2} \\
\Rightarrow \quad \text{FF-LOAD}(\tau, \frac{m}{3m-1}) & > (m-(m-1)\frac{m}{3m-1}) \times \frac{1}{2} \\
\equiv \quad \text{FF-LOAD}(\tau, \frac{m}{3m-1}) & > \frac{3m^2-m-m^2+m}{2(3m-1)} \\
\equiv \quad \text{FF-LOAD}(\tau, \frac{m}{3m-1}) & > m\frac{m}{3m-1}
\end{aligned}$$

It therefore follows from Lemma 3 above that τ is not feasible upon a platform comprised of m speed-$\frac{m}{3m-1}$ processors. ■

By taking the contrapositive of Lemma 4 above and observing that $1/(\frac{m}{3m-1})$ is equal to $(3 - \frac{1}{m})$, we have

Theorem 2 *The processor speedup factor of a DM schedulability test based on checking Condition 5 has a processor speedup factor of* $(3 - \frac{1}{m})$ *on an* m*-processor platform.*

6 Improved schedulability tests

Condition 4, the contrapositive of the EDF unschedulability condition of Lemma 1, asserts that in order to show a given task system τ EDF-schedulable, it suffices to demonstrate the existence of a $\sigma \geq \delta_{\max}(\tau)$ such that

$$\text{FF-DBF}(\tau, t, \sigma) \leq (m - (m-1)\sigma) \times t \tag{6}$$

for all values of $t \geq 0$. Let us refer to such a σ as a *witness* to the EDF-schedulability of τ. In order to obtain a schedulability test with the optimal processor speedup factor of $(2 - \frac{1}{m})$, we have seen (Theorem 1 above) that it suffices to only consider $\sigma \leftarrow m/(2m-1)$ as a potential witness, declaring the task set not schedulable if this value of σ fails to satisfy Inequality 6 for all $t \geq 0$. Indeed, this is in essence the schedulability test presented in [15]: determine whether $\sigma \leftarrow m/(2m-1)$ satisfies Inequality 6 for all $t \geq 0$.

However, by testing only one out of all the values of σ that could bear witness to a task system's schedulability, this test clearly fails to make full use of the insight into EDF-schedulability that Lemma 1 affords us. In the remainder of this section, we derive an algorithm that fully exploits the insight of Lemma 1, by correctly identifying all task systems for which *any* σ would cause Inequality 6 to evaluate to true for all $t \geq 0$. In other words, *the algorithm we derive in this section identifies all systems satisfying Condition 4*, whereas the test in [15] only identifies those systems that satisfy this condition instantiated with $\sigma \leftarrow m/(2m-1)$.

(A similar exercise can easily be conducted for the DM schedulability test derived from Condition 5, the contrapositive of the DM unschedulability condition of Lemma 2. The steps are essentially identical to the ones for EDF as described below; hence, we omit the details for DM schedulability analysis.)

Now there are infinitely many different values of σ that could potentially be witnesses to the EDF-schedulability of a task system; for each such potential witness, there are infinitely many values of t for which it must be validated that Inequality 6 is satisfied. Two questions must therefore now be answered:

Q1: What values of σ would we need to consider as potential witnesses to the EDF-schedulability of τ? and

Q2: In order to determine whether a particular σ is indeed a witness or not, for which values of t do we need to evaluate Condition 6?

We address the second of these questions first, in Section 6.1 below; the second one is addressed in Section 6.2.

6.1 Bounding the range of time-values that must be tested

We now address Q2, the second of the questions listed above: *for a given value of σ, for which values of t must we validate Condition 6 in order to be able to conclude that it holds for all t?*

Claim 1 *For a given σ and τ, if Condition 6 is violated for any t then it is violated for some t in*

$$\bigcup_{\tau_i \in \tau} \Big\{ kT_i + D_i, kT_i + D_i - \min(C_i/\sigma, D_i) \mid k \in \mathbb{N} \Big\} \tag{7}$$

Proof Sketch: This follows from the observation (also see Figure 1) that FF-DBF(τ_i, t, σ) increases with a constant slope between $kT_i + D_i - \min(C_i/\sigma, D_i)$ and $kT_i + D_i$, and remains unchanged elsewhere, for all $k \in \mathbb{N}$. Hence the LHS of Condition 6 increases with constant slope with increasing t between two consecutive t's in this set; since the RHS also increases with constant slope with increasing t, it is guaranteed that if this condition is violated at some $\tilde{t}$ it will be violated at one of the two t's in this set that neighbor $\tilde{t}$. ■

Claim 1 above tells us that we need evaluate Condition 6 for only countably many t's; Claims 2 and 3 below allow us to bound the actual number.

