Journal of Systems Architecture 57 (2011) 487–497

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Tests for global EDF schedulability analysis

Marko Bertogna [a], Sanjoy Baruah [b,*]

[a] Scuola superiore Sant' Anna, Pisa, Italy
[b] University of North Carolina Chapel Hill, NC, USA

ARTICLE INFO

Article history:
Received 26 January 2010
Received in revised form 6 August 2010
Accepted 8 September 2010
Available online 18 September 2010

Keywords:
Multiprocessor scheduling
Earliest Deadline First
Global scheduling
Schedulability analysis

ABSTRACT

Several schedulability tests have been proposed for global EDF scheduling on identical multiprocessors. All these tests are sufficient, rather than exact. These different tests were, for the most part, independently developed. The relationships among such tests have not been adequately investigated, so that it is difficult to understand which test is most appropriate in a particular given scenario. This paper represents an attempt to remedy this, by means of three major contributions. First, we summarize the main existing results for the schedulability analysis of multiprocessor systems scheduled with global EDF, showing, when possible, existing dominance relations. We compare these algorithms taking into consideration different aspects, namely, run-time complexity, average performances over randomly generated workloads, sustainability properties and speedup factors. Second, based on this comparative evaluation we propose a recommended approach to schedulability analysis, that suggests a particular order in which to apply preexisting tests, thereby accomplishing both good provable performance and good behavior in practice. And finally, we propose a further improvement to one of these preexisting tests to improve its run-time performance by an order of magnitude, while completely retaining its ability to correctly identify schedulable systems.

1. Introduction

Scheduling and schedulability analysis of multiprocessor systems with migration support has recently been receiving increasing attention in the real-time research community. Recent advancements in multiprocessor and multicore technology are reducing the migration-related penalties of global scheduling algorithms, rendering more likely the adoption of such kind of schedulers in actual systems.

From a real-time system design perspective, it is not only important to have a test that is able to detect the schedulability of a given task set, but also to do that in a reasonable amount of time. In this paper we compare, both theoretically and experimentally, the performances of the main existing schedulability tests for sporadic task systems with hard real-time requirements, scheduled with global Earliest Deadline First (EDF) on an identical multiprocessor platform. In particular, we report on an exhaustive set of simulations which analyze which test is able to detect the larger number of schedulable task sets, for different randomly generated distributions of task set parameters. We identify dominance and incomparability relationships among the different tests. (A schedulability test *dominates* another if all task systems identified as being schedulable by one are also so identified by the other; two tests are *incomparable* if there are task systems identified as schedulable by each that the other fails to identify as being so.)

Based on this comparison, we present observations regarding the practical application of the considered tests. We propose a specific order in which a subset of the tests – those that dominate all other considered tests – be applied, such that the likelihood of identifying schedulable systems with relatively small computational effort is increased, while ensuring (i) that the entire suite of tests is together able to identify all schedulable task systems that are identified by any considered test; and (ii) the theoretical performance of the suite of tests, as quantified by its processor speedup factor, is the best possible.

This recommended suite of tests has, as its final test (i.e., the one to be applied if none of the prior tests is able to determine that the system being tested is EDF schedulable), a recently-proposed test that is based on the notion of *forced-forward demand bound function (FFDBF)* [10]. The inclusion of this test is necessary in order to retain the optimality property (on the processor speedup factor); however, prior implementations of this test have tended to be relatively inefficient. We propose here a means of speeding up the run-time of this test by incorporating ideas from the *Quick convergence Processor-demand Analysis (QPA)* test, proposed by Zhang and Burns [25] in the context of *uniprocessor* EDF schedulability analysis.

Organization: The remainder of this paper is organized as follows. In Section 2, we briefly describe the task and machine

* Corresponding author.
E-mail address: baruah@cs.unc.edu (S. Baruah).

1383-7621/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2010.09.004

model that we use, and provide notation and additional definitions that will be used in the remainder of the paper. We also discuss the notions of predictability, sustainability, and processor speedup factors, which are used as metrics for comparing the effectiveness of different suficient EDF schedulability tests. In Section 3, we describe (Sections 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7) the previously-proposed global EDF schedulability tests that we will be comparing in this research, and (in Section 3.8) the new and improved one that we propose as an improvement over prior tests. We also present the salient features of each test. In Section 4 we draw some informal conclusions from our survey of all these tests, and exploit these conclusions to propose a combined suite of tests that, we believe, provides the best features of these tests in a practically applicable manner. In Section 5 we describe the array of experiments that we have conducted in order to test the different schedulability tests, present the data generated by some of these tests, and draw more concrete conclusions about their relative effectiveness.

2. System model

We consider a set τ of n sporadic tasks [12] to be scheduled on m identical processors using global EDF. Each task $\tau_k \in \tau$ is characterized by a three-tuple (C_k, D_k, T_k) composed by a *worst-case computation time* C_k, a *relative deadline* $D_k \geqslant C_k$, and a *period*, or *minimum inter-arrival time*, T_k. In this paper, we will consider only *constrained deadline systems*, for which $D_i \leqslant T_i, \ \forall \tau_i \in \tau$.

A task τ_k is composed by a sequence of jobs J_k^j, where each job is characterized by an arrival time r_k^j, a finishing time f_k^j and an absolute deadline d_k^j. We say that a job is ready at time t, if $t \in \left[r_k^j, f_k^j\right)$. By extension, a task is ready (or backlogged) whenever it has a ready job.

With global EDF, each task ready to execute is placed in a system-wide queue, ordered by non-decreasing absolute deadline, from which the first m tasks are extracted to execute on the available processors.

The tasks are assumed to be *independent* of each other, meaning that no structure is shared, except for the computing units, and no data is simultaneously requested by more than one task. There are therefore no blocking effects caused by contemporary accesses to serially usable resources.

The *utilization* (resp. *density*) of a task τ_k is defined as $U_k = \frac{C_k}{T_k}$ (resp. $\lambda_k = \frac{C_k}{D_k}$), while the *total utilization* (resp. *total density*) is $U_{\text{tot}} = \sum_{\tau_i \in \tau} U_i$ (resp. $\lambda_{\text{tot}} = \sum_{\tau_i \in \tau} \lambda_i$). Let $U_{\max}$ (resp. $\lambda_{\max}$) be the largest utilization (resp. the largest density) among all tasks. The *response time* R_k of τ_k is the worst-case finishing time among all jobs of τ_k: $R_k \doteq \max_{J_k^j \in \tau_k}(f_k^j - r_k^j)$. The *minimum slack* S_k of τ_k is the minimum distance between the absolute deadline and the finishing time of any job of τ_k. It is therefore $S_k \doteq \min_{J_k^j \in \tau_k}(d_k^j - f_k^j) = D_k - R_k$. Note that when a task set is schedulable, each task has a non-negative slack and a response time lower than or equal to the deadline.

The *demand bound function* of a task τ_k is defined as

$$dbf_i(t) = \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1\right) C_i.$$

In the following, we will use $(x)_0$ as a short notation for $\max(0, x)$.

2.1. Predictability and Sustainability

A system that becomes unschedulable when less stringent timing parameters are used is not sufficiently robust for critical applications. To capture this concept, Baruah and Burns introduced in [11] the concept of *sustainability*. We report here the definition of sustainability with relation to the sporadic task model adopted in this paper.

Definition 1. *(Sustainability) A scheduling algorithm $\mathfrak{A}$ is sustainable if and only if the $\mathfrak{A}$-schedulability of a sporadic task system implies the $\mathfrak{A}$-schedulability of the same task system modified in any of the following ways: (i) decreasing execution requirements; (ii) increasing periods or inter-arrival times; (iii) increasing relative deadlines.*

For the sporadic task model, Baker and Baruah showed in [1] that EDF is sustainable with respect to decreased execution times and later arrivals. Thanks to this result, all EDF-schedulability tests for sporadic task systems are also valid for strictly periodic task systems. However, sustainability w.r.t. deadline relaxations has not previously been studied.

