Evaluation of existing schedulability tests for global EDF

Marko Bertogna
Scuola Superiore Sant'Anna
Pisa, Italy
Email: marko@sssup.it

***Abstract*—The increasing attention on global scheduling algorithms for identical multiprocessor platforms produced different, independently developed, schedulability tests. However, the existing relations among such tests have not been sufficiently clarified, so that it is difficult to understand which strategy provides the best performances in a particular scenario.**

In this paper, we will summarize the main existing results for the schedulability analysis of multiprocessor systems scheduled with global EDF, showing, when possible, existing dominance relations. We will compare these algorithms taking into consideration different aspects, namely, run-time complexity, average performances over a randomly generated workload, sustainability properties and speedup factors.

I. Introduction

The theoretical scheduling and schedulability analysis of multiprocessor systems with migration support is recently receiving an increasing attention in the real-time community. The recent advancements in the multiprocessor technology are reducing the migration-related penalties of global scheduling algorithms, rendering it more likely the adoption of such kind of schedulers in a real platform.

In this paper we will 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 will propose an exhaustive set of simulations, analyzing which test is able to detect the larger number of schedulable task sets, for different randomly generated distributions of task set parameters.

Moreover, we will present observations regarding the practical application of the considered tests. 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. Having a fast schedulability test allows, for example, to check on-line if tasks can be dynamically admitted into a running system without causing any deadline to be missed. Efficient *run-time admission control tests* are particularly useful for highly varying workloads, when it is important to promptly decide if to admit a new hard real-time instance, avoiding to waste a significant processor share for non-functional scheduling purposes. More complex solutions can instead be used for off-line analysis, when there are no particular timing requirements.

This work has been supported in part by the European Commission under the ACTORS ICT project (FP7/2008/ICT/216586).

II. 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* $D_k \geq C_k$, and a *period*, or *minimum interarrival time*, T_k. In this paper, we will consider only *constrained deadline systems*, for which $D_i \leq T_i, \ \forall \tau_i \in \tau$.

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 $t \in [r_k^j, f_k^j)$. 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.

All tasks are assumed to be *independent*, meaning that no structure is shared, except for the computing units, and no data is contemporarily requested by more than one task. This simplification allows ignoring the blocking effects associated to 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

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

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

A. 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 valid as well for strictly periodic task systems. Nothing is known instead about the sustainability w.r.t. deadline relaxations.

The concept of sustainability can be as well extended to schedulability tests. A test is sustainable if, given a task set passing the test, another task set with reduced timing requirements would as well pass the test. It is possible to build sustainable schedulability tests even for task systems scheduled with a non-sustainable algorithm.

B. Processor speedup factor

A metric that can be used to characterize the performances 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 systems 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.*

The "distance" of the processor speedup factor of an EDF-schedulability test from this tight bound of $(2 - \frac{1}{m})$ can be used as a measure of the optimality of the test.

III. Schedulability tests for global EDF

We will remind here 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 below described.

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 is not a critical instant for sporadic task systems, the above result cannot be used for the systems considered in this paper.

Given the extreme complexity of finding exact results, a more conceivable approach is considering sufficient albeit not necessary schedulability conditions. Finding efficient schedulability tests is therefore at least as important as finding "good" scheduling algorithms.

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.

A. 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*

$$\lambda_{\text{tot}} \leq m(1 - \lambda_{\max}) + \lambda_{\max}. \tag{1}$$

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

B. 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.

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

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

where

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

The overall complexity of the above schedulability test is $\mathrm{O}(n^3)$. A simplified $\mathrm{O}(n^2)$ 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 can be different from

periods, both test are incomparable, as we will show in our simulations.

Baker and Cirinei later modified Theorem 3 in [5], trying to integrate it with techniques described in [15], [16]. Anyway, simulations by the same authors show that the comparison with the BAK test is not favorable in the EDF case. We will therefore avoid to describe the EDF test derived in [5].

C. BAR

Somewhat similar techniques have been applied by Baruah in [6], deriving the following condition[1].