Claim 2 *If Condition 6 is violated at some t for a given τ and m, and a particular $\sigma \leq (m - U(\tau))/(m - 1)$, then it is violated at some t no larger than $P(\tau)$.*

Proof: Recall that $P(\tau)$ denotes the hyperperiod — the least common multiple of the task period parameters — of τ.

Since for all τ_i the period T_i divides the hyperperiod $P(\tau)$, it follows from Equation 1 (also see Figure 1) that

$$\begin{aligned} \text{FF-DBF}(\tau, t + P(\tau), \sigma) &= P(\tau) \times U(\tau) + \text{FF-DBF}(\tau_i, t, \sigma) \\ &\leq P(\tau) \times (m - (m-1)\sigma) + \text{FF-DBF}(\tau_i, t, \sigma) \end{aligned}$$

(since we are assuming that $\sigma \leq (m - (U(\tau))/(m - 1))$. Hence if Condition 6 is to be violated for some $t_v > P(\tau)$, it will also be violated for $t_v \bmod P(\tau)$. ■

Claim 2 tells us that $P(\tau)$ is an upper bound on the values of t for which Condition 6 needs to be evaluated. Claim 3 below provides another upper bound.

Claim 3 *If Condition 6 is violated at some t for a given τ and m, and a particular $\sigma \leq (m - U(\tau))/(m - 1)$, then t is no larger than*

$$\frac{\sum_{\tau_i \in \tau} C_i}{m - (m-1)\sigma - U(\tau)} \tag{8}$$

Proof: We first observe that it directly follows from the definition of FF-DBF (Equation 1 – also see Figure 1) that for all $t \geq D_i$ and for all σ,

$$\text{FF-DBF}(\tau_i, t, \sigma) \leq \left(\frac{t}{T_i} + 1 \right) C_i$$

Suppose that FF-DBF$(\tau, t, \sigma) > \big(m - (m-1)\sigma\big)t$ for some $t > \max_{\tau_i \in \tau}\{D_i\}$. We then have

$$\begin{aligned}
&\text{FF-DBF}(\tau, t, \sigma) > \big(m - (m-1)\sigma\big)t \\
\Rightarrow\ & \sum_{\tau_i \in \tau} \left(t\frac{C_i}{T_i} + C_i \right) > \big(m - (m-1)\sigma\big)t \\
\equiv\ & tU(\tau) + \sum_{\tau_i \in \tau} C_i > \big(m - (m-1)\sigma\big)t \\
\equiv\ & t < \frac{\sum_{\tau_i \in \tau} C_i}{m - (m-1)\sigma - U(\tau)}
\end{aligned}$$

and the lemma is proved. ■

Testing set. For a given σ and τ, let $\mathcal{TS}(\tau, \sigma)$ denote the *testing set* of values of t that lie in the set defined in Equation 7 and are no larger than both $P(\tau)$ and the bound defined by Equation 8.

How large can this testing set be? As shown in Claim 2, we need not consider any t exceeding the hyperperiod $P(\tau)$. It is easily seen that there are at most exponentially many points in the set defined in Equation 7 not exceeding $P(\tau)$; hence, the testing set contains at most exponentially many points.

Suppose, however, that we were to enforce an additional restriction that we would not consider any σ greater than

$$\Big(m - U(\tau) - \epsilon\Big)/(m-1) \tag{9}$$

where ϵ is an arbitrarily small positive constant. It would then follow from Inequality 8 that the upper bound on the values in $\mathcal{TS}(\tau, \sigma)$ is guaranteed to be $\leq \big(\sum_{\tau_i \in \tau} C_i\big)/\epsilon$, which is pseudo-polynomial in the representation of the task system τ. Thus, this restriction immediately yields a pseudo-polynomial upper bound on the number of elements in $\mathcal{TS}(\tau, \sigma)$.

The consequence of enforcing the restriction of Equation 9 above is that the test we develop is no longer able to identify all task systems satisfying Condition 4: task systems that only satisfy Condition 4 for values of σ in $((m - U(\tau))/(m-1) - \epsilon, (m - U(\tau))/(m-1)]$ would not be identified by our test. In exchange for this slight loss of optimality (the degree of which can be controlled by choosing ϵ to be appropriately small), we would restrict the size of the testing set to be pseudo-polynomial.

