

On the Pitfalls of Resource Augmentation Factors and Utilization Bounds in Real-Time Scheduling*

Jian-Jia Chen¹, Georg von der Brüggen², Wen-Hung Huang³, and Robert I. Davis⁴

1 TU Dortmund University, Dortmund, Germany

jian-jia.chen@cs.uni-dortmund.de

2 TU Dortmund University, , DortmundGermany

georg.von-der-brueggen@tu-dortmund.de

3 TU Dortmund University, Dortmund, Germany

wen-hung.huang@tu-dortmund.de

4 University of York, York, UK; and

INRIA Paris, Paris, France

rob.davis@york.ac.uk

Abstract

In this paper, we take a careful look at speedup factors, utilization bounds, and capacity augmentation bounds. These three metrics have been widely adopted in real-time scheduling research as the *de facto* standard theoretical tools for assessing scheduling algorithms and schedulability tests. Despite that, it is not always clear how researchers and designers should interpret or use these metrics. In studying this area, we found a number of surprising results, and related to them, ways in which the metrics may be misinterpreted or misunderstood. In this paper, we provide a perspective on the use of these metrics, guiding researchers on their meaning and interpretation, and helping to avoid pitfalls in their use. Finally, we propose and demonstrate the use of *parametric augmentation functions* as a means of providing nuanced information that may be more relevant in practical settings.

1998 ACM Subject Classification C.3 Real-Time and Embedded Systems, I.1.2 Analysis of Algorithms

Keywords and phrases real-time systems, speedup factors, utilization bounds, capacity augmentation bounds

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2017.9

1 Introduction

The performance of schedulability tests and scheduling algorithms for real-time systems can be compared in different ways. These can be broadly classified into two categories:

- *Theoretical methods* include deriving dominance relationships, utilization bounds, and various forms of resource augmentation factors, such as speedup factors, capacity augmentation bounds, or approximation ratios. These latter approaches typically give a worst-case comparison against a specific competitor, i.e. against an alternative schedulability test for the same or a different scheduling algorithm.

* This paper is supported by the DFG, as part of the Collaborative Research Center SFB876 (<http://sfb876.tu-dortmund.de/>), and by the EPSRC grant, MCCps (EP/K011626/1), and by the Inria International Chair program. EPSRC Research Data Management: No new primary data was created during this study.



- *Empirical methods* include simulation of the scheduling algorithm, evaluation of the schedulability test on synthetic task sets, case studies, and experiments on real hardware. These approaches typically facilitate an average-case comparison against a number of different scheduling algorithms or schedulability tests. See [36] for a review.

This paper is concerned primarily with theoretical methods. The main approaches are outlined in more detail below. Note, when we discuss comparisons between scheduling algorithms, then we are normally referring to comparisons between exact schedulability tests for those algorithms. Comparisons are also possible using sufficient schedulability tests, thus evaluating the performance of different approximations.

- *Dominance Relationships* are used to indicate if one scheduling algorithm or schedulability test always outperforms another. For example, schedulability test \mathcal{X} is said to dominate test \mathcal{Y} if every task set that is schedulable according to test \mathcal{Y} is also schedulable according to test \mathcal{X} , and there are some task sets that are schedulable according to \mathcal{X} but not according to \mathcal{Y} . Proving a dominance relationship shows that the dominant method is always better, at least in terms of schedulability; however, no indication is given as to how good the schedulability tests (or algorithms) actually are; a dominant test may still have poor performance, just not quite as poor as that of the test that it dominates.
- *Utilization Bounds* [65, 1, 5, 49] seek to bound the minimum total utilization of any unschedulable task set for a given scheduling algorithm and task model. Thus any task set with a total utilization U_{sum} no greater than the bound is guaranteed to be schedulable. Examples include the Liu and Layland bounds for implicit-deadline sporadic task sets scheduled using earliest deadline first preemptive (EDF-P) ($U_{sum} \leq 1.0$) or fixed priority preemptive (FP-P) scheduling with rate-monotonic priority assignment (also called rate-monotonic (RM) scheduling) ($U_{sum} \leq \ln 2 \approx 0.693$) on a single processor. We note that there are also utilization-based schedulability tests that make use of task utilizations in hyperbolic or quadratic forms [23, 25, 32, 31, 51].
- *Speedup Factors* [56, 68] indicate the factor ρ by which the overall speed of a system would need to be increased so that any task set that was schedulable under a scheduling algorithm \mathcal{A} is guaranteed to be schedulable under scheduling algorithm \mathcal{B} . We note that the increase in speed implies that the worst-case execution time (WCET) of each task is reduced by a factor of ρ . Speedup factors illustrate the worst-case performance that one scheduling algorithm can have relative to another. For example, the speedup factor for FP-P scheduling versus EDF-P scheduling is $\frac{1}{\ln 2} \approx 1.44269$ for implicit-deadline task sets on a uniprocessor. Speedup factors can be used to explore sub-optimality with respect to an optimal algorithm, e.g. comparing non-preemptive scheduling algorithms against EDF-P [35] or to make relative comparisons between two non-optimal algorithms, e.g. comparing fixed priority non-preemptive (FP-NP) and EDF non-preemptive (EDF-NP) scheduling [35, 70]. We note that the usefulness of speedup factors is diminished, if no schedulability test is available for the reference algorithm.
- *Capacity Augmentation Bounds* [2, 63, 64] for homogeneous multiprocessor systems quantify scheduling algorithms or schedulability tests via a threshold b , such that the algorithm or test guarantees schedulability of any task set τ provided that $\max_{\tau_i \in \tau} U_i \leq \frac{1}{b}$ and $U_{sum} \leq \frac{M}{b}$, where M is the number of processors, and U_i is the utilization of task τ_i in task set τ with total utilization U_{sum} . The notion of capacity augmentation bounds was formally introduced in 2013 by Li et al. [63] to quantify global-EDF scheduling of task sets where each task can be further characterized using a directed acyclic graph (DAG). In this case, as sub-tasks may execute in parallel, the condition $\max_{\tau_i \in \tau} U_i \leq \frac{1}{b}$

is replaced by $\max_{\tau_i \in \tau} \frac{Critical_i}{T_i} \leq \frac{1}{b}$, where $Critical_i$ is the length of the critical path in the DAG of task τ_i . The capacity augmentation bound of global-EDF was shown to be $\frac{3+\sqrt{5}}{2} \approx 2.6181$ [63, 64] in this case. We note that capacity augmentation bounds differ from speedup factors. For example global-EDF has a speedup factor of 2 in the above case of DAG tasks.

- *Approximation Ratios* for homogeneous multiprocessor systems compare the number of processors needed by (i) scheduling algorithm \mathcal{A} and (ii) an optimal algorithm, to schedule any given task set, as the number of processors required by the optimal algorithm tends to infinity. See [39] for a precise definition. Approximation ratios have been used to characterize partitioned multiprocessor scheduling, with ratios of 1.7 and 1.5 derived for EDF First-Fit [46] and RM Matching [57] respectively. While these approximation ratios can be used to compare different algorithms, their practical use is severely limited. This is because determining the minimum number of processors required by an optimal algorithm is NP-hard, equivalent to the Bin Packing problem.

In this paper, we take a careful look at speedup factors, utilization bounds, and capacity augmentation bounds. Although these three metrics, referred to as *the resource augmentation factors and bounds*, have been widely adopted and accepted by the real-time scheduling research community as the *de facto* standard theoretical tools for assessing scheduling algorithms and schedulability tests, it is not always clear how researchers and designers should view or use such theoretical results. In studying this area, we found a number of surprising results, and related to them, ways in which these metrics can be misinterpreted or misunderstood. The aim of this work is to provide a perspective on the use of these metrics, guide researchers on their meaning and interpretation, and help avoid common pitfalls. Central to this purpose, we seek to answer the following questions:

- (Q1) What are the actual meanings of the resource augmentation factors and bounds and how should they be interpreted?
- (Q2) Algorithm \mathcal{A} has a better resource augmentation factor or bound than Algorithm \mathcal{B} . Does this mean that the performance of \mathcal{A} is *always* better than that of \mathcal{B} ?
- (Q3) Enforcement rules may be employed in algorithm design to achieve good resource augmentation factors or bounds; can they result in design pitfalls that are detrimental to performance?
- (Q4) Are resource augmentation factors meaningful when the reference algorithm is not optimal?
- (Q5) How can we enhance the information provided by resource augmentation factors and bounds to give a broader perspective on performance?