The concept of sustainability has also been extended to schedulability tests. A test is sustainable if, given a task set passing the test, another task set with less stringent timing requirements is guaranteed to also pass the test. It is possible to build sustainable sufficient schedulability tests even for task systems scheduled with a non-sustainable algorithm.

2.2. Processor speedup factor

A metric that can be used to characterize the performance of a schedulability test is the *processor speedup factor*. This is a lower bound on the speedup factor s that guarantees that each feasible task set on a platform composed by identical processors will pass the considered schedulability test on a platform in which each processor is s times as fast. In other words, this means that if the schedulability tests fails, the task set cannot be scheduled with any algorithm on a platform which is at most $1/s$ times as fast.

Phillips *et al.* proved in [24] the following (tight) resource augmentation bound for collections of independent jobs that are scheduled with EDF.

Theorem 1 (*from [24]*). *Any collection of independent jobs that is feasible upon m processors of a given speed, is schedulable with global* EDF *upon m processors each of which is $(2 - \frac{1}{m})$ times as fast. There are collections of independent jobs feasible on m processors of a given speed that are not* EDF*-schedulable on a platform of m identical processors each of speed $< (2 - \frac{1}{m})$ times as fast.*

Although Theorem 1 does not generalize from collections of independent jobs to sporadic task systems, it does provide a *lower bound* on processor speedup factor for EDF schedulability tests for sporadic task systems. Specifically, Theorem 1 implies that no schedulability test for global EDF of sporadic task systems may have a processor speedup factor less than $(2 - \frac{1}{m})$; the amount by which the processor speedup factor of a particular EDF-schedulability test exceeds this tight bound of $(2 - \frac{1}{m})$ may be used as a measure of the optimality of the test.

3. Schedulability tests for global EDF

We now survey the main existing results in the schedulability analysis of sporadic task systems scheduled with global EDF. We omit existing works that are generalized by the results described below.

Cucu and Goossens showed in [19] that to check the EDF-schedulability of a synchronous periodic task set, it is sufficient to verify if any deadline is missed in the generated schedule until the hyperperiod, i.e., the least common multiple of all tasks periods. However, since the synchronous periodic case does not characterize worst-case behavior for sporadic task systems, the above result cannot be used for the systems considered in this paper.

We hereafter recall the main existing sufficient schedulability tests for multiprocessor systems scheduled with global EDF. Each test will be denoted with an acronym, mostly derived taking the first letter of the name of each author.

3.1. GFB

In [23], the resource augmentation result of Theorem 1 has been used by Goossens *et al.* as a basis to prove a utilization-based schedulability test for implicit deadline sporadic task systems scheduled with EDF. We report here the straightforward generalization for constrained deadline systems.

Theorem 2 (*GFB*). *A task set τ is schedulable with global* EDF *if*

$$\lambda_{tot} \leqslant m(1-\lambda_{max})+\lambda_{max}. \quad (1)$$

The test is sustainable w.r.t. all considered relaxations.

3.2. BAK

A different approach has been proposed by Baker in [2,3], analyzing the workload that must be executed in a particular window to cause a deadline miss. For a job J_k^j of task τ_k to miss its deadline d_k^j, it is necessary that whenever τ_k does not execute, all m processor are busy executing other tasks (as in Fig. 1). Considering that τ_k does not complete before its deadline, the total workload inside interval $\left[r_k^j, d_k^j\right)$ should be $> m(D_k - C_k) + C_k$.

To derive a schedulability condition, the above lower bound on the workload is compared with the worst-case workload produced by the jobs executing in the same window $\left[r_k^j, d_k^j\right)$. An upper bound on the workload of such jobs can be found considering all tasks in a strictly periodic release sequence and with a common deadline at d_k^j, as in Fig. 2. However, computing a tight upper bound is not straightforward due to the unknown contributions of the *carry-in jobs*, i.e., instances arriving before t_o but with a deadline in $[t_o, t_d)$. For this reason, Baker proposes to enlarge the considered window with the *maximal μ-busy interval*, which is defined as follows (see Fig. 3). Consider a task τ_k that misses a deadline at time-instant t_d. Let t_o be an earlier time-instant such that the total amount of workload executed by all tasks in $[t_o, t_d)$ is $\geqslant \mu^*(t_d - t_o)$. The maximal μ-busy interval is the largest interval $[t_o, t_d)$ with the above property. Taking $\mu \leqslant m - (m-1)\lambda_k$, it is possible to prove that there exist a unique maximal μ-busy interval. The particular way in which this interval is defined allows finding

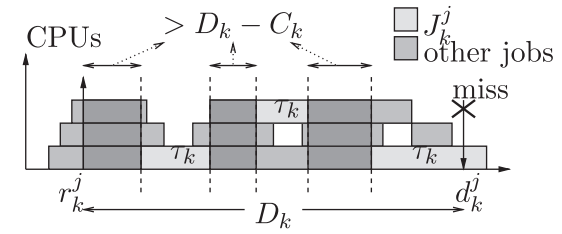

Fig. 1. Problem window.

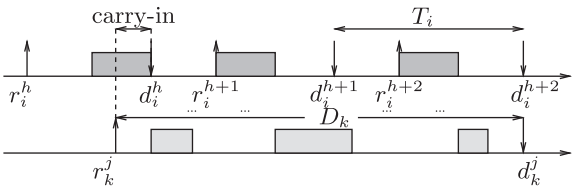

Fig. 2. Scenario that produces the maximum possible interference of task τ_i on a job of task τ_k when EDF is used.

a tighter upper bound on each carry-in contribution, improving the estimation of the overall interference. By mathematical derivation, the following theorem follows.

Theorem 3 (*BAK*, from [3]). *A task set τ is schedulable with global* EDF *if, for all $\tau_k \in \tau$, there is a $\lambda \in \{\lambda_k\} \cup \{U_\ell | U_\ell \geqslant \lambda_k, \ell < k\}$ such that*

$$\sum_{\tau_i \in \tau} \min(1, \beta_{i,k}(\lambda)) \leqslant m(1-\lambda)+\lambda, \quad (2)$$

where

$$\beta_{i,k}(\lambda) = \begin{cases} U_i\left(1+\frac{\max(0,T_i-D_i)}{D_k}\right) & \text{if} \quad U_i \leqslant \lambda \\ U_i\left(1+\frac{T_i}{D_k}\right) - \lambda\frac{D_i}{D_k} & \text{if} \quad U_i > \lambda. \end{cases}$$

The overall complexity of the above schedulability test is $O(n^3)$. A simplified $O(n^2)$ test can be derived testing only the case $\lambda = \lambda_k$. When deadlines are equal to periods, it is possible to reduce the above test to the GFB test of Theorem 2. However, when deadlines may be different from periods the tests are incomparable, as we will show in our simulations.

Baker and Cirinei [6] later modified Theorem 3, integrating it with techniques described in [15,16]. However, simulations in [6] show that the comparison with the BAK test is not favorable in the EDF case. We will therefore not consider the EDF test derived in [6] in our simulations.

3.3. BAR

Somewhat similar techniques have been applied by Baruah in [7], deriving another sufficient schedulability condition, presented below as Theorem 4. The rationale behind this derivation is as follows.

Consider any minimal 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. 4). Let t_a denote this job's arrival time: $t_a = t_d - D_k$. Let t_o denote the latest time-instant $\leqslant t_a$ at which at least one processor is idled in this EDF schedule. Let $A_k \doteq t_a - t_o$.