6.2 Choosing potential witnesses to test

In this section, we address Q1, the first of the two questions listed earlier in this section. That is, we set about restricting the candidate field of σ's that need be tested as potential witnesses to the EDF-schedulability of τ.

Claim 4 *No value of σ that is greater than $(m - U(\tau))/(m-1)$ can possibly result in Condition 6 evaluating to true for all values of t (and hence, no such value of σ can attest to the EDF-schedulability of τ).*

Proof: Observe that FF-DBF(τ_i, t, σ) asymptotically approaches $t \times (C_i/T_i)$ as $t \to \infty$. Hence FF-DBF(τ, t, σ) asymptotically approaches $t \times U(\tau)$ with increasing t. In order to have FF-DBF$(\tau, t, \sigma) \leq$

$(m-(m-1)\sigma)t$ for all t, therefore, we need

$$\begin{aligned} U(\tau) &\leq m-(m-1)\sigma \\ \equiv \quad \sigma &\leq \frac{m-U(\tau)}{m-1}\,. \end{aligned}$$

■

As a consequence of Claim 4 above, we can restrict the range of values for σ that are potential witnesses to the EDF-schedulability of τ. However, there are still infinitely many distinct values in this range, and we clearly cannot exhaustively check all these infinitely many values. Fortunately it so happens that we can restrict the actual number of values of σ within this range that need be considered as potential witnesses to the EDF-schedulability of τ, as we will now show.

Let us suppose that we have identified a particular σ_{cur}, such that we know that no $\sigma < \sigma_{\text{cur}}$ can possibility bear witness to the EDF-schedulability of τ. Suppose that we then test σ_{cur}, and determine that it is not a witness to the EDF-schedulability of τ, either — t_{cur} is a value of t that causes Condition 6 to evaluate to false when $\sigma \leftarrow \sigma_{\text{cur}}$. Let σ_{new} denote the smallest value of $\sigma > \sigma_{\text{cur}}$ such that Condition 6 evaluates to true with $(\sigma \leftarrow \sigma_{\text{new}}; t \leftarrow t_{\text{cur}})$. (We describe below, in Section 6.4, how the value of σ_{new} is computed.) It is clear that t_{cur} rules out the possibility of any $\sigma \in [\sigma_{\text{cur}}, \sigma_{\text{new}})$ bearing witness to the EDF-schedulability of τ; hence, the *next* value of σ that we will need to consider is σ_{new}.

So we've seen how, if we know a constant σ_{cur} such that no $\sigma \leq \sigma_{\text{cur}}$ can be a witness to the EDF-schedulability of τ, we can determine the next potential witness σ_{new} that we must consider. Claim 5 below tells us that in considering σ_{new}, we need not revisit values of $\mathcal{TS}(\tau, \sigma_{\text{new}})$ that are $\leq t_{\text{cur}}$:

Claim 5 *Suppose that τ is not EDF-schedulable. Consider some s_1 and t_1 such that*

$$\text{FF-DBF}(\tau, t_1, s_1) > (m-(m-1)s_1) \times t_1.$$

For any $s_2 > s_1$, there is a $t_2 \geq t_1$ such that

$$\text{FF-DBF}(\tau, t_2, s_2) > (m-(m-1)s_2) \times t_2.$$

Proof: This follows from the observation that the jobs $j_1, j_2, \ldots$ that are defined according to the pseudo-code given in Section 4 for a given value of s (say, $s \leftarrow s_1$), are also valid for larger values of s (say, $s \leftarrow s_2$). This is easily shown by induction. Assume that $j_1, \ldots, j_{i-1}$ as defined for $s \leftarrow s_1$ are valid for $s \leftarrow s_2$: this implies that t_{i-1} is the same when $s \leftarrow s_1$ and $s \leftarrow s_2$. The job j_i as defined for $s \leftarrow s_1$ has executed for less than $(t_{i-1} - t_i)s_1$ prior to t_{i-1}. But since $s_2 > s_1$, it has also executed for less than $(t_{i-1} - t_i)s_2$ prior to t_{i-1}, and hence satisfies the condition to be considered as job j_i for $s \leftarrow s_2$ as well. ■

6.3 Putting the pieces together: the EDF schedulability test

We are now ready to put the pieces together, and specify our schedulability test. This schedulability test is a methodical quest for a value of σ for which there is no t causing Condition 6 to evaluate to false (and which is thus a witness to the EDF-schedulability of τ). Based on Claim 5 above, we will start out testing a small value for σ; if this fails, we can try a larger value for σ and use the result of Claim 5 to trim the set of potential values of t that need to be tested for this larger value of σ. In greater detail, our algorithm is the following.