We answer these questions and present our key observations based on several research results for different scheduling problems and models.

2 Task Model and Recap on Uniprocessor Scheduling

In this section, we introduce the well-known sporadic task model, uniprocessor platform, and EDF and fixed priority scheduling algorithms. These are used in many of the subsequent examples. Other examples introduce more complex models e.g. self-suspending tasks, and multiprocessor platforms by building upon the notation, and models described below.

In the sporadic real-time task model, a task set τ comprises n tasks identified by their indices from 1 to n . Each task τ_i has a WCET of C_i , relative deadline D_i , and period or minimum inter-arrival time T_i . Each task τ_i releases a potentially unbounded number of

task instances (or jobs), separated by at least T_i . If $D_i = T_i$ holds for every task, then τ is an *implicit-deadline* task set. Similarly, if $D_i \leq T_i$ holds for every task, then τ is a *constrained-deadline* task set. Finally, *arbitrary-deadline* task sets are the most general and may also have tasks with deadlines that are longer than their periods. The utilization U_i of task τ_i is defined as C_i/T_i . The total utilization U_{sum} of the task set is the sum of the utilizations of its tasks. The task set executes on a platform which has M identical processors, where $M = 1$ for a uniprocessor system. The tasks are assumed to be independent, i.e., they do not share any resources except for the processor, and they do not suspend themselves. Note that we return to multiprocessor scheduling and self-suspending tasks later. Under fixed priority scheduling, each task is assigned a unique static priority, which is inherited by all of its jobs. Without loss of generality, we assume that the task index reflects this priority; thus task τ_1 has the highest priority and task τ_n the lowest.

For the sporadic task model described above, EDF-P is an optimal uniprocessor scheduling algorithm [45]. Although FP-P scheduling is not optimal in the uniprocessor case, it is widely used in practice. With fixed priority scheduling, priority assignment is an important factor in obtaining a schedulable system [42]. In the preemptive case, rate-monotonic (RM) priority assignment is optimal [65] for implicit-deadline task sets, deadline-monotonic (DM) priority assignment is optimal [62] for constrained-deadline task sets, and for arbitrary-deadline task sets, the Optimal Priority Assignment (OPA) algorithm of Audsley et al. [6] may be used to obtain an optimal priority ordering. For ease of reference, we refer to fixed priority scheduling with RM priority assignment as RM scheduling, and similarly with DM priority assignment as DM scheduling. For FP-P scheduling exact schedulability tests have been developed that have pseudo-polynomial-time complexity [61, 55, 6, 60], as well as sufficient schedulability tests, with polynomial-time complexity [65, 23, 34, 25, 32, 31].

A scheduling algorithm is called work-conserving if it never idles the processor when there is a job ready to be executed. Among uniprocessor work-conserving non-preemptive scheduling algorithms for sporadic task sets, EDF-NP scheduling is optimal [47]. In the case of FP-NP scheduling, RM and DM priority assignments are no longer optimal; however, Audsley's OPA algorithm can be used to obtain optimal priority orderings for all three classes of task sets. Exact schedulability tests have been derived for FP-NP scheduling [47, 41], that have exponential time complexity, as well as sufficient schedulability tests that have pseudo-polynomial-time complexity [41, 72], or polynomial-time complexity [34, 5, 70].

3 The Meaning and Interpretation of Augmentation Factors

Utilization bounds, speedup factors, and capacity augmentation bounds have been widely used in the literature to theoretically quantify the performance of scheduling algorithms and schedulability tests. However, due to the way these quantification metrics are defined, they *focus entirely on the worst-case scenario* and quantify it via a single value. This can make the augmentation bounds poorly suited to distinguishing between the performance of different algorithms or tests. Different algorithms or tests may have identical performance in the worst-case, but very different performance across a broad spectrum of other cases. Further, the worst-case scenario may be a specific corner case that is far removed from practical interest. To illustrate these points, we consider uniprocessor FP-P scheduling of implicit-deadline task sets, and then broaden our view to constrained- and arbitrary-deadline task sets and also to non-preemptive scheduling.

In 1973, Liu and Layland [65] presented the seminal utilization bound $\ln 2 \approx 69.3\%$ for RM scheduling of periodic tasks, which directly leads to a speedup factor of $1/\ln(2) \approx 1.44269$

with respect to EDF-P. In 1989 Lehoczky et al. [61] provided a stochastic analysis, showing that the average case is much better than the worst-case behavior, with an average breakdown utilization of 88%. Bini [22] later showed that the optimality degree is even higher (over 90%) when task utilization is uniformly distributed.

A more precise schedulability test that also considers the ratios of task periods was presented by Burchard et al. [26] in 1995. In 1997, Han and Tyan [50] proposed a task transformation technique to convert a set of periodic tasks into a corresponding harmonic task set, such that T_i is an integer multiple of T_j if $T_i \geq T_j$. The utilization bound in [50] analytically dominates those given by Liu and Layland [65] and Burchard et al. [26]. In 1998 Lauzac et al. [59] proposed a utilization bound of $\ln r + 2/r - 1$ based on the ratio r of the minimum task period to the maximum task period if $1 \leq r \leq 2$. When r is 2, this bound is $\ln 2$, the same as the Liu and Layland bound. The harmonic relationship of the task periods was further exploited by Kuo et al. [58] to improve the utilization bound. In 2001, Bini and Buttazzo [21] presented the hyperbolic bound $\prod_{\tau_i \in \tau} (1 + U_i) \leq 2$. More recently, in 2015, Chen et al. [31] developed a utilization-based analysis framework called k2U that can provide hyperbolic bounds almost automatically.

The Liu and Layland utilization bound of $\ln 2$ is independent of the task parameters, while the improvements in the other utilization bounds are based on characteristics of the task set. However, when n is sufficiently large the following worst-case scenario [65] for RM scheduling remains valid for all of the tests mentioned above:

- $T_1 = D_1 = 1, C_1 = (2^{\frac{1}{n}} - 1)$.
- $T_i = T_{i-1} + C_{i-1}, C_i = (2^{\frac{1}{n}} - 1)T_i, \forall i = 2, 3, \dots, n - 1$.
- $T_n = T_{n-1} + C_{n-1}, C_n = (2^{\frac{1}{n}} - 1)T_n + \varepsilon T_n$ where $\varepsilon > 0$ is an arbitrarily small number.

This task set, denoted by τ^{RM} , has a utilization $n(2^{\frac{1}{n}} - 1) + \varepsilon = \ln 2 + \varepsilon$ as $n \rightarrow \infty$ and is not schedulable by RM scheduling since task τ_n misses its deadline. All the schedulability tests mentioned above, including [65, 61, 26, 50, 21, 58, 59, 31], indicate that the above task set is not schedulable using RM scheduling. Since all the schedulability tests mentioned above analytically dominate the Liu and Layland bound of $\ln 2$, we conclude that the speedup factor of any of the above schedulability tests is $\frac{1}{\ln 2}$ with respect to EDF-P. Further, since task set τ^{RM} remains schedulable under EDF-P at speed s as long as $\sum_{\tau_i \in \tau^{RM}} \frac{U_i}{s} \leq 1$, it follows that a lower bound on the speedup factor of RM scheduling is also $\frac{1}{\ln 2}$.