In order for τ_k's job to execute for strictly less than C_k time-units over $[t_a, t_d)$, it is necessary that all m processors be executing jobs other than τ_k's job for strictly more than $(D_k - C_k)$ time-units over $[t_a, t_d)$. Let us denote by Γ_k a collection of intervals, not necessarily contiguous, of cumulative length $(D_k - C_k)$ over $[t_a, t_d)$, during which all m processors are executing jobs other than τ_k's job in this EDF schedule.

For each $i, 1 \leqslant i \leqslant n$, let $I_k(\tau_i)$ denote the contribution of τ_i to the work done in this EDF schedule during $[t_o, t_a) \bigcup \Gamma_k$. Let us say that τ_i has a *carry-in job* in this EDF schedule if there is a job of τ_i that arrives before t_o and has not completed execution by t_o. By the definition of t_o, there can be at most $m-1$ such jobs. Upper bounds are computed in [7] on $I_k(\tau_i)$ if τ_i has no carry-in job (this is denoted as $I'_k(\tau_i)$), or if it does (denoted as $I''_k(\tau_i)$). These definitions and computations together yield the following sufficient schedulability condition; see [7] for the derivation.

Theorem 4 (*BAR from* [7]). *A task set τ is schedulable with global* EDF *if, for all $\tau_k \in \tau$ and all*

$$0 \leqslant A_k \leqslant \frac{C_\Sigma - D_k(m - U_{tot}) + \sum_{\tau_i \in \tau}(T_i - D_i)U_i + mC_k}{m - U_{tot}},$$

it is the case that[1]

[1] Note that [7] incorrectly presents Inequality 3 as not strict (⩽rather than<). We use the corrected version here.

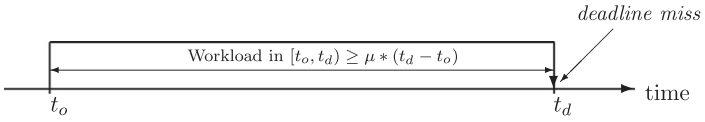

Fig. 3. μ-Busy interval.

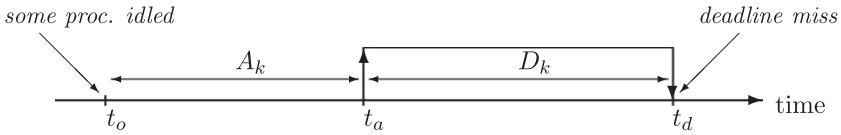

Fig. 4. Notation. A job of task τ_k arrives at t_a and misses its deadline at time-instant t_d. The latest time-instant prior to t_a when not all m processors are busy is denoted t_o.

$$\sum_{\tau_i \in \tau} I'_k(\tau_i) + I^{\epsilon}_k < m(A_k + D_k - C_k), \tag{3}$$

with

$$I^{\epsilon}_k \doteq \sum_{i|(m-1)\text{largest}} (I''_k(\tau_i) - I'_k(\tau_i)),$$

$$I'_k(\tau_i) \doteq \begin{cases} \min(dbf_i(A_k + D_k), A_k + D_k - C_k), & \text{if } i \neq k \\ \min(dbf_i(A_k + D_k) - C_k, A_k), & \text{if } i = k, \end{cases}$$

$$I''_k(\tau_i) \doteq \begin{cases} \min\left(\left\lfloor \frac{A_k + D_k}{T_i} \right\rfloor C_i + \min(C_i, (A_k + D_k) \bmod T_i), \right. \\ \qquad A_k + D_k - C_k), & \text{if } i \neq k \\ \min\left(\left\lfloor \frac{A_k + D_k}{T_i} \right\rfloor C_i + \min(C_i, (A_k + D_k) \bmod T_i) \right. \\ \qquad -C_k, A_k), & \text{if } i = k \end{cases}$$

and C_Σ denoting the sum of the $(m-1)$ largest execution times among all tasks.

When $U_{\text{tot}} < m$, this condition can be checked in pseudo-polynomial time.

3.4. LOAD

A different category of EDF-schedulability tests is based on the computation of the LOAD of a task set, defined as

$$load = \max_t \frac{\sum_{\tau_i \in \tau} dbf_i(t)}{t}. \tag{4}$$

Fisher et al. showed in [21] that it is sufficient to evaluate the maximum in the RHS of Eq. (4) over each point $\{D_j + kT_j \mid k \in \mathbb{N}, 1 \leqslant j \leqslant n\}$ until the least common multiple of all task periods. In the same paper, they show as well methods to further reduce the number of points to consider. However, the complexity of such methods is still exponential in the worst-case. To decrease the overall complexity, polynomial and pseudo-polynomial algorithms are proposed to compute an approximated estimation of the load within a given margin of error. See [20] for a detailed discussion concerning exact and approximation computations of load.

Different load-based sufficient schedulability tests for EDF have been proposed in [22,9,8]. In [4], the following result due to Baker and Baruah is shown to dominate the previous load-based conditions.

Theorem 5 (*LOAD from* [4]). *A task set τ is schedulable with global* EDF *if*

$$load \leqslant \max\{\mu - \lambda^{\mu}_{\max}, (\lceil \mu \rceil - 1) - \lambda^{\lceil \mu \rceil - 1}_{\max}\}, \tag{5}$$

where $\mu \doteq m - (m-1)\lambda_{\max}$, and $\lambda^{x}_{\max}$ is the sum of the $(\lceil x \rceil - 1)$ largest densities among all tasks.

The authors proved that the above EDF-schedulability test (i) is sustainable and (ii) has a processor speedup bound of

$$\frac{2(m-1)}{(3m-1) - \sqrt{5m^2 - 2m + 1}},$$

approaching $\frac{3+\sqrt{5}}{2} \simeq 2.62$ as $m \to \infty$.

3.5. BCL

Bertogna et al. presented in [15] a schedulability test with polynomial complexity, bounding, for each task τ_k, the interfering workload that can be produced in the scheduling window $\left[r^j_k, d^j_k\right)$ of a generic job J^j_k. This test has been later improved in [17], presenting an iterative procedure that allows tightening the estimation of the interfering workload, exploiting the information on the slack of each tasks. Suppose a lower bound S^{lb}_i is known on the slack of an interfering task τ_i. As shown in Fig. 5, the interfering contribution of τ_i in $\left[r^j_k, d^j_k\right)$ can be reduced – compared to Fig. 1 – by noting that the execution of the carry-in job J^h_i should be at least S^{lb}_i time-units from its deadline d^h_i. We hereafter describe the procedure used by the BCL test to check the schedulability of a task set by iteratively refining the slack lower bound of each task:

- The slack S^{lb}_k of each task is initialized to zero.
- Then, for each task τ_k, the following expression is computed

$$D_k - C_k - \left\lfloor \frac{1}{m} \sum_{i \neq k} \min\left(\Im^i_k, D_k - C_k + 1\right) \right\rfloor, \tag{6}$$

with

$$\Im^i_k \doteq \left\lfloor \frac{D_k}{T_i} \right\rfloor C_i + \min\left(C_i, \left(D_k \bmod T_i - S^{\text{lb}}_i\right)_0\right), \tag{7}$$

If the returned value is $> S^{\text{lb}}_k$, it is assigned to S^{lb}_k; if instead it is < 0, τ_k is marked as "potentially not schedulable".

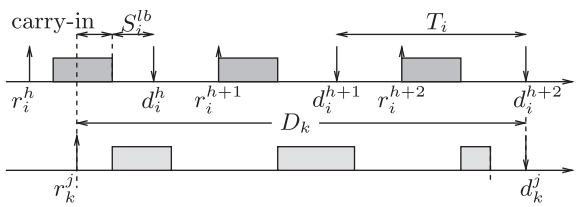

Fig. 5. Scenario with the maximum possible interference of τ_i on a job of τ_k with EDF, when S^{lb}_i is a safe lower bound on the slack of τ_i.