Theorem 4 (BAR from [6])**.** *A task set τ is schedulable with global* EDF *if, for all $\tau_k \in \tau$ and all*

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

it is

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

with

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

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

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

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

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

D. LOAD

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

$$\text{LOAD} = \max_t \frac{\sum_{\tau_i \in \tau} \text{DBF}_i(t)}{t}. \tag{4}$$

Fisher *et al.* showed in [21] that it is sufficient to evaluate the maximum in the RHS of Equation (4) over each point $\{D_j + kT_j \mid k \in \mathbb{N}, 1 \leq j \leq n\}$ 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. For a complete survey on how to efficiently check load-based conditions, see [20].

Different load load-based sufficient schedulability tests for EDF have been proposed in [22], [8], [7]. In [9], the following result due to Baker and Baruah is shown to dominate the previous load-based conditions.

Theorem 5 (LOAD from [9])**.** *A task set τ is schedulable with global* EDF *if*

$$\text{LOAD} \leq \max\{\mu - \lambda^\mu_{\max}, (\lceil \mu \rceil - 1) - \lambda^{\lceil \mu \rceil - 1}_{\max}\}, \tag{5}$$

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 $m \to \infty$.

E. 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 $[r^j_k, r^j_k + D_k]$ 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. We hereafter describe this procedure (BCL).

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

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

with

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

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".
- 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

[1] In the original paper, Inequality (3) is not strict. However, we found that this would underestimate the case in which there are exactly m tasks interfering for more than $(A_k + D_k - C_k)$. To preserve the correctness of the test, we restated the test using a strict inequality.

$\sum_k (D_k - C_k) = O(nD_{\max})$. 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).

F. 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 $> D_k$:

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

  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

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

It can be proved that 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.

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.

G. FF-DBF

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

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

The following theorem makes use of the above definition to derive a sufficient schedulability condition.

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

$$\sum_{\tau_i \in \tau} \text{FF DBF}_i(t,\sigma) \le (m - (m-1)\sigma)t \tag{9}$$

It can be proved that it is sufficient to check only those values of t in

$$\left\{ kT_j + D_k, kT_j + D_k - \min\left(\frac{C_i}{\sigma}, D_i\right) \mid k \in \mathbb{N} \right\}_{i=1}^{n}$$

that are smaller than[2]

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

Considering the given range of σ, the set of values of t to be checked has pseudopolynomial 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 (9) for the current σ_{cur}, for each t in the testing set in increasing order.
- if the condition is violated at a given t, compute the first $\sigma \ge \sigma_{\text{cur}}$ that satisfies the condition, and assign $\sigma_{\text{cur}} \leftarrow \sigma$. If $\sigma \ge \frac{m - U_{\text{tot}}}{m-1} - \epsilon$, 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 pseudopolynomial. Moreover, the schedulability test has a processor speedup factor of $\left(2 - \frac{1}{m}\right)$, which is the smallest possible for EDF, as proved by the tightness of Theorem 1.

IV. Considerations

Only few dominance results can be claimed for the presented schedulability tests. Namely, it is possible to analytically prove that (i) FF-DBF 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 viceversa (differently from what is claimed in some of the related papers). We hereafter try to outline the main peculiar 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.

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

- 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 in not bounded);
- The iterative way in which the problem window is defined in FF-DBF 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. Since we found examples of task sets proving the incomparability between any two of the above techniques, we decided to analyze as well the performance of a new test (denoted as COMP) realized composing some of the techniques adopted by the described tests. 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 the positive slack lower bounds for a refined computation of the term $I''_k(\tau_i)$ (note the analogy with the term $\mathfrak{I}^i_k$ of Eq. (7) in BCL and RTA):

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

3) If also this test fails, apply FF-DBF.
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 $1/\left(2-\frac{1}{m}\right) = \frac{m}{2m-1}$ times as fast.

The proposed integrated composition of RTA and BAR allows finding a larger number of schedulable task sets than with the simple serial combination of both tests. This is because COMP is 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.

Moreover, we added the serial combination of FF-DBF 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, FF-DBF and RTA, while BAK and LOAD would add very few schedulable task set detections to COMP (less than one over $100\ 000$ generated task sets).

Since some of the composing algorithms can be not sustainable, it is difficult to prove any sustainability property for COMP, needing further investigation.

V. 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, FF-DBF 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 [4]. 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.

A. Experiment Setup

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

For each experiment, we generated $1\ 000\ 000$ 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 [4]. 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.