Somewhat surprisingly, we can therefore conclude that every one of the schedulability tests in [55, 6, 65, 61, 26, 50, 21, 58, 59, 31] is a *speedup-optimal* schedulability test for RM scheduling of implicit deadline task sets, with respect to an optimal scheduling algorithm, i.e., EDF-P. In other words all of these tests have the minimum possible speedup factor for the class of scheduling algorithms considered, i.e. fixed priority preemptive scheduling. This is the case despite the fact that only [61, 6, 55] are exact tests. Contrary to fact that all of the tests cited above have the same speedup factor, it is well known that they have very different performance in terms of schedulability, as demonstrated by empirical evaluations using synthetic task sets to examine schedulability by measuring acceptance ratios.

The above discussion illustrates the lack of discrimination between these tests when assessed using speedup factors. This is also apparent when utilization bounds (i.e., $\ln 2$) and capacity augmentation bounds (i.e., $\frac{1}{\ln 2}$) are used for assessment. Moreover, Table 1 presents the speedup factors for DM scheduling in both preemptive and non-preemptive cases. The results in Table 1 show that the exact schedulability tests, with pseudo-polynomial or exponential time complexity, have the same speedup factor as some corresponding linear-time sufficient schedulability tests. Further, although DM is not an optimal priority assignment policy for FP-P scheduling of arbitrary-deadline task sets, or for FP-NP scheduling of any of

■ **Table 1** Speedup factors: lower bounds, upper bounds for linear-time schedulability tests, and upper bounds for pseudo-polynomial / exponential-time schedulability tests.

Constraints	Preemptive			Non-Preemptive		
	lower bound	upper bound (DM, linear)	upper bound (DM, expo.)	lower bound	upper bound (DM, linear)	upper bound (DM, expo.)
implicit-deadline	$1/\ln(2) \approx 1.44269$ [65]			$1/\Omega \approx 1.76322$ [43]	$1/\Omega \approx 1.76322$ [70]	$1/\Omega \approx 1.76322$ [70]
constrained-deadline	$1/\Omega \approx 1.76322$ [44]	$1/\Omega \approx 1.76322$ [31]	$1/\Omega \approx 1.76322$ [44]	$1/\Omega \approx 1.76322$ [43]	$1/\Omega \approx 1.76322$ [70]	$1/\Omega \approx 1.76322$ [70]
arbitrary-deadline	2 [40]	2 [69]	2 [37]	2 [40]	2 [69]	2 [43]

the three classes of task set, it is an *optimal* fixed-priority scheduling strategy with respect to the speedup factors, since the upper bounds on the speedup factors for DM priority assignment [69, 70] are the same as the lower bounds assuming optimal priority assignment and exact tests [40, 43].

The lack of discrimination between different schedulability tests does not just hold for uniprocessor fixed-priority scheduling, but also for multiprocessor partitioned fixed-priority scheduling with constrained-deadline and arbitrary-deadline sporadic task sets, as recently reported by Chen [29]. The analyses in [29] showed that the achieved speedup factors are identical when using exponential-time exact schedulability tests and polynomial-time sufficient schedulability tests. Similarly, for uniprocessor mixed-criticality scheduling, Baruah showed in a series of papers [11, 12, 14] that EDF-VD and its generalization have the same speedup factor for different task models.

► **Observation 1.** *Speedup factors, utilization bounds and capacity augmentation bounds often lack the power to discriminate between the performance of different scheduling algorithms and schedulability tests even though the performance of these algorithms and tests may be very different when viewed from the perspective of empirical evaluation.*

The rationale for Observation 1 is that utilization bounds, capacity augmentation bounds and speedup factors *only* reflect the worst-case corner cases. Having a constant factor or bound, e.g., $\frac{1}{\ln 2}$ or $\ln 2$, does not have any implication with regard to the performance of the algorithm or test in typical cases or in the average cases. Worse, the structure of the corner cases may be easily captured by simple tests, e.g., the hyperbolic bound or the Liu and Layland utilization bound. As other cases, in the broad space of possible task sets, do not contribute to the metric even if the algorithm or the test has relatively poor performance there, then they are simply ignored. Therefore, it is possible that very simple algorithms or very imprecise sufficient schedulability tests may be classified as excellent or even optimal results as long as they can also handle these corner cases well. This explains the results listed in Table 1; even though their performance, in terms of schedulability across a wide range of task sets, is very different.

► **Observation 2.** *Speedup factors, utilization bounds and capacity augmentation bounds should only be considered for their negative implications, since these metrics only provide information on performance in the worst case.*

► **Observation 3.** *Proving that an algorithm or test has the best possible (or optimal) speedup factor or bound for that class of algorithms does not imply that the algorithm or test cannot be substantially improved upon.*

If an algorithm does not have a constant speedup factor, utilization bound, or capacity augmentation bound, it may still perform reasonably well; however, it may also perform terribly in the worst case. A constant bound or factor only ensures that the performance of the algorithm at least reaches some minimum level in the worst case. Even showing that an

algorithm or test has the best possible (or optimal) speedup factor or bound for the studied problem [13, 15] does *not* imply that its performance will necessarily be good in other cases. As a result, as researchers, we should not be satisfied with just deriving algorithms or tests that have optimal speedup factors or bounds. Rather these can be seen as a step towards developing algorithms and tests that, while retaining performance in the worst-case, provide improved performance in practice.

4 Non-dominance Based on Speedup Factors

In this section, we use uniprocessor FP-P scheduling as an example to examine the relationship between dominance results based on speedup factors and utilization bounds, and schedulability as assessed via empirical evaluation in terms of acceptance ratios. To verify the schedulability of a constrained-deadline task τ_k under uniprocessor fixed-priority scheduling, time-demand analysis (TDA) [61] can be applied. That is, task τ_k is schedulable under FP-P scheduling if and only if

$$\exists t | 0 < t \leq D_k, \quad C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t, \quad (1)$$

where $hp(\tau_k)$ is the set of tasks with higher-priority than task τ_k .

Throughout this section, we consider implicit-deadline task sets indexed in the rate-monotonic order i.e., τ_i has higher priority than τ_j if $i \leq j$, and thus $T_i \leq T_{i+1}$. There are many sufficient tests in the literature for testing the schedulability of task τ_k , as explained in Section 3. Here, we consider two sufficient schedulability tests:

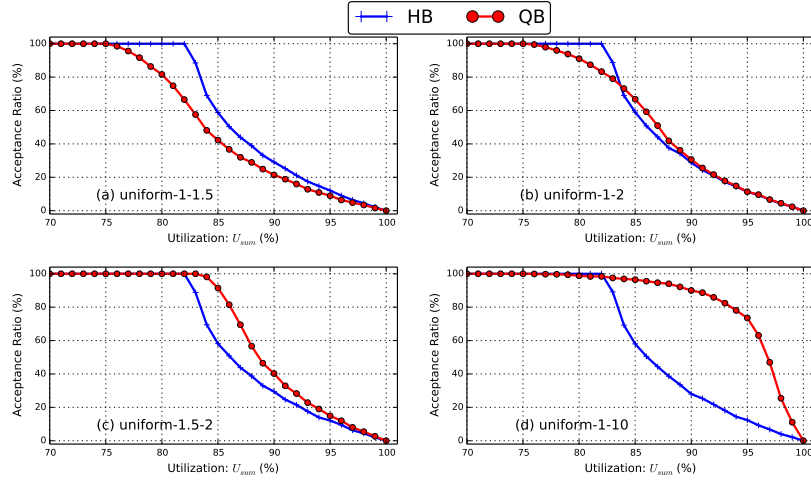
- *Hyperbolic bound* (HB) by Bini and Buttazzo [21]: task τ_k is schedulable by RM if $\prod_{i=1}^k (U_i + 1) \leq 2$. This test has a utilization bound of $\ln 2 \approx 0.693147$, and hence a speedup factor of $\frac{1}{\ln 2} \approx 1.44269$ compared to EDF-P.
- *Quadratic bound* (QB) by Davis and Burns in Equation (26) in [34], and Bini et al. in Equation (11) in [24]: task τ_k is schedulable by RM scheduling if

$$\sum_{i=1}^k U_i + \frac{\sum_{i=1}^{k-1} C_i - \sum_{i=1}^{k-1} U_i C_i}{T_k} \leq 1. \quad (2)$$

This test has a utilization bound of 0.5, and hence a speedup factor of 2 compared to EDF-P. The utilization bound was not explicitly presented in [34, 24]. A tighter quadratic bound that has a utilization bound of $2 - \sqrt{2}$ can be found in more general task models from [1, 51] or can be derived from the k2Q framework [32].