- If no task has been marked as potentially not schedulable, the task set is declared *schedulable*. Otherwise, the previous step is repeated.
- If during the last round no slack has been updated, the iteration stops and the task set is declared *not schedulable*.

The complexity of the procedure depends on the number of iterations, each one having complexity $O(n^2)$. A rough upper bound on the total number of iterations is $\sum_k (D_k - C_k) = O(nD_{\max})$. However, the test can be stopped after a finite number N of iterations. In this case, the total complexity of the test is $O(n^2 N)$. Simulations show negligible losses even with a very low N (equal to a few units).

3.6. RTA

In [14], a schedulability test has been derived based on the iterative estimation of the response time of each task. The procedure (RTA) is the same as procedure BCL, replacing the second step with the following one:

- For each task τ_k, compute R_k^{ub} as the smallest fixed point of the following expression, starting with $R_k^{\text{ub}} = C_k$, and failing if R_k^{ub} becomes $> D_k$:

$$R_k^{\text{ub}} \leftarrow C_k + \left\lfloor \frac{1}{m} \sum_{i \neq k} \min\left(\mathfrak{W}_i(R_k^{\text{ub}}), \mathfrak{I}_k^i, R_k^{\text{ub}} - C_k + 1\right) \right\rfloor,$$

with $\mathfrak{I}_k^i$ given by Eq. 7, and

$$\mathfrak{W}_i(L) = \left\lfloor \frac{L + D_i - C_i - S_i^{lb}}{T_i} \right\rfloor C_i + \min\left(C_i, \left(L + D_i - C_i - S_i^{lb}\right) \bmod T_i\right).$$

If the operation failed, mark τ_i as "potentially not schedulable"; otherwise, assign

$$S_k^{lb} = D_k - R_k^{\text{ub}}. \tag{8}$$

With this modification, the estimation of the interference on a task τ_k is tightened: while BCL considers the workload that the interfering tasks execute in the whole window $\left[r_k^j, d_k^j\right)$, RTA considers just the share that these tasks can execute during an upper bound of the response time of τ_k, i.e., in $\left[r_k^j, r_k^j + R_k^{\text{ub}}\right) \equiv \left[r_k^j, d_k^j - S_k^{lb}\right)$. To find the maximum workload an interfering task can execute in a window of length $R_k^{\text{ub}} = L$, consider Fig. 6, where the densest possible packing of jobs of a task τ_i, having a slack lower bound S_i^{lb}, is depicted. Based on this figure, it is possible to prove [14] that $\mathfrak{W}_i(L)$ is a valid upper bound on the workload a task τ_i can execute in a window of length L.

The overall complexity of the test is $O(n^3 D_{\max}^2)$, and therefore pseudo-polynomial. Differently from the BCL case, if the number of rounds is limited to N, the overall complexity is still pseudo-polynomial: $O(n^2 N D_{\max})$. The *average* complexity of the test can be significantly reduced, with a very limited performance degradation, replacing the term $R_k^{\text{ub}} - C_k - 1$ with $D_k - C_k - 1$ in the minimum of the fixed point iteration expression.

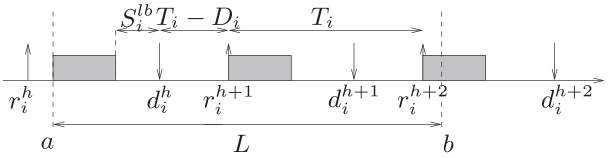

Fig. 6. Densest possible packing of jobs of τ_i, when S_i^{lb} is a safe lower bound on the slack of τ_i.

Both RTA and BCL are *sustainable* with respect to task periods [13]. More work is needed to verify the sustainability with respect to execution times and deadlines.

As a final remark, note that an improved version of BAR can be derived exploiting the slack lower bounds computed by RTA. In case RTA fails, BAR can be applied using a refined expression of the term $I_k''(\tau_i)$ (note the analogy with the term $\mathfrak{I}_k^i$ of Eq. 7 in BCL and RTA):

$$I_k''(\tau_i) \doteq \begin{cases} \min\left(\left\lfloor \frac{A_k + D_k}{T_i} \right\rfloor C_i + \min(C_i, (A_k + D_k) \bmod T_i - S_i^{\text{lb}}), A_k + D_k - C_k\right), & \text{if } i \neq k \\ \min\left(\left\lfloor \frac{A_k + D_k}{T_i} \right\rfloor C_i + \min(C_i, (A_k + D_k) \bmod T_i - S_i^{\text{lb}}) - C_k, A_k\right), & \text{if } i = k \end{cases} \tag{9}$$

The proposed integrated composition of RTA and BAR allows finding a larger number of schedulable task sets than with the simple serial combination of these tests. This is because we are able to *contemporarily* exploit both BAR's limitation on the number $(m-1)$ of carry-in jobs, and RTA's improved estimation of the potential carry-in contribution of each task.

3.7. FFDBF

Recently, Baruah et al. proposed in [10] a schedulability test based on the forced-forward demand bound function. The forced-forward demand bound function of a task τ_i is defined as follows:

$$\mathit{ffdbf}_i(t, \sigma) = \left\lfloor \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) C_i + \sigma \left((t - D_i) \bmod T_i - T_i + \frac{C_i}{\sigma} \right)_0 \right\rfloor.$$

Informally speaking, $\mathit{ffdbf}_i(t, \sigma)$ can be thought of the maximum cumulative execution requirement by jobs of task τ_i over an interval of length t, provided that execution outside the interval occurs on a speed-σ processor – see Fig. 7.

The forced-forward demand bound function of a task system τ is simply the sum of the forced-forward demand bound functions of its constituent tasks:

$$\mathit{ffdbf}(t, \sigma) = \sum_{\tau_i \in \tau} \mathit{ffdbf}_i(t, \sigma).$$

The following theorem makes use of the above definition to derive a sufficient schedulability condition – see [10] for a derivation.

Theorem 6 (*FFDBF from* [10]). *A task set τ is schedulable with global* EDF *if $\exists \sigma | \lambda_{\max} \leqslant \sigma < \frac{m - U_{\text{tot}}}{m-1} - \epsilon$ (with an arbitrarily small ϵ), such that $\forall t \geqslant 0$,*

$$\mathit{ffdbf}(t, \sigma) \leqslant (m - (m-1)\sigma) t \tag{10}$$

It can be proved that it is sufficient to check only those values of t in $\{kT_i + D_i | \, k \in \mathbb{N}\}_{i=1}^n$ that are smaller than[2]

$$\frac{\sum_{\tau_i \in \tau} C_i \left(1 - \frac{D_i}{T_i}\right)}{m - (m-1)\sigma - U_{\text{tot}}}. \tag{11}$$

Considering the given range of σ, the set of values of t to be checked has pseudo-polynomial size. Assume this testing set to be ordered. The following algorithm optimally exploits the result of Theorem 6, reducing the set of σ to be checked:

- Start considering $\sigma_{\text{cur}} = \lambda_{\max}$.
- Check condition 10 for the current σ_{cur}, for each t in the testing set in increasing order.

[2] We present here a tighter bound that the one in [10].