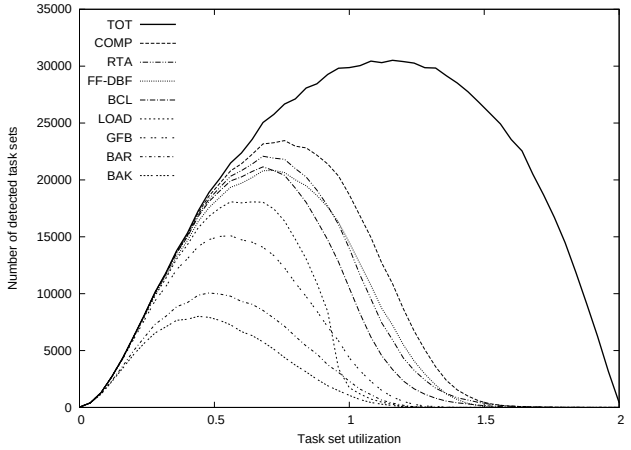

Figure 1. Experiment with 2 processors and $\sigma_u = 0.25$.

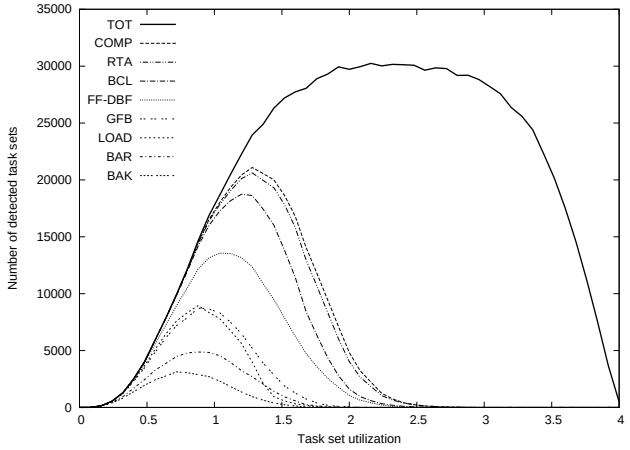

Figure 2. Experiment with 4 processors and $\sigma_u = 0.25$.

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

B. 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, each one representing the collection of task sets that have total utilization in a range of 4% near the point. 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 understanding 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 Figure 1, 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 RTA and FF-DBF. 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 (10 task sets) and not by COMP, over 1 000 000 generated task sets.

In Figure 2, we present the case with $m = 4$ processors. The situation is more or less the same as before, except for the worse behavior of FF-DBF, which has now a much lower curve. This can be 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. (9).

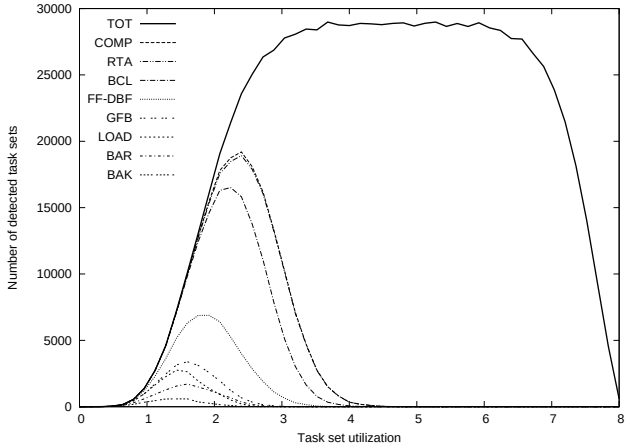

Figure 3. Experiment with 8 processors and $\sigma_u = 0.25$.

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 doesn't seem a weak point of the tests. Further increasing the number of processors, the above results are magnified, as shown in Figure 3 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 can be noticed as well noting 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 Figure 4. Again,

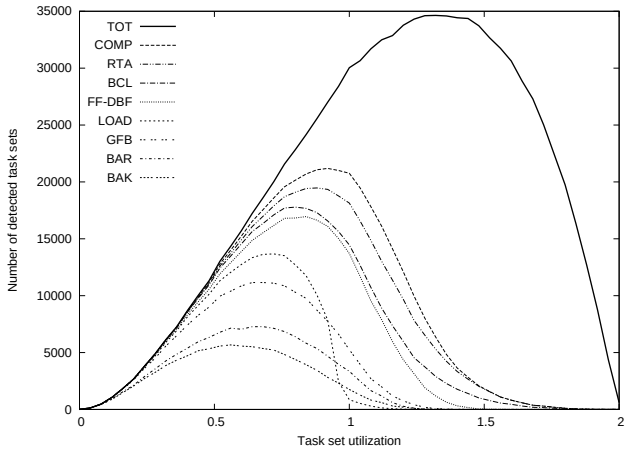

Figure 4. Experiment with 2 processors and $\sigma_u = 0.50$.

it is possible to see that the results are similar to the above cases. 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.