Consider different settings of $\frac{T_1}{T_2}$ with $T_2 > T_1$. Suppose that $U_1 = 0.4$. By adopting the HB, we know that task τ_2 is schedulable under RM if $U_2 \leq 2/1.4 - 1 \approx 42.8\%$. By adopting the QB in (2), we know that task τ_2 is schedulable under RM if $0.4 + U_2 + \frac{0.4T_1 - 0.4^2T_1}{T_2} = 0.4 + U_2 + 0.24\frac{T_1}{T_2} \leq 1$. That is, if $\frac{T_1}{T_2} > 0.715$, then the QB is better; otherwise, the HB is better. As a result, HB and QB are incomparable, neither dominates the other.

According to its speedup factor and utilization bound, HB is better than QB; however, QB is better than HB as long as T_1/T_2 is smaller than ≈ 0.715 when $U_1 = 0.4$. To demonstrate the impact of different distributions of T_1/T_2 , we conducted the following evaluation for implicit-deadline sporadic task sets with *only two* tasks. We considered four different configurations, in each case T_1 was set to 1 and T_2 was chosen from a different uniform distribution: (a) [1, 1.5], (b) [1, 2], (c) [1.5, 2], and (d) [1, 10], thus $T_2 \geq T_1$. In each configuration, for each utilization level, 10,000 task sets were generated as follows:



■ **Figure 1** Adopting the hyperbolic bound (HB) and the quadratic bound (QB) for RM uniprocessor scheduling with $k = 2$ for different uniform distributions of T_2/T_1 .

- For a given target utilization level U_{sum} , U_1 was chosen from a uniform distribution $[0, U_{sum}]$, with U_2 set to $U_{sum} - U_1$.
- T_1 was set to 1, and C_1 to $U_1 T_1$.
- T_2 was chosen from the uniform distribution specified for the configuration, and then C_2 was set to $U_2 T_2$.

The metric used to compare performance was the *acceptance ratio* of the HB and QB tests with respect to a given task set utilization level U_{sum} . The acceptance ratio is the percentage of generated task sets that are schedulable at that utilization level according to the test.

Figure 1 shows the results for the above configurations. The acceptance ratios of QB are highly dependent on the configuration settings for T_2/T_1 , while those for HB are in general independent of these settings. QB is worse than HB if T_2/T_1 is small. Therefore, as shown in Figure 1(a), when T_2 is uniformly distributed in $[1, 1.5] \cdot T_1$, HB is in general better. In contrast, when T_2 is uniformly distributed in $[1.5, 2] \cdot T_1$, QB is always better in the evaluation, as shown in Figure 1(c). The issue with QB is that its worst case happens when T_2 is equal to T_1 . The setting $T_2 \in [1.5, 2] \cdot T_1$ avoids such cases and QB benefits from such a setting. When T_2 is uniformly distributed in $[1, 2] \cdot T_1$ there is no clear winning scheme between QB and HB as shown in Figure 1(b). When U_{sum} is below 84%, HB is better, whereas when $0.85 \leq U_{sum} \leq 0.9$, QB is better. When $U_{sum} \geq 0.9$, the two tests perform almost identically. When T_2 is uniformly distributed in $[1, 10] \cdot T_1$ QB still deems most of the task sets schedulable even when U_{sum} is above 95%, as shown in Figure 1(d); however, there are still a few task sets deemed unschedulable when U_{sum} is below 82%. With this configuration, we can conclude that QB is in general much better than HB, since the schedulable utilization level of QB is much higher than that of HB.

It is clear from the evaluation results for the different configurations shown in Figure 1 that although HB has a superior speedup factor and utilization bound to QB, the relative performance of these two schedulability tests is highly dependent on the configurations used, i.e. the task set parameters.

► **Observation 4.** *A scheduling algorithm or schedulability test with a worse speedup factor or utilization bound may perform (much) better in practice than another algorithm or test*

with a superior speedup factor or utilization bound, dependent on the task set configurations and parameters used. Conclusions on the relative merits of algorithms or tests drawn from speedup factors or utilization bounds can therefore be in direct contradiction with those drawn from empirical performance evaluation.

This apparent contradiction between dominance in terms of speedup factors or utilization bounds and performance observed from evaluation occurs because of the disconnect between the settings for the task set parameters in the two cases. Speedup factors and utilization bounds are dependent solely on corner cases, which may have parameters that rarely occur or are far removed from practical settings. Further examples can be found in the literature:

- For global-RM scheduling, the schedulability test based on the forced forward method [16] and the test by Bertogna and Cirinei [20] both have a speedup factor of 3, compared to an optimal algorithm. When adopting bounded carry-in [48], the schedulability tests based on the k2U and k2Q frameworks by Chen et al. [31, 32] have worse speedup factors, i.e., 3.62143 for k2U and 3.73 for k2Q; however, the evaluation results in [32, 30] show that k2U and k2Q perform much better than the other two tests.
- For implicit-deadline DAG task sets (explained in Sec. 6.1) on M homogeneous processors, Jiang et al. [54] developed a decomposition algorithm which assigns a relative deadline for each DAG subtask. They proved that the capacity augmentation bound of the algorithm is in the range of $[2 - \frac{1}{M}, 4 - \frac{2}{M})$. The capacity augmentation bound under federated scheduling was proved to be $2 - \frac{1}{M}$ by Li et al. [64]. From the capacity augmentation perspective, one may conclude that federated scheduling dominates the decomposition algorithm; however, the experimental results by Jiang et al. [54] showed that their decomposition algorithm outperforms other algorithms in the experimental settings used.

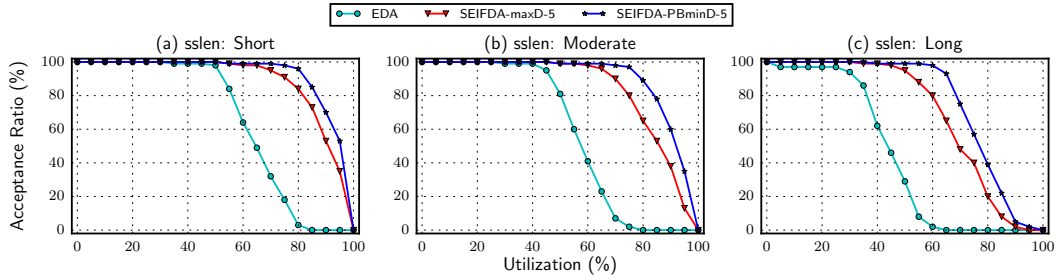
► **Observation 5.** *Identifying regions of dominance in terms of schedulability, between scheduling algorithms and schedulability tests provides valuable information in addition to theoretical analysis in terms of speedup factors or bounds, and empirical evaluations in terms of acceptance ratios.*

5 Speedup Factors Based on Enforced Algorithms

In this section, we consider *enforcements* that are sometimes used when designing scheduling algorithms to simplify the structure of the scheduling problem, and thus allow speedup factors to be derived. If these enforcements are too strong and are applied at an early stage of the algorithm they may make it easy to derive a speedup factor for the scheduling algorithm or schedulability test. However, this may come at the expense of poor performance in practical settings when compared to other algorithms or tests that have worse speedup factors or no speedup factor at all. We illustrate this effect with two examples in the context of scheduling tasks with self-suspensions on uniprocessor and tasks that share resources on multiprocessors.