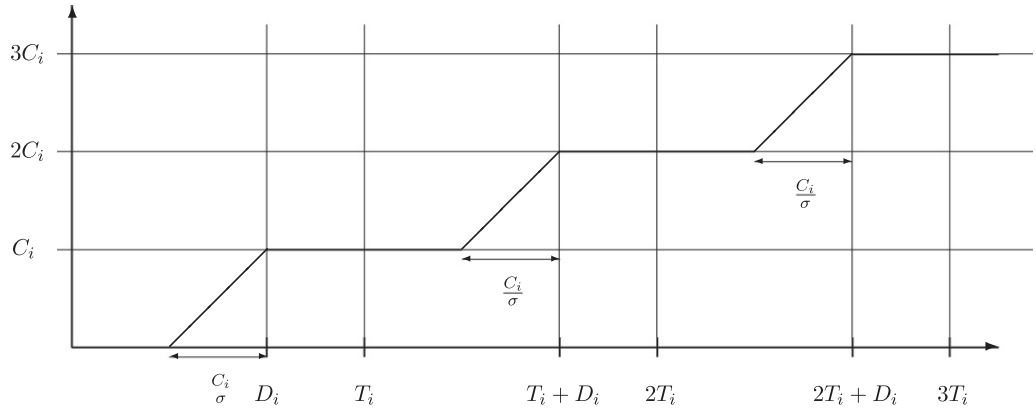

Fig. 7. Illustrating $\mathit{ffdbf}_i(t, \sigma)$.

- If the condition is violated at a given t, compute the first $\sigma \geqslant \sigma_{\text{cur}}$ that satisfies the condition, and assign $\sigma_{\text{cur}} \leftarrow \sigma$. (It is explained in [10] how this smallest $\sigma \geqslant \sigma_{\text{cur}}$ satisfying the condition can be computed.) If $\sigma \geqslant \frac{m-U_{\text{tot}}}{m-1} - \epsilon$, the test fails. Otherwise, return to the previous step, continuing with the new σ_{cur}, without re-checking the values of t that have been already checked with an older σ.

The complexity of the above algorithm is pseudo-polynomial. Moreover, the schedulability test has a processor speedup factor of $(2 - \frac{1}{m})$, which is the smallest possible for EDF, as proved by the tightness of Theorem 1.

3.8. QPA-based FFDBF

A faster version of the above algorithm can be derived extending the Quick convergence Processor-demand Analysis (QPA) described in [25] to the multiprocessor case under consideration. While the original QPA algorithm has been implemented to derive a necessary and sufficient EDF-schedulability condition for uniprocessor systems, we present here a QPA-based sufficient schedulability test for multiprocessor systems scheduled with global EDF. The algorithm modifies the way in which FFDBF evaluates for possible values of σ and t, detecting the same number of schedulable task sets with a smaller number of steps.

The schedulability condition of Theorem 6 is checked in the following way:

1. Start considering $\sigma_{\text{cur}} = \lambda_{\max}$.
2. Take t equal to the value given by 11.
3. While condition 10 is satisfied, take as the next point t' the minimum between (i) the last deadline before t, and (ii) the time at which $(m - (m-1)\sigma)t'$ equals $\mathit{ffdbf}(t, \sigma)$.

```
PrevD(t_end)
    t_max ← 0;
1   for (i : 1 ≤ i ≤ n) {
2       if(D_i < t_end){
3           t ← ⌊(t_end − D_i)/T_i⌋ ∗ T_i + D_i;
4           if (t = t_end) t ← t − T_i;
5           if (t > t_max) t_max ← t;
        }
    }
6   return t_max;
```

```
QPA-FFDBF(τ)
    σ ← λ_max.
1   while (σ < (m − U_tot)/(m − 1)){
2       t ← (Σ_{τ_i∈τ} C_i(1 − D_i/T_i)) / (m − (m − 1)σ − U_tot);
3       while (D_min < FFDBF(t,σ)/(m − (m − 1)σ) ≤ t)
4           t ← min{FFDBF(t,σ)/(m − (m − 1)σ), PrevD(t)};
5       if (FFDBF(t,σ)/(m − (m − 1)σ) ≤ D_min)
6           return (Feasible);
7       σ ← σ + ε;
    }
8   return (Infeasible);
```

Fig. 8. QPA-based version of FFDBF.

4. If the minimum deadline is reached, the algorithm declares the task set feasible.
5. Otherwise, increase σ by ϵ, until condition 10 is satisfied and return to step 2. If σ exceeds the limit $\frac{m-U_{tot}}{m-1}-\epsilon$, the algorithm fails.

Basically, instead of checking all deadlines in increasing order, the algorithm starts from the last deadline before the limit of (11) and jumps back to a previous point, skipping many intermediate deadlines. It continues until either the minimum deadline is reached – returning a positive answer – or a value is found for which condition 10 is violated. In this latter case, σ is incremented until the condition is satisfied. If σ reaches its limit, the algorithm fails. The pseudocode of the algorithm is given in Fig. 8.

The following theorem shows that procedure $\mathtt{QPA-FFDBF}(\tau)$ is correct and that it fully exploits the insight of Theorem 6.

Theorem 7. *Algorithm* $\mathtt{QPA-FFDBF}(\tau)$ *detects the same number of schedulable task sets as* FFDBF.

Proof. The proof is identical to the proof contained in [25], replacing the uniprocessor necessary and sufficient schedulability condition ($dbf(t) \leqslant t$) with Condition 10. Since $ffdbf(t,\sigma)$ and $t \rightarrow (m-(m-1)\sigma)t$ are monotonically non-decreasing functions of t, the same reasonings involving $dbf(t)$ and t in [25] can be applied to show that procedure $\mathtt{QPA-FFDBF}(\tau)$ optimally exploits Condition (10). Since all meaningful σ's are checked, $\mathtt{QPA-FFDBF}(\tau)$ detects the same number of schedulable task sets as FFDBF.

To prove that procedure $\mathtt{QPA-FFDBF}(\tau)$ converges in a finite number of steps, observe that the while loop at line 1 is evaluated a finite number of times, because σ is incremented by a fixed value ϵ at each round. Instead, the convergence of the inner loop at line 3 descends from the minimum at line 4; since $\mathtt{PrevD}(t)$ returns the last deadline before t, the loop is iterated at most once for each deadline. Since there are pseudo-polynomially many deadlines, the algorithm converges in a pseudo-polynomial number of steps. □

In our simulations, we found that the total number of points checked by $\mathtt{QPA-FFDBF}(\tau)$ is typically one order of magnitude smaller than with FFDBF.

4. Considerations

Only few dominance results can be claimed for the presented schedulability tests. Namely, it is possible to analytically prove that (i) FFDBF dominates GFB, and (ii) RTA dominates BCL. All other pairs of tests are incomparable, meaning that it is possible to find schedulable task sets that are detected by only one of the test, and vice-versa (differently from what is claimed in some of the related papers). We hereafter try to outline the main features that uniquely characterize each test. The differences among the tests are mainly related to the considered *problem window* – i.e., a window ending with a missed deadline – and the adopted bound on the *carry-in* contributions – i.e., the interference due to jobs not entirely contained inside the considered problem window.

- The condition used by BAK is derived considering a particular scheduling window, that allows deriving a bound on the maximum carry-in contribution of *each* task.
- The particularity of BAR is a bound provided on the *total number* of carry-in contributions, limited by $(m-1)$.
- A different bound on the *total number* of carry-in contributions is used in LOAD: $\lceil \mu \rceil - 1$, (see Theorem 5 for the definition of μ), along with another bound on *each* carry-in contribution.
- The peculiar advantage of BCL and RTA is the iterative estimation of the maximum carry-in contribution of *each* task (however, the total number of carry-in contributions is not bounded);
- The iterative way in which the problem window is defined in FFDBF allows deriving a bound on the *total amount* of carry-in that can be imposed on *any* task. A similar, although weaker, bound is found in the GFB case as well.