S1 Let $\sigma_{\rm cur}$ denote the value of σ currently being evaluated (i.e., the potential witness currently under consideration). This is initialized as follows: $\sigma_{\rm cur} \leftarrow \delta_{\max}(\tau)$.

We will also use an additional variable $t_{\rm cur}$, initialized to zero: $t_{\rm cur} \leftarrow 0$.

S2 If $\sigma_{\rm cur}$ is larger than $\left(\frac{m-U(\tau)}{m-1} - \epsilon\right)$ where ϵ is an arbitrarily small positive constant that has been a priori determined, then we ***exit the test, having failed to show τ is EDF-schedulable***. (Here, we are using the result shown in Claim 4, modified as discussed in Equation 9 to yield a testing set of pseudo-polynomial size, to restrict the range of values of σ that we need test as potential witnesses to the EDF-schedulability of τ.)

Otherwise by the results of Section 6.1, we need only evaluate Condition 6 for values of $t \in \mathcal{TS}(\tau, \sigma_{\rm cur})$ to determine whether it is satisfiable or not. We begin at the smallest value in $\mathcal{TS}(\tau, \sigma_{\rm cur})$ that is greater than $t_{\rm cur}$, and consider the values in $\mathcal{TS}(\tau, \sigma_{\rm cur})$ in increasing order. If no value of t in $\mathcal{TS}(\tau, \sigma_{\rm cur})$ causes Condition 6 to evaluate to false for this current value of $\sigma_{\rm cur}$, then we ***exit the test, having succeeded in showing that τ is EDF-schedulable***.

S3 Suppose, however, that there *is* some value of t that causes Condition 6 to evaluate to false for this value of $\sigma_{\rm cur}$. Assign $t_{\rm cur}$ this value of t. By Claim 5, if τ is not EDF-schedulable then for all values of $\sigma > \sigma_{\rm cur}$ there is some $t \geq t_{\rm cur}$ which causes Condition 6 to evaluate to false. Let $\sigma_{\rm new}$ denote the smallest value of $\sigma' > \sigma_{\rm cur}$, such that

$$\text{FF-DBF}(\tau, t_{\rm cur}, \sigma') \leq (m - (m-1)\sigma') \times t_{\rm cur} \ .$$

We compute $\sigma_{\rm new}$ using the technique described in Section 6.4 below, assign $\sigma_{\rm cur}$ this value $\sigma_{\rm new}$, and goto Step S2.

Computational complexity. Observe that the values assigned to the variable $t_{\rm cur}$ during the above algorithm are monotonically increasing — once we assign $t_{\rm cur}$ a particular value, we never assign it a smaller value even after we have changed the value assigned to $\sigma_{\rm cur}$. This observation can be used to show that the total number of values assigned to $t_{\rm cur}$ is no more than the cardinality of $\mathcal{TS}(\tau, \sigma)$, for the largest value of σ that is tested. And we have seen in Section 6.1 that this number is pseudo-polynomially bounded in the representation of the task system τ. We will see in Section 6.4 below that $\sigma_{\rm new}$ can be computed in polynomial time, while the rest of the processing above for a given value of $t_{\rm cur}$ is easily seen to also take polynomial time. This yields the following result:

Theorem 3 *This EDF-schedulability test has pseudo-polynomial time complexity.*

6.4 Computing $\sigma_{\rm new}$

Given *fixed* values for $t_{\rm cur}$ and $\sigma_{\rm cur}$ such that

$$\text{FF-DBF}(\tau, t_{\rm cur}, \sigma_{\rm cur}) > (m - (m-1)\sigma_{\rm cur}) \times t_{\rm cur} \ ,$$

our objective is to compute $\sigma_{\rm new}$, the smallest $\sigma' > \sigma_{\rm cur}$ such that

$$\text{FF-DBF}(\tau, t_{\rm cur}, \sigma') \leq (m - (m-1)\sigma') \times t_{\rm cur} \ .$$