5.1 One-Segmented Self-Suspension

Equal Deadline Assignment (EDA) by Chen and Liu [33] and Shortest Execution Interval First Deadline Assignment (SEIFDA) by von der Brüggen et al. [71] are two algorithms for scheduling one-segment self-suspending tasks with implicit deadlines on a uniprocessor, using a Fixed-Relative-Deadline (FRD) scheduling strategy [33]. A one-segment self-suspending task τ_i is a periodic task that has two execution segments $C_{i,1}$ and $C_{i,2}$, with total execution time $C_i = C_{i,1} + C_{i,2}$, that are separated by one suspension interval S_i , i.e., τ_i is characterized by $((C_{i,1}, S_i, C_{i,2}), T_i, D_i)$. When a job of task τ_i is released, it has to finish $C_{i,1}$ time units



■ **Figure 2** Comparison of the acceptance ratio for EDA [33] and SEIFDA [71], showing that enforcements in the algorithm design can lead to a huge performance loss.

of computation before it suspends itself for at most S_i time units. After that, the job has to finish $C_{i,2}$ time units of computation before its deadline. A Fixed-Relative-Deadline (FRD) scheduling strategy assigns two relative deadlines $D_{i,1}$ and $D_{i,2}$ to the first and second computation segment respectively. For implicit-deadline task sets, we can always assume that $D_{i,1} + D_{i,2} + S_i = T_i$. The interesting question for FRD is how to assign $D_{i,1}$ and $D_{i,2}$.

An intuitive strategy is the proportional deadline assignment presented by Liu et al. [67], i.e., $D_{i,1} = \frac{C_{i,1}}{C_i} \cdot (T_i - S_i)$ and $D_{i,2} = \frac{C_{i,2}}{C_i} \cdot (T_i - S_i)$. While this assignment policy sounds very reasonable, the speedup factor is unbounded as shown by Chen and Liu [33]. They propose to use EDA instead, i.e., $D_{i,1} = D_{i,2} = (T_i - S_i)/2$, and derive a speedup factor of 2 compared to any other FRD strategy and a speedup factor of 3 compared to an optimal scheduling algorithm for one-segmented self-suspension task sets. The speedup factor of 2 compared to FRD strategies easily follows from the enforcement for the relative deadlines. As both segments have half of the execution interval $T_i - S_i$ to execute the necessary workload, any other FRD scheduling strategy could assign at most twice the relative deadline that EDA assigns to any segment. However, this enforcement is strong as it gives the same deadline to two computation segments even if $C_{i,1} = \varepsilon$ and $C_{i,2} = C_i - \varepsilon$, where ε is a very small positive value, jeopardizing schedulability.

The main problem with EDA is that the assignment of deadlines for one task is independent from the deadline assignment for other tasks. This problem was tackled by von der Brüggen et al. [71] when they proposed the SEIFDA algorithm that assigns the relative deadlines of the tasks in decreasing order of the execution intervals $T_i - S_i$, taking the previously assigned deadlines into account when those deadlines are assigned. For the shorter execution segment, which for ease of explanation we assume is $C_{i,1}$, they calculate the possible values of $D_{i,1} \in [C_{i,1}, T_i - S_i]$ and choose one of these values according to one of three strategies: (i) the minima value (minD), (ii) the maxima value (maxD) or (iii) the minima value larger than $\frac{C_{i,1}}{C_i} \cdot (T_i - S_i)$, i.e., proportionally bounded minD (PBminD). While SEIFDA-maxD dominates EDA and SEIFDA-PBminD dominates proportional assignment, they also show that SEIFDA-minD and SEIFDA-maxD are incomparable. All three assignment strategies have the same speedup factors as EDA while clearly outperforming EDA in terms of acceptance ratios, as can be seen in Figure 2 for SEIFDA-maxD-5 and SEIFDA-PBminD-5. As in [71] an approximated schedulability test is used, the suffix 5 indicates that five periods are calculated exactly before the approximation takes place. The reason for the significantly better performance is that SEIFDA does not enforce the deadlines but chooses them dependent on the other tasks. The results shown in Figure 2 are taken directly from [71] and consider randomly generated task sets with 10 tasks. They are shown for short, medium, and long suspension lengths (sslen) respectively.

5.2 Multiprocessor Scheduling with Resource Sharing

A further example of how enforcement can compromise performance comes from scheduling algorithms for tasks that share resources and execute on a platform with M homogeneous processors. The number of shared resources is denoted by R . Here, we assume that $R \leq M$ and consider the simplified execution structure presented by Andersson and Raravi [4], where each task has only one critical section where it may access shared resources guarded by semaphores. This means, each sporadic task τ_i has three execution segments with WCETs $C_{i,1}^N$, C_i^{Crit} , and $C_{i,2}^N$ representing the part before the critical section, the critical section itself, and the part after the critical section. The WCET of task τ_i is $C_i = C_{i,1}^N + C_i^{Crit} + C_{i,2}^N$. Here, we compare two algorithms for implicit-deadline task sets with known speedup factors.

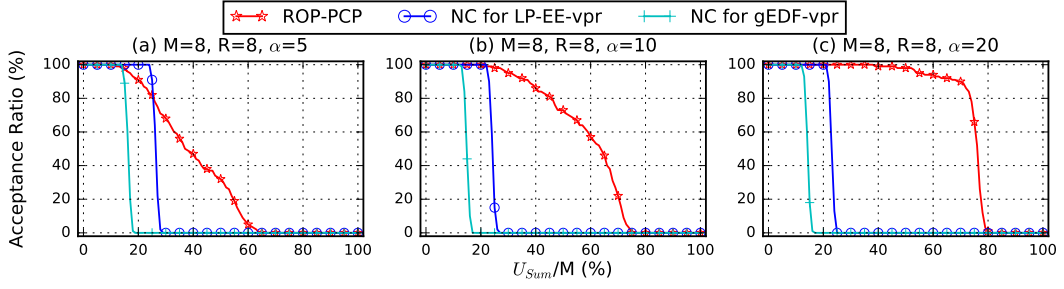
- Andersson and Raravi [4] developed the algorithm *LP-EE-vpr* that has a speedup factor of $4 \cdot (1 + \lceil \frac{R}{M} \rceil) = 8$ in the given setting, since $R \leq M$. Here, we simplify the formulas given in [4] to match the case we analyse. The model in [4] is more general. *LP-EE-vpr* creates M virtual processors with speed $\frac{1}{2}$ to schedule the two non-critical sections of task τ_i and assigns relative deadlines of $\frac{C_{i,1}^N}{C_i} \cdot \frac{T_i}{2}$ and $\frac{C_{i,2}^N}{C_i} \cdot \frac{T_i}{2}$ to them. Non-critical sections are scheduled on virtual processors by using partitioned EDF-P. It also creates M virtual processors with speed $\frac{1}{2}$ to schedule the critical section C_i^{Crit} with a relative deadline $\frac{T_i}{2}$. Critical sections guarded by one semaphore are executed exclusively on one virtual processor using EDF-NP.
- Huang et al. [53] proposed resource-oriented partitioned PCP, called *ROP-PCP*, that has a speedup factor of $11 - \frac{6}{M+1}$. *ROP-PCP* uses two dedicated subsets of the M processors to execute the critical and the non-critical sections individually. Although *ROP-PCP* is applicable to systems in which a task has multiple critical sections, here we limit consideration to one critical section for comparison with *LP-EE-vpr*.

Comparing the speedup factors, one might assume that *LP-EE-vpr* outperforms *ROP-PCP*; however, as *LP-EE-vpr* uses enforcements early in the algorithm to be able to provide the speedup factor while *ROP-PCP* first provides the algorithm and then analyzes the resulting speedup factor, this is not the case. To be precise, the enforcements of *virtualization* at slower speeds in *LP-EE-vpr* and shortened relative deadlines substantially reduce schedulability.