To better clarify the relative performances of each one of the above techniques, we performed an exhaustive set of simulations. We decided to analyze as well the performance of a new test (denoted as COMP) realized composing RTA, BAR and FFDBF. In particular, COMP is based on the following procedure:

1. Apply RTA to the given task set, storing each computed positive slack lower bound for later use.
2. If RTA does not succeed, apply BAR using Eq. 9 to compute the term $I''_k(\tau_i)$, exploiting the positive slack lower bounds derived at the previous step.
3. If also this test fails, apply FFDBF.
4. When, again, a negative result is obtained, it is at least possible to state a processor speedup result: the task set is not feasible on a platform in which each processor is $1/(2-\frac{1}{m}) = \frac{m}{2m-1}$ times as fast.s

As previously mentioned, the proposed integration of RTA and BAR allows finding a larger number of schedulable task sets than with the simple serial combination of both tests. Moreover, we added FFDBF to be able to claim a processor speedup result in case the test fails. We decided not to include other tests, because GFB and BCL are already dominated by, respectively, FFDBF and RTA, while BAK and LOAD would add very few schedulable task set detections to COMP (less than one over 10^6 generated task sets).

5. Experimental results

In this section, we show the results of the simulations we performed computing the number of schedulable task sets, among a randomly generated distribution, that are detected by each one of the analyzed schedulability tests; namely: GFB, BAK, BAR, LOAD, BCL, RTA, FFDBF and COMP.

To properly compare the schedulability tests described throughout this work, a first problem is how to generate a distribution of task sets that is representative of the general behavior of a real-time system. Using the method described in [18] to generate a uniform distribution of task sets with a desired total utilization is not so straightforward in the multiprocessor case, unless accepting tasks with utilization larger than one. Since such tasks would be trivially not feasible, we will adopt a different technique.

Another problem is related to the absence of exact feasibility and schedulability tests for globally scheduled multiprocessor systems. Since no such test is known, we don't have any term of comparison against which to evaluate the absolute performances of a given schedulability test. To sidestep this problem, an option is to use necessary (albeit not sufficient) conditions for feasibility. The tightest known necessary test with reasonable complexity is the one described in [5]. We will implement a pseudo-polynomial version of this test, and use this algorithm to decide which task set to consider from a randomly generated distribution: every task set that is rejected by this test, will also be excluded by our testing set.

5.1. Experiment setup

Each task is generated in the following way: utilization extracted according to an exponential distribution with mean σ_u,

removing tasks with utilization $U_i > 1$; period from a uniform distribution in [0,2000], and execution time accordingly computed as $C_i = U_i T_i$; deadline from a uniform distribution between C_i and T_i.

For each experiment, we generated 10^6 task sets according to the following procedure:

1. Initially, we extract a set of $m + 1$ tasks.
2. We then check if the generated task set passes the necessary condition for feasibility proposed in [5]. If not, we discard the task set, returning to step 1.
3. If instead the answer is positive, we apply each considered schedulability test to the generated task set.
4. Then, a new set is created adding a new task to the old set, returning to step 2.

This method allows generating task sets with a progressively larger number of elements, until the necessary condition for feasibility is violated.

The simulations have been performed for many different configurations, varying the number of processors m and the mean task utilization σ_u.

5.2. Evaluation of experiments

The results are shown in the following histograms. Each line represents the number of task sets proved schedulable by one specific test. The curves are drawn connecting a series of points, where each point at a utilization U_x represents the collection of task sets with total utilization in $\left[U_x - \frac{2}{100}^* m, U_x + \frac{2}{100}^* m\right)$. To give an *upper bound* on the number of feasible task sets, we included a continuous curve labeled with TOT, representing the distribution of generated task sets that meet the necessary feasibility condition. To help the reader understand the relative performances of the various algorithms, *keys are always ordered according to the total number of task sets detected* by the corresponding test: tests with a lower key position detect a lower number of task sets.

In Fig. 9, we show the case with $m = 2$ processors. As we can see, COMP is able to detect a larger number of schedulable task sets than any of the existing tests. Among those latter ones, the best performances are shown by BAR, RTA and FFDBF. The curve of BCL is pretty close, although slightly lower. The remaining tests have instead much worse performances.

We found that only very few task sets are found schedulable by BAK (1 task set) or LOAD (12 task sets) and not by COMP over 10^6 generated task sets.

In Fig. 10, we present the case with $m = 4$ processors. The situation is more or less the same as before, except for the worse behavior of BAR and FFDBF. In the BAR case, this can be explained with the larger number of carry-in contributions $(m - 1)$ that the test needs to take into account when the number of processors increases. The worse performance of FFDBF is instead due to the larger number n of tasks per set when more processors are added. When n is large, there are more chances to extract a large $\lambda_{\max}$, reducing the RHS term of Eq. 10.

The best performances are still shown by COMP and RTA, although the difference between the curves of both algorithms is reduced. BCL's curve is still close, while all other algorithms have significant losses. The larger distance from the TOT curve is motivated by the worse performances of EDF when the number of processor increases, and does not seem a weak point of the tests. Further increasing the number of processors, the above results are magnified, as shown in Fig. 11 for the case with $m = 8$ processors. In this case, COMP, RTA and BCL are the only tests that guarantee reasonable acceptance rates, while the remaining algorithms are almost useless. This is confirmed by the negligible distance between the curves of RTA and COMP. Among sustainable tests, GFB allows achieving performances comparable to LOAD, at a much smaller computational cost.

Increasing the mean utilization of the generated task sets to $\sigma_u = 0.50$, we obtained the histograms in Fig. 12 for the case with $m = 2$. Again, it is possible to see that the results are similar to the corresponding case with $\sigma_u = 0.25$. We also performed experiments with $\sigma_u = 0,10$: even if the shape of the curves slightly changes, the relative ordering of the tests in terms of schedulability performances remains the same.

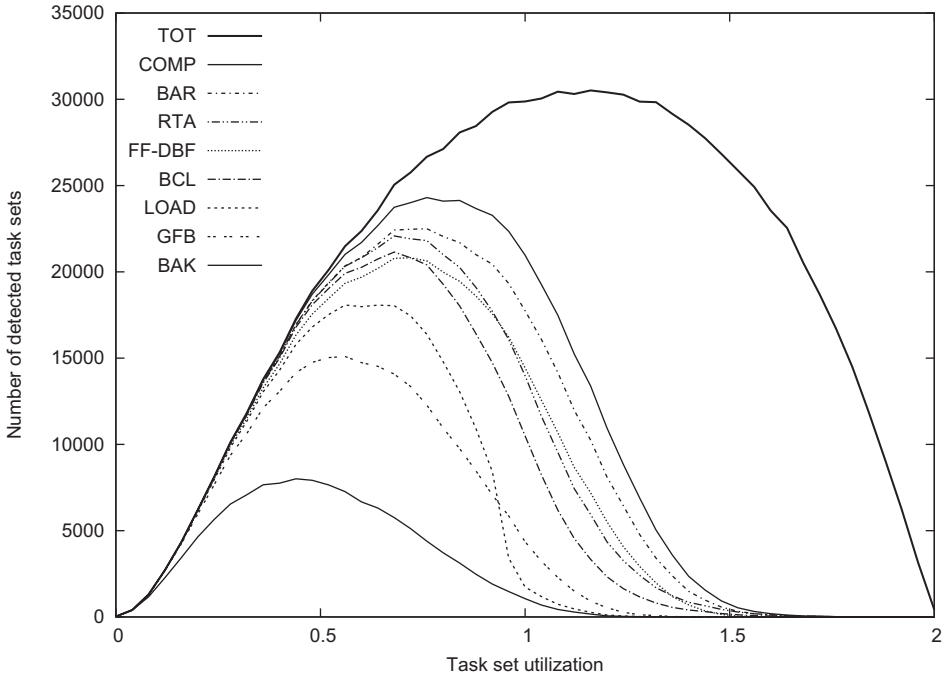

Fig. 9. Experiment with two processors and $\sigma_u = 0.25$.