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 [4] — 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.

VI. 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.

We proposed an algorithm (COMP) that combines the major advantages of the existing techniques. This algorithm can be efficiently used to check the schedulability of a set of task sets with hard real-time requirements, with an optimal processor speedup factor and a pseudopolynomial run-time complexity. When 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 Baker and Sanjoy Baruah. Sustainable multiprocessor scheduling of sporadic task systems. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, Dublin, Ireland, July 2009. IEEE Computer Society Press.

[2] Theodore P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *Proceedings of the IEEE Real-Time Systems Symposium*. IEEE Computer Society Press, December 2003.

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

[4] Theodore P. Baker and 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.

[5] Theodore P. Baker and Michele Cirinei. A unified analysis of global EDF and fixed-task-priority schedulability of sporadic task systems on multiprocessors. *Journal of Embedded Computing*, 2007. To appear. TR available at http://www.cs.fsu.edu/research/reports/TR-060401.pdf.

[6] Sanjoy Baruah. Techniques for mutliprocessor global schedulability analysis. In *Proceedings of the IEEE Real-time Systems Symposium*, Tucson, December 2007. IEEE Computer Society Press.

[7] Sanjoy Baruah and Theodore P. Baker. Global EDF schedulability analysis of arbitrary sporadic task systems. In *Euromicro Conference on Real Time Systems*, Prague, Czech Republic, 2008.

[8] Sanjoy Baruah and Theodore P. Baker. Schedulability analysis of global EDF. *Real-Time Systems: The International Journal of Time-Critical Computing*, 38(3):223–235, 2008.

[9] Sanjoy Baruah and Theodore P. Baker. An analysis of global EDF schedulability for arbitrary sporadic task systems. *Real-Time Systems: The International Journal of Time-Critical Computing*, 2009. Accepted for publication.

[10] Sanjoy Baruah, Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, and Sebastian Stiller. Implementation of a speedup-optimal global EDF schedulability test. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, Dublin, Ireland, July 2009. IEEE Computer Society Press.

[11] Sanjoy Baruah and Alan Burns. Sustainable scheduling analysis. In *Proceedings of the IEEE Real-time Systems Symposium*, Rio de Janeiro, December 2006. IEEE Computer Society Press.

[12] Sanjoy Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the 11th Real-Time Systems Symposium*, Orlando, Florida, 1990. IEEE Computer Society Press.

[13] Marko Bertogna. *Real-Time Scheduling Analysis for Multiprocessor Platforms*. PhD thesis, Scuola Superiore Sant'Anna, Pisa, 2008.

[14] Marko Bertogna and 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, and Giuseppe Lipari. 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.

[16] Marko Bertogna, Michele Cirinei, and 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*, Pisa, Italy, December 2005. IEEE Computer Society Press.

[17] Marko Bertogna, Michele Cirinei, and Giuseppe Lipari. Schedulability analysis of global scheduling algorithms on multiprocessor platforms. *IEEE Transactions on Parallel and Distributed Systems*, 20(4):553–566, April 2009.

[18] Enrico Bini and 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.

[19] Liliana Cucu and 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*. PhD thesis, Department of Computer Science, The University of North Carolina at Chapel Hill, 2007.

[21] Nathan Fisher, Theodore P. Baker, and 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*, Sydney, Australia, August 2006. IEEE Computer Society Press.

[22] Nathan Fisher and Sanjoy Baruah. The global feasibility and schedulability of general task models on multiprocessor platforms. In *Proceedings of the EuroMicro Conference on Real-Time Systems*, Pisa, Italy, July 2007. IEEE Computer Society Press.

[23] Joël Goossens, Shelby Funk, and Sanjoy Baruah. Priority-driven scheduling of periodic task systems on multiprocessors. *Real Time Systems*, 25(2–3):187–205, 2001.

[24] Cynthia A. Phillips, Cliff Stein, Eric Torng, and 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.