As no schedulability test for *LP-EE-vpr* was provided by Andersson and Raravi [4], we use two necessary conditions for the schedulability of the non-critical execution and the critical section, respectively. This is compared to a sufficient schedulability test for *ROP-PCP* by Huang et al. [53]. For task τ_i , considering the non-critical sections, workload amounting to $C_{i,1}^N + C_{i,2}^N$ must be finished with a relative deadline $\frac{T_i}{2}$ at speed $\frac{1}{2}$. After that, this amount of workload has to be finished every period T_i , leading to the following necessary condition, based on the demand bound function:

$$\forall 0 < t \leq \frac{T_{max}}{2}, \quad \sum_{\tau_i} \left(\left\lceil \frac{t + \frac{T_i}{2}}{T_i} \right\rceil \times 2(C_{i,1}^N + C_{i,2}^N) \right) \leq M \times t, \quad (3)$$

where T_{max} is the maximum among the periods in the task set. The factor 2 in the summation is due to the speed $\frac{1}{2}$ of the virtual processors. We test this necessary condition at each time point $t \in (0, \frac{T_{max}}{2}]$ where the workload changes, i.e., $\forall i \frac{T_i}{2} + \ell \cdot T_i \leq \frac{T_{max}}{2}$ where $\ell \in \mathbb{N}$. For the critical sections, each virtual processor has to be analyzed individually. Let the shared resources be numbered from 1 to R , let $r \in \{1, \dots, R\}$ be one of those resources, and let $\tau^r \subseteq \tau$ be the subset of the tasks that access resource r . We create a necessary condition based on the exact schedulability test for uniprocessor EDP-NP presented by



■ **Figure 3** Comparison of the necessary condition for *LP-EE-vpr* [4], the necessary condition for *gEDF-vpr* [3], and the sufficient schedulability test for *ROP-PCP* [53], showing that enforcements that are made to guarantee a speedup factor can lead to a huge loss in performance.

George et al. [47] for each processor individually. For the virtual processor executing task set τ^r , the demand bound function of the task set is $DBF^r(t) = \sum_{\tau_i \in \tau^r} \left(\left\lfloor \frac{t + \frac{T_i}{2}}{T_i} \right\rfloor \times 2C_i^{crit} \right)$ for any $t > 0$, where the factor 2 is again due to the speed of $\frac{1}{2}$ of the virtual processor. In addition, for each virtual processor the blocking time $B^r(t)$, due to the shared resource r , is $B^r(t) = \max_{\forall \tau_i \in \tau^r, D_i > t} (2 \cdot C_i - \Delta)$, where $\Delta > 0$ but infinitesimally small. The exact schedulability test for uniprocessor EDP-NP by George et al. [47] tests whether (i) $\sum_{\tau_i \in \tau^r} \frac{2C_i^{crit}}{T_i} \leq 1$ and (ii) $DBF^r(t) + B^r(t) \leq t \forall t > 0$. We perform a necessary test by only checking at values of t equal to the deadlines $\frac{T_i}{2}$ of the tasks $\tau_i \in \tau^r$ for each resource r .

A comparison between the sufficient test for *ROP-PCP* and the necessary condition for *LP-EE-vpr* is shown in Figure 3 for a system with 8 processors, 80 tasks, and 1 resource per task that is accessed at most once. We evaluate three ratios $\alpha = 5$, $\alpha = 10$, and $\alpha = 20$ for the length of non-critical sections to the length of critical sections, i.e., $C^{crit} = \frac{1}{1+\alpha} \cdot C_i$ and $C_{i,1}^N + C_{i,2}^N = \frac{\alpha}{1+\alpha} \cdot C_i$. Detailed configurations can be found in [53]. In addition, we also compared to a necessary condition for *gEDF-vpr* by Anderson and Easwaran [3], which is the predecessor of *LP-EE-vpr* and was the first algorithm with a proven speedup factor, i.e., $12(1 + 3R/4M) = 21$ in our setting when $R = M$. Figure 3 shows that the acceptance ratio for *LP-EE-vpr* drops dramatically from a utilization of roughly $M \times 25\%$ and is zero by $M \times 28\%$ utilization in all cases, while the acceptance ratio for *ROP-PCP* decreases more slowly and is still over 50% when the utilization is at $M \times 76\%$ when $\alpha = 20$, $M \times 61\%$ when $\alpha = 10$ and $M \times 36\%$ when $\alpha = 5$. *gEDF-vpr* even drops before $M \times 20\%$ utilization. These results clearly show that the enforcements, which assign stringent relative deadlines to sub-tasks and enforce slow virtual processors in *LP-EE-vpr*, have significant drawbacks in terms of performance even though the speedup factor obtained is better than that for *ROP-PCP*. As a result, both *gEDF-vpr* and *LP-EE-vpr* have barely any chance to schedule task sets with a total utilization above a certain small threshold value, depending on the exact configuration. We adopt necessary conditions for *LP-EE-vpr* and *gEDF-vpr* to show that the performance loss for those methods is due to the early and restrictive enforcements.

► **Observation 6.** *Adding enforcements tailoring the design of a scheduling algorithm or test to facilitate the derivation of a bounded speedup factor can be counterproductive; it may severely compromise performance in practical settings.*

6 Relative Speedup Factors

The speedup factor for a scheduling algorithm or schedulability test is typically given with respect to an optimal algorithm for the same class of problem. For example, we have a speedup factor of $\frac{1}{\ln 2}$ for exact tests for RM scheduling, and also for Liu and Layland's utilization-based test with respect to EDF-P, which is an optimal scheduling algorithm for implicit-deadline sporadic task sets on a uniprocessor. Speedup factors may also be derived relative to another non-optimal algorithm or test. For example the speedup factor of FP-NP scheduling with respect to EDF-NP scheduling on a uniprocessor is 2 for arbitrary-deadline sporadic task sets and ≈ 1.76 for constrained-deadline task sets [43, 40].

It is interesting to consider how bounds on the values of speedup factors can be composed from existing results. Let $S^{\mathcal{A} \rightarrow \mathcal{B}}$ denote the speedup factor of some algorithm or test \mathcal{A} with respect to algorithm or test \mathcal{B} . When one algorithm \mathcal{O} dominates another algorithm \mathcal{X} in terms of schedulability, then we have $S^{\mathcal{O} \rightarrow \mathcal{X}} = 1$ and $S^{\mathcal{X} \rightarrow \mathcal{O}} > 1$. Note this is the case when \mathcal{O} is an optimal algorithm for the class of problem. Further, if two algorithms \mathcal{X} and \mathcal{Y} are incomparable, then we have $S^{\mathcal{X} \rightarrow \mathcal{Y}} > 1$ and $S^{\mathcal{Y} \rightarrow \mathcal{X}} > 1$, i.e. there are non-trivial speedup factors in both directions.

We now consider some simple graphs or chains of speedup factors as examples: Let \mathcal{O} dominate \mathcal{Z} which in turn dominates \mathcal{Y} , then the following relationships hold between the speedup factors: $\max(S^{\mathcal{Y} \rightarrow \mathcal{Z}}, S^{\mathcal{Z} \rightarrow \mathcal{O}}) \leq S^{\mathcal{Y} \rightarrow \mathcal{O}} \leq S^{\mathcal{Y} \rightarrow \mathcal{Z}} \times S^{\mathcal{Z} \rightarrow \mathcal{O}}$. Note the first inequality holds only as a result of the dominance relationships, while the second holds regardless.

Let \mathcal{O} dominate both \mathcal{Z} and \mathcal{X} , and let \mathcal{Z} and \mathcal{X} be incomparable, then the following holds: $S^{\mathcal{Z} \rightarrow \mathcal{X}} \leq S^{\mathcal{Z} \rightarrow \mathcal{O}}$ and $S^{\mathcal{X} \rightarrow \mathcal{Z}} \leq S^{\mathcal{X} \rightarrow \mathcal{O}}$.