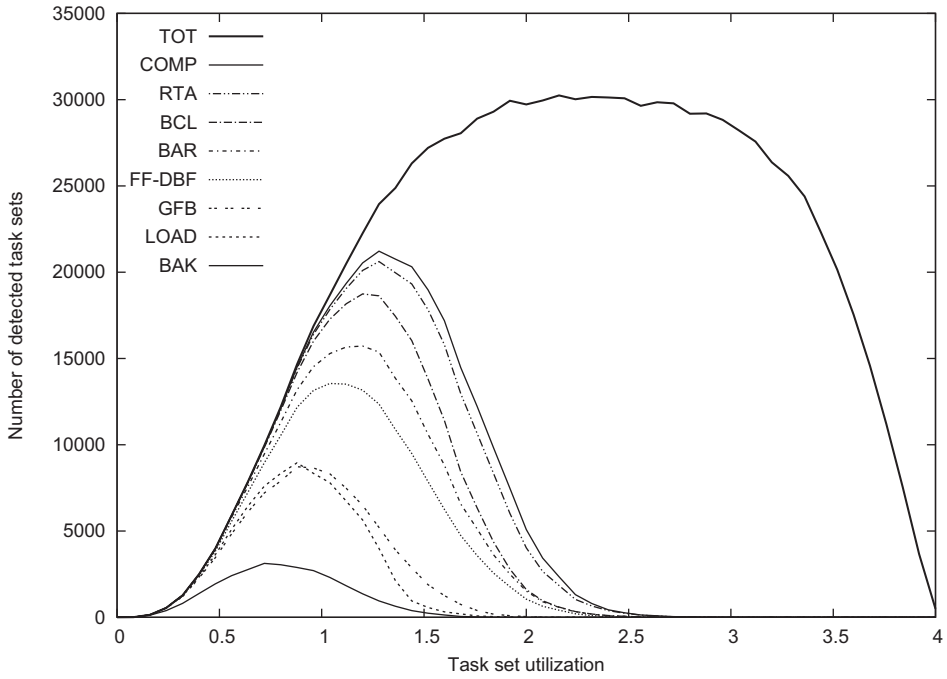

Fig. 10. Experiment with four processors and $\sigma_u = 0.25$.

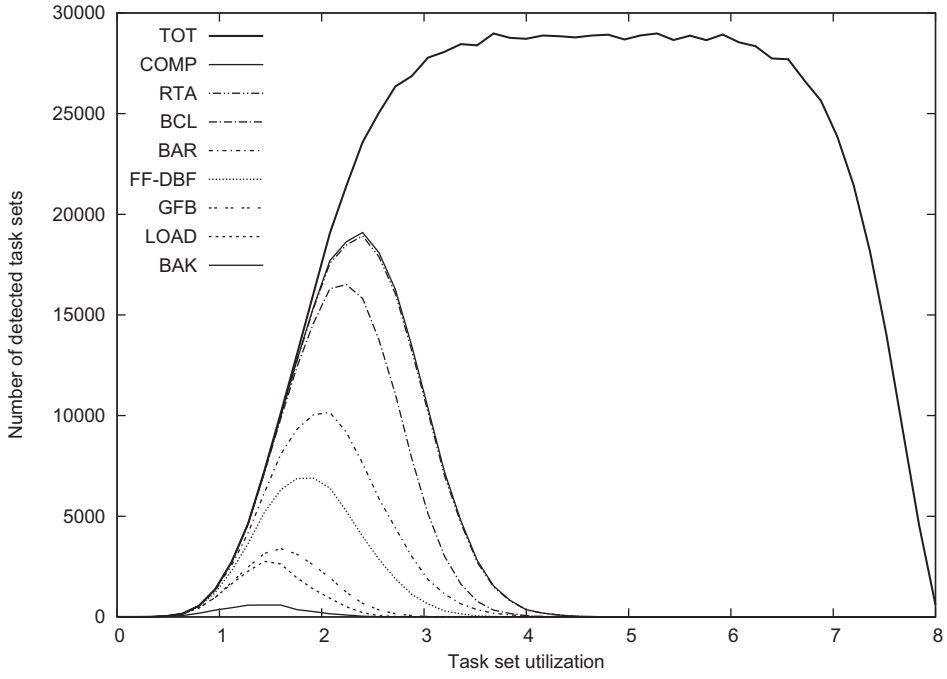

Fig. 11. Experiment with eight processors and $\sigma_u = 0.25$.

As a side remark, note that, since we are using the constrained deadline model, no scheduling algorithm can reach a schedulable utilization in the number of processors. We included the continuous curve labeled with TOT just to give an *upper bound* on the number of feasible task sets. This curve does not represent the number of EDF-schedulable task sets, neither it indicates how many task sets are feasible. It gives an indication on how many generated task sets are not for sure infeasible – using techniques from [5] – at the considered utilizations. If an exact feasibility test existed, its curve would be below the TOT curve. Moreover, considering that EDF is not optimal for multiprocessors, a hypothetical necessary and sufficient schedulability test for EDF would have an even lower curve.

Finally, to evaluate the effectiveness of our QPA-based implementation of FFDBF, we compared the number of steps taken in all experiments by procedure QPA − FFDBF with the number of deadlines checked using the original FFDBF algorithm. With two processors, the total number of points checked by

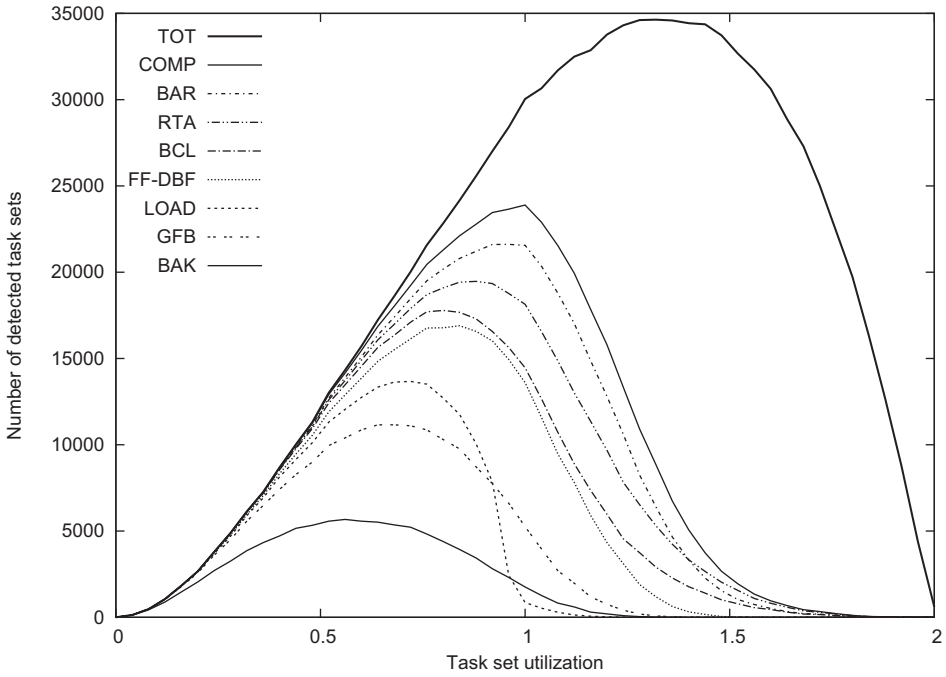

Fig. 12. Experiment with two processors and $\sigma_u = 0.50$.

$\texttt{QPA} - \texttt{FFDBF}(\tau)$ is 87% less than with the original algorithm. It is worth noting that we found task sets for which the original algorithm needed to check more than 5×10^6 deadlines, while $\texttt{QPA} - \texttt{FFDBF}(\tau)$ never needed to check more than 10^5 points. Increasing the mean utilization of the generated task sets to 0.50, the improvement is similar (89%), as it is increasing the number of processor to $m = 4(87\%)$ and $m = 8(90\%)$.