Let us examine how FF-DBF$(\tau_i, t_{\rm cur}, \sigma)$ changes as σ is increased in the neighborhood of $\sigma_{\rm cur}$, while the task τ_i and the interval-length $t_{\rm cur}$ are kept unchanged. From Equation 1, we know that FF-DBF$(\tau_i, t_{\rm cur}, \sigma)$ depends on q_i and r_i, where $q_i = \lfloor t_{\rm cur}/T_i \rfloor$ and $r_i = t_{\rm cur} \bmod T_i$. Notice that the values of q_i and r_i do not depend on σ. Hence,

(a) FF-DBF$(\tau_i, t_{\rm cur}, \sigma)$ does not vary with σ if $r_i \geq D_i$ (the first case in Equation 1).

(b) It varies linearly with σ while $D_i > r_i \geq D_i - \frac{C_i}{\sigma}$. That is, if $r_i < D_i$ then FF-DBF$(\tau_i, t_{\rm cur}, \sigma)$ decreases linearly with increasing σ while $\sigma \leq C_i/(D_i - r_i)$. This is the second case in Equation 1.

(c) Once σ increases such that it is $> C_i/(D_i - r_i)$, FF-DBF$(\tau_i, t_{\rm cur}, \sigma)$ remains unchanged with further increase in the value of σ. This is the third case in Equation 1.

In order to compute $\sigma_{\rm new}$ given values for $t_{\rm cur}$ and $\sigma_{\rm cur}$, we would therefore

L1 Classify each task τ_i as being either in class ***(a)***, ***(b)***, or ***(c)*** according to the above classification. That is, a task τ_i for which $r_i \geq D_i$ would be classified as being in class ***(a)***; one with $r_i < D_i$ and $r_i \geq D_i - \frac{C_i}{\sigma_{\rm cur}}$ would be classified as being in class ***(b)***; while one with $r_i < D_i - \frac{C_i}{\sigma_{\rm cur}}$ would be classified as being in class ***(c)***.

For each task τ_i in class ***(b)***, let $\hat{\sigma}_i \stackrel{\text{def}}{=} C_i/(D_i - r_i)$. Increasing σ to become greater than $\hat{\sigma}_i$ would cause τ_i to no longer be a class ***(b)*** task.

L2 Observing that only tasks in class ***(b)*** have their FF-DBF's change –linearly– with changing σ, we can set up and solve a linear equation to determine σ_o, the smallest $\sigma' > \sigma_{\rm cur}$ for which

$$\text{FF-DBF}(\tau, t_{\rm cur}, \sigma_o) = (m - (m-1)\sigma_o) \times t_{\rm cur} \ .$$

L3 If this computed value of σ_o is $\leq \hat{\sigma}_i$ for all tasks τ_i in class ***(b)***, then we have computed the desired value for $\sigma_{\rm new}$. Else,

(a) assign $\sigma_{\rm cur}$ the value of the smallest $\hat{\sigma}_i$ from among all those computed for tasks in class ***(b)***, and

(b) repeat from step L1 above.

Computational complexity. It is not difficult to see that $\sigma_{\rm new}$ can be computed in time polynomial in the representation of τ. This follows from the observations that

- Steps L1, L2, and L3 above each take polynomial time.
- During each iteration of the 3-step process L1-L3 either (i) we determine the value of $\sigma_{\rm new}$ and exit; or (ii) at least one task that was in class ***(b)*** will henceforth be placed in class ***(c)*** in the subsequent iteration, whereas no additional tasks become class ***(b)*** tasks. Thus, the number of iterations of L1-L3 is bounded from above by the number of tasks initially in class ***(b)***, which is, of course, itself bounded by the number of tasks in τ.

7 Summary and conclusions

Recently, Bonifaci et al. have obtained [15] a speedup-optimal global EDF schedulability test for sporadic task systems implemented upon identical multiprocessor platforms. In this paper, we have generalized the technique so that it is applicable to the analysis of any work-conserving global scheduling algorithm. We have used this generalization to come up with a schedulability test for global DM which has a speedup factor superior to any previously known.