Where a speedup factor $S^{\mathcal{X} \rightarrow \mathcal{Z}}$ for some algorithm or test \mathcal{X} is determined relative to some other algorithm \mathcal{Z} that dominates it, but is not optimal for the scheduling problem considered, then great care needs to be taken in the interpretation of the result. In particular, if it turns out that the speedup factor of \mathcal{Z} is unbounded with respect to the optimal algorithm \mathcal{O} , i.e. $S^{\mathcal{Z} \rightarrow \mathcal{O}} = \infty$, then so will be the speedup factor of \mathcal{X} relative to the optimal algorithm, i.e., $S^{\mathcal{X} \rightarrow \mathcal{O}} = \infty$ regardless of the value of $S^{\mathcal{X} \rightarrow \mathcal{Z}}$. Utilization and capacity augmentation bounds are more robust in this respect, since their reference is the capacity of the processor. Below we give two examples based on recent studies that illustrate the pitfalls in using relative speedup factors, rather than those immediately grounded by optimal algorithms.

6.1 Federated Scheduling

To handle sporadic real-time tasks with intra-task parallelism based on DAG structures on multiprocessor platforms, Li et al. [64] and Baruah [8, 9, 10] proposed to use *federated scheduling*, defined as follows: *Either* a task is restricted to execute sequentially on a single processor while sharing this processor with other tasks; *Or*, the task has exclusive access to the processors assigned to it. This strategy was originally proposed in [64] for implicit-deadline task sets, and later extended to constrained-deadline and arbitrary-deadline task sets in [8, 9, 10]. The existing speedup factor results for federated scheduling on M identical processors can be summarized as follows:

- The capacity augmentation bound of the federated scheduling algorithm in [64] for implicit-deadline task sets is 2. Therefore, its speedup factor with respect to *an optimal scheduling algorithm* is also 2.
- The speedup factor of the federated scheduling algorithms in [8, 10] for constrained-deadline task sets is $3 - 1/M$ with respect to *an optimal federated scheduling algorithm*.

- The speedup factor of the federated scheduling algorithms in [9, 10] for arbitrary-deadline task sets is $4 - 2/M$ with respect to *an optimal federated scheduling algorithm*.

These speedup factor values are effectively the same as those for the EDF-FFID partitioning algorithm [17, 18, 19] for sporadic task sets, hence with these results, Baruah made the following conclusions about the algorithms in [8, 9, 10]:

Baruah [8, 9, 10]: *... in terms of the speedup metric, there is no loss in going from the three-parameter sporadic tasks model to the more general sporadic DAG tasks model.*

The resulting speedup factors and conclusions in [8, 9, 10] are however only meaningful if an optimal federated scheduling algorithm also has a bounded speedup factor with respect to an optimal scheduling algorithm for this problem. If federated scheduling itself is not a provably good scheduling strategy in this case, then the constant speedup factors of 3 and 4 become less useful. Unfortunately, the following conclusion was recently presented in [28]:

Chen [28]: *... in terms of the speedup metric with respect to any optimal scheduling algorithm, federated scheduling strategies do not yield any constant speedup factors for constrained-deadline task systems with DAG structures.*

Hence the relative speedup factors derived in [8, 9, 10] cannot be related back to an optimal scheduling algorithm, such a relation is effectively unbounded.

6.2 Uniprocessor Self-Suspension Systems

For the dynamic self-suspension task model, Huang et al. [52] studied how to schedule constrained-deadline self-suspending tasks using fixed-priority scheduling, with a unique priority level assigned to each task. It was shown in [52] that several heuristic priority assignments, including rate-monotonic (RM), deadline-monotonic (DM) and laxity-monotonic (LM), have unbounded speedup factors (Theorem 1 in [52]). Huang et al. [52] proposed using Audsley's optimal priority assignment (OPA) algorithm [7] together with an OPA-compatible schedulability test [38]. They showed that this algorithm has a speedup factor 2 with respect to the *optimal fixed-priority schedule*. Unfortunately, a recent result by Chen [27] shows that the existing scheduling algorithms (that are commonly used) do not yield any constant speedup factors in relation to an optimal algorithm for this problem:

Chen [27]: *For dynamic self-suspending task systems, the speedup factor for any FP preemptive scheduling, compared to the optimal schedules, is not bounded by a constant if the suspension time cannot be reduced by speeding up. Such a statement of unbounded speedup factors can also be proved for earliest-deadline-first (EDF), least-laxity-first (LLF), and earliest-deadline-zero-laxity (EDZL) scheduling algorithms.*

6.3 Remarks on Relative Speedup Factors

Based on the two concrete examples above, we showed that arguments based on relative speedup factors may be undermined and inconclusive if the reference scheduling strategies cannot be related back to optimal algorithms. In both examples, the algorithms proposed (including the reference class of algorithms) actually have unbounded speedup factors with respect to optimal solutions.

► **Observation 7.** *Where relative speedup factors are used in relation to a non-optimal algorithm or class of algorithms, then great care needs to be taken in the interpretation of the results. If the reference algorithm has an unbounded speedup factor with respect to optimal solutions, then the speedup factors may not be that meaningful.*

We note that relative speedup factors can still be meaningful if (i) the reference scheduling strategies are well-accepted, defined, and constrained by the system properties, or (ii) the reference strategy facilitates comparison with an optimal algorithm. For example, considering uniprocessor scheduling, in some cases workload-conserving non-preemptive scheduling may be the only implementation option. For such systems, EDF-NP is an optimal scheduling strategy for sporadic real-time tasks. Therefore, the speedup factors between FP-NP and EDF-NP [43, 40] can be useful. In addition, though neither is an optimal algorithm, they can also be related back to EDF-P [35].

7 Parametric Augmentation Functions

As illustrated in the previous sections, using a single factor or bound to represent the theoretical quantification of the performance of scheduling algorithms or schedulability tests can prove inadequate. It would be better to always show the analytical or theoretical dominance of an algorithm; however, this is not always possible. In this section, we propose a more nuanced way to compare algorithms using a *parametric augmentation function* $\mathcal{A}(\vec{x})$, defined as follows:

- \vec{x} is a vector of user-defined parameters that are of interest to classify different troublesome cases. The elements can be, for example, $\max_{\tau_i \in \tau} U_i$, $\max_{\tau_i \in \tau} \frac{Critical_i}{T_i}$, etc. as used in the definition of capacity augmentation bounds.
- The parametric augmentation function $\mathcal{A}(\vec{x})$ represents the augmentation factor (or the utilization bound) respecting all of the parameters described in \vec{x} .

In this section, we perform theoretical comparisons of two priority assignment schemes for uniprocessor FP-P scheduling, using parametric augmentation functions.

- rate-monotonic (RM): If $T_i < T_j$, then task τ_i has higher-priority than task τ_j ;
- slack-monotonic (SM): If $T_i - C_i < T_j - C_j$, then τ_i has higher-priority than τ_j .

In both cases, ties are broken arbitrarily.

We consider sporadic task sets in which $D_i \geq T_i$ for every task τ_i in the task set. For such a setting, Lehoczky [60] presented utilization bounds for RM scheduling, and the utilization-based scheduling k2U framework by Chen, Huang, and Liu [31] can also be applied. Instead of using the speedup factor (or the utilization bound) for comparing these two algorithms (or the schedulability tests of these two algorithms), we demonstrate why it is more meaningful to compare the algorithms and tests across a broader spectrum.

We assume that the tasks are indexed according to their priority levels, so that τ_1 has the highest priority and τ_n the lowest. We first recall polynomial-time schedulability tests for RM from the literature.