6. Conclusions

We presented a detailed description, with a homogenous notation, of the main existing schedulability tests for global EDF scheduling on an identical multiprocessor platform, considering sporadic task sets with constrained deadlines. The performances of all tests have been compared by means of exhaustive simulations as well as of analytical considerations, taking into account dominance relations, processor speedup factors, run-time complexities, sustainability issues, and average performances over different sets of randomly generated loads.

For many of the existing tests, we proposed extensions that are able to either find a larger number of schedulable task sets, or decrease the run-time complexity. An accurate analysis of the performances allowed the creation of an algorithm (COMP) that combines the major advantages of the existing techniques, in terms of both run-time complexity and success ratio. This algorithm can be efficiently used to check the schedulability of a set of tasks with hard real-time requirements, with an optimal processor speedup factor and a pseudo-polynomial run-time complexity. Our proposed improvement to the FFDBF test, obtained by incorporating the notion of quick convergence from Zhang and Burns' QPA [25], provides a further performance improvement to this combined test. When event faster schedulability tests are needed, comparable performances can be reached using a polynomial $O(n^2)$ test (BCL). For even faster decisions, a linear complexity test (GFB) can be adopted with acceptable performances, as long as the number of processors is small, or there is no heavy task. This latter test allows as well a sustainable schedulability analysis at a small computational cost.

References

[1] Theodor P. Baker, Sanjoy Baruah, Sustainable multiprocessor scheduling of sporadic task systems, in: Proceedings of the EuroMicro Conference on Real-Time Systems, IEEE Computer Society Press, Dublin, Ireland, 2009.
[2] Theodore P. Baker, Multiprocessor EDF and deadline monotonic schedulability analysis, in: Proceedings of the IEEE Real-Time Systems Symposium, IEEE Computer Society Press, 2003, pp. 1–5.
[3] Theodore P. Baker, An analysis of EDF schedulability on a multiprocessor, IEEE Transactions on Parallel and Distributed Systems 16 (8) (2005) 760–768.
[4] Theodore P. Baker, Sanjoy Baruah, An analysis of global EDF schedulability for arbitrary sporadic task systems, Real-Time Systems: The International Journal of Time-Critical Computing 43 (1) (2009) 3–24.
[5] Theodore P. Baker, Michele Cirinei, A necessary and sometimes sufficient condition for the feasibility of sets of sporadic hard-deadline tasks, in: Proceedings of the Work-In-Progress (WIP) session of the 27th IEEE Real-Time Systems Symposium (RTSS'06), Rio de Janeiro, Brazil, December 2006.
[6] Theodore P. Baker, Michele Cirinei, A unified analysis of global EDF and fixed-task-priority schedulability of sporadic task systems on multiprocessors, Journal of Embedded Computing, in press. TR available at <http://www.cs.fsu.edu/research/reports/TR-060401.pdf>>.
[7] Sanjoy Baruah, Techniques for mutliprocessor global schedulability analysis, in: Proceedings of the IEEE Real-time Systems Symposium, IEEE Computer Society Press, Tucson, December 2007.
[8] Sanjoy Baruah, Theodore P. Baker, Global EDF schedulability analysis of arbitrary sporadic task systems, in: Euromicro Conference on Real Time Systems, PCzech Republic, Berlin, 2008.
[9] Baruah Sanjoy, Theodore P. Baker, Schedulability analysis of global EDF, Real-Time Systems: The International Journal of Time-Critical Computing 38 (3) (2008) 223–235.
[10] Sanjoy Baruah, Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, Sebastian Stiller, Implementation of a speedup-optimal global EDF schedulability test, in: Proceedings of the EuroMicro Conference on Real-Time Systems, IEEE Computer Society Press, Dublin, Ireland, July 2009.
[11] Sanjoy Baruah, Alan Burns, Sustainable scheduling analysis, in: Proceedings of the IEEE Real-time Systems Symposium, IEEE Computer Society Press, Rio de Janeiro, December 2006.
[12] Sanjoy Baruah, Aloysius K. Mok, Louis E. Rosier, Preemptively scheduling hard-real-time sporadic tasks on one processor, in: Proceedings of the 11th Real-Time Systems Symposium, IEEE Computer Society Press, Orlando, Florida, 1990.
[13] Marko Bertogna, Real-Time Scheduling Analysis for Multiprocessor Platforms, Ph.D. thesis, Scuola Superiore Sant'Anna, Pisa, 2008.
[14] Marko Bertogna, Michele Cirinei, Response-time analysis for globally scheduled symmetric multiprocessor platforms, in: 28th IEEE Real-Time Systems Symposium (RTSS), Tucson, Arizona (USA), 2007.
[15] Marko Bertogna, Michele Cirinei, Giuseppe Lipari, Improved schedulability analysis of EDF on multiprocessor platforms, in: Proceedings of the EuroMicro

Conference on Real-Time Systems, IEEE Computer Society Press, Palma de Mallorca, Balearic Islands, Spain, July 2005.
[16] Marko Bertogna, Michele Cirinei, Giuseppe Lipari, New schedulability tests for real-time tasks sets scheduled by deadline monotonic on multiprocessors, in: Proceedings of the 9th International Conference on Principles of Distributed Systems, IEEE Computer Society Press, Pisa, Italy, December 2005.
[17] Bertogna Marko, Cirinei Michele, Lipari Giuseppe, Schedulability analysis of global scheduling algorithms on multiprocessor platforms, IEEE Transactions on Parallel and Distributed Systems 20 (4) (2009) 553–566. April.
[18] Enrico Bini, Giorgio C. Buttazzo, Biasing effects in schedulability measures, in: ECRTS '04: Proceedings of the 16th Euromicro Conference on Real-Time Systems, Catania, Italy, July 2004, pp. 196–203.
[19] Liliana Cucu, Joël Goossens, Feasibility intervals for fixed-priority real-time scheduling on uniform multiprocessors, in: ETFA, Prague, September 2006.
[20] Nathan Fisher, The Multiprocessor Real-Time Scheduling of General Task Systems, Ph.D. Thesis, Department of Computer Science, The University of North Carolina at Chapel Hill, 2007.
[21] Nathan Fisher, Theodore P. Baker, Sanjoy Baruah, 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, IEEE Computer Society Press, Sydney, Australia, August 2006.
[22] Nathan Fisher, Sanjoy Baruah, The global feasibility and schedulability of general task models on multiprocessor platforms, in: Proceedings of the EuroMicro Conference on Real-Time Systems, IEEE Computer Society Press, Pisa, Italy, July 2007.
[23] Goossens Joël, Funk Shelby, Baruah Sanjoy, Priority-driven scheduling of periodic task systems on multiprocessors, Real Time Systems 25 (2–3) (2001) 187–205.
[24] Cynthia A. Phillips, Cliff Stein, Eric Torng, Joel Wein, 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.
[25] Zhang Fengxiang, Burns Alan, Schedulability analysis for real-time systems with edf scheduling, IEEE Transactions on Computers 58 (2009) 1250–1258.

Marko Bertogna is assistant professor at the Scuola Superiore S. Anna, Pisa. He got a Ph.D. in Computer Sciences in 2008 with a dissertation on Real-Time Systems for Multicore Platforms. He graduated magna cum laude in Telecommunication Engineering at the University of Bologna in 2002. In 2006 he visited the University of North Carolina at Chapel Hill, working with prof. Sanjoy Baruah on scheduling algorithms for single and multicore real-time systems. His research interests include scheduling and schedulability analysis of real-time multiprocessor systems, protocols for the exclusive access to shared resources, resource reservation algorithms and reconfigurable devices.

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 resource-allocation and sharing in distributed computing environments.