Although the speedup optimality property makes the EDF schedulability condition in [15] very significant from a conceptual perspective, its applicability in the analysis of actual real-time systems is somewhat limited. We have built upon the theoretical foundations provided by the [15] test, to obtain a sufficient EDF schedulability test of wider applicability and lower pessimism that retains the optimal processor speedup factor. We have shown that this schedulability test can be implemented to have a run-time that is pseudo-polynomial in the representation of the task system being analyzed. Since many algorithms that are currently used in the analysis of real-time systems are of similar computational complexity, this suggests that our schedulability test is efficient enough to be of use in the design and analysis of actual multiprocessor real-time application systems.

References

[1] BAKER, T. Multiprocessor EDF and deadline monotonic schedulability analysis. In *Proceedings of the IEEE Real-Time Systems Symposium* (December 2003), IEEE Computer Society Press, pp. 120–129.

[2] BAKER, T., AND CIRINEI, M. A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks. In *Proceedings of the IEEE Real-time Systems Symposium* (Rio de Janeiro, December 2006), IEEE Computer Society Press, pp. 178–187.

[3] BAKER, T., AND CIRINEI, M. Brute-force determination of multiprocessor schedulability for sets of sporadic hard-deadline tasks. In *Proceedings of the 10th International Conference on Principles of Distributed Systems* (Guadeloupe, December 2007), pp. 62–75.

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

[5] BAKER, T. P. Comparison of empirical success rates of global vs. partitioned fixed-priority and EDF scheduling for hard real time. Tech. Rep. TR-050601, Department of Computer Science, Florida State University, 2005.

[6] BAKER, T. P. A comparison of global and partitioned EDF schedulability tests for multiprocessors. In *Proceeding of the International Conference on Real-Time and Network Systems* (Poitiers, France, 2006).

[7] BARUAH, S. Schedulability analysis of global deadline-monotonic scheduling. Available at `http://www.cs.unc.edu/~ baruah`, 2007.

[8] BARUAH, S. Techniques for multiprocessor global schedulability analysis. In *Proceedings of the Real-Time Systems Symposium* (Tucson, AZ, December 2007), IEEE Computer Society Press, pp. 119–128.

[9] BARUAH, S., AND BAKER, T. Global EDF schedulability analysis of arbitrary sporadic task systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems* (Prague, Czech Republic, July 2008), IEEE Computer Society Press.

[10] BARUAH, S., AND BAKER, T. Schedulability analysis of global EDF. *Real- Time Systems 38*, 3 (2008), 223–235.

[11] BARUAH, S., AND FISHER, N. Global deadline-monotonic scheduling of arbitrary-deadline sporadic task systems. In *Proceedings of the 11th International Conference on Principles of Distributed Systems* (Guadeloupe, French West Indies, December 2007), Springer-Verlag.

[12] BERTOGNA, M. *Real-Time Scheduling Analysis for Multiprocessor Platforms*. PhD thesis, Scuola Superiore Santa Anna, Pisa, Italy, 2008.

[13] BERTOGNA, M., AND CIRINEI, M. Response-time analysis for globally scheduled symmetric multiprocssor platforms. In *Proceedings of the Real-Time Systems Symposium* (Tucson, AZ, December 2007), IEEE Computer Society Press, pp. 149–158.

[14] BERTOGNA, M., CIRINEI, M., AND LIPARI, G. 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, pp. 209–218.

[15] BONIFACI, V., MARCHETTI-SPACCAMELA, A., AND STILLER, S. A constant-approximate feasibility test for multiprocessor real-time scheduling. In *Proceedings of the 16th Annual European Symposium on Algorithms* (Karlsruhe, Germany, 2008), pp. 210–221.

[16] DERTOUZOS, M. Control robotics : the procedural control of physical processors. In *Proceedings of the IFIP Congress* (1974), pp. 807–813.

[17] FISHER, N., AND BARUAH, S. Global static-priority scheduling of sporadic task systems on multiprocessor platforms. In *Proceeding of the IASTED International Conference on Parallel and Distributed Computing and Systems* (Dallas, TX, November 2006), IASTED.

[18] LEONTYEV, H., AND ANDERSON, J. A unified hard/soft real-time schedulability test for global EDF multiprocessor scheduling. In *Proceedings of the Real-Time Systems Symposium* (Barcelona, December 2008), IEEE Computer Society Press.

[19] LEUNG, J., AND WHITEHEAD, J. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation 2* (1982), 237–250.

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

[21] PHILLIPS, C. A., STEIN, C., TORNG, E., AND WEIN, J. 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.