► **Theorem 1** (Chen, Huang, and Liu [31]). *Suppose that $D_k = fT_k$ where f is a positive integer. Task τ_k is schedulable under RM scheduling if*

$$\prod_{i=1}^k (1 + U_i/f) \leq (f + 1)/f. \quad (4)$$

The utilization bound can be further expressed as

$$\sum_{i=1}^{k-1} U_i \leq f \ln \left(\frac{f + 1}{f(1 + U_k/f)} \right) \quad \text{and} \quad U_k \leq 1 \quad (5)$$

Proof. Equation (4) is due to Corollary 3 in [31]. The condition in (5) is obtained with similar techniques to those reported in [31]. ◀

► **Theorem 2** (Lehoczy [60]). *Suppose that $D_k = fT_k$ where f is a positive integer. Task τ_k is schedulable under RM scheduling if*

$$\sum_{i=1}^k U_i \leq \begin{cases} k \left(2^{\frac{1}{k}} - 1 \right) & \text{if } f = 1 \\ f(k-1) \left(\left(\frac{f+1}{f} \right)^{\frac{1}{k-1}} - 1 \right) & \text{if } f = 2, 3, \dots \end{cases} \quad (6)$$

$$\text{When } k \rightarrow \infty, \text{ the utilization bound is } \sum_{i=1}^k U_i \leq f \ln((f+1)/f) \quad (7)$$

Proof. These two conditions come from Equations (3.1) and (3.2) in [60]. ◀

We next derive sufficient schedulability tests for SM.

► **Theorem 3.** *Suppose that $D_k = fT_k$ with $f > 0$. Task τ_k is schedulable under SM scheduling if*

$$\sum_{i=1}^k U_i \leq 1 \quad \text{and} \quad U_k + (1 - U_k + f) \sum_{i=1}^{k-1} U_i \leq f \quad (8)$$

Proof. For the rest of the proof we implicitly consider that $\sum_{i=1}^k U_i \leq 1$. Using the response time upper bounds given by both Bini et al. [25] and Chen et al. [32], for such a task sets, a simple schedulability test for task τ_k is to validate whether

$$\frac{C_k + \sum_{i=1}^{k-1} C_i - \sum_{i=1}^{k-1} U_i C_i}{1 - \sum_{i=1}^{k-1} U_i} \leq D_k.$$

By the definition of SM, we know that

$$T_i - C_i \leq T_k - C_k \Rightarrow T_i(1 - U_i) \leq T_k(1 - U_k) \Rightarrow (1 - U_i) \leq \frac{T_k}{T_i}(1 - U_k).$$

Therefore, we have

$$C_i - C_i U_i = C_i(1 - U_i) \leq C_i \frac{T_k}{T_i}(1 - U_k) = T_k U_i(1 - U_k).$$

As a result, the following inequality holds.

$$\frac{C_k + \sum_{i=1}^{k-1} C_i - \sum_{i=1}^{k-1} U_i C_i}{1 - \sum_{i=1}^{k-1} U_i} \leq \frac{C_k + T_k(1 - U_k) \sum_{i=1}^{k-1} U_i}{1 - \sum_{i=1}^{k-1} U_i}.$$

When $D_k = fT_k$, a sufficient schedulability condition for SM scheduling is given by:

$$\frac{C_k + T_k(1 - U_k) \sum_{i=1}^{k-1} U_i}{1 - \sum_{i=1}^{k-1} U_i} \leq fT_k.$$

By dividing both sides by T_k , this condition can be rewritten as

$$U_k + (1 - U_k) \sum_{i=1}^{k-1} U_i \leq f \left(1 - \sum_{i=1}^{k-1} U_i \right) \Rightarrow U_k + (1 - U_k + f) \sum_{i=1}^{k-1} U_i \leq f. \quad \blacktriangleleft$$

► **Theorem 4.** *Suppose that $D_k = fT_k$ with $f \geq 1$. Task τ_k is schedulable under SM scheduling if*

$$\sum_{i=1}^k U_i \leq \frac{f}{f+1} \quad (9)$$

Proof. This theorem is proved by finding the infimum $\sum_{i=1}^k U_i$ where the condition that $U_k + (1 - U_k + f) \sum_{i=1}^{k-1} U_i > f$ holds using the schedulability condition in Theorem 3. Note that the condition $\sum_{i=1}^k U_i \leq 1$ automatically holds. This is equivalent to the following problem:

$$\text{minimum } x + y \quad \text{s. t. } x + (1 - x + f)y = f \text{ and } 0 \leq x \leq 1,$$

where x is U_k and y is $\sum_{i=1}^{k-1} U_i$. By $x + (1 - x + f)y = f$, we can express y as $(f - x)/(1 - x + f)$. Therefore, $x + y$ is $x + (f - x)/(1 - x + f)$. The first order derivative of $x + y$ is

$$\frac{\partial}{\partial x} \left(x + \frac{f - x}{1 - x + f} \right) = 1 - \frac{1}{(1 - x + f)^2} \geq 0,$$

since $f \geq 1$ and $0 \leq x \leq 1$ in our assumption. Therefore, $x + y$ is minimized when x is 0 and y is $f/(1 + f)$ and thus the theorem follows. ◀

7.1 Comparing RM/SM Based on Traditional Utilization Bounds

Suppose that $f = \min_{\forall \tau_i} (D_i/T_i)$ for the rest of this section, where $f \geq 1$ due to the problem definition. By (7), we can conclude that the utilization bound of RM is $(\lfloor f \rfloor \ln \frac{\lfloor f \rfloor + 1}{\lfloor f \rfloor})$. Similarly, by (9) the utilization bound of SM is $\frac{f}{f+1}$.

► **Lemma 5.** $\frac{x+1}{x+2} - x \ln(1 + \frac{1}{x}) \leq 0$ for any positive integer x .

Proof. Using Taylor series expansion, $\ln(1 + z)$ can be over-approximated by $z - \frac{z^2}{2} + \frac{z^3}{3}$ when $-1 < z < 1$. Therefore, for any $x \geq 2$, we have

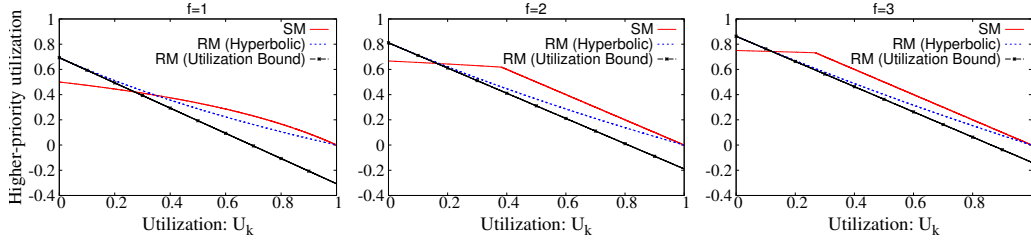
$$\frac{x+1}{x+2} - x \ln(1 + \frac{1}{x}) \leq 1 - \frac{1}{x+2} - x \left(\frac{1}{x} - \frac{1}{2x^2} + \frac{1}{3x^3} \right) = \frac{-1}{x+2} + \frac{1}{2x} - \frac{1}{3x^2} < 0,$$

where the last inequality is due to $\frac{-1}{x+2} + \frac{1}{2x} \leq 0$ for any $x \geq 2$. When x is 1, the expression is $\frac{2}{3} - 2 \ln 1.5 \approx -0.144$, hence the lemma is proved. ◀

► **Theorem 6.** *Suppose that $\min_{\forall \tau_i} (D_i/T_i) = f$ where $f \geq 1$. The utilization bound $(\lfloor f \rfloor \ln \frac{\lfloor f \rfloor + 1}{\lfloor f \rfloor})$ in (7) for RM dominates the utilization bound $\frac{f}{f+1}$ in (9) for SM.*

Proof. Suppose that $\lfloor f \rfloor$ is x . We know that the utilization bound $\frac{f}{f+1}$ in (9) is upper bounded by $\frac{x+1}{x+2}$, hence the theorem follows directly from Lemma 5. ◀

Since the utilization bound for RM is better than the utilization bound for SM for a given $f \geq 1$, the same holds for the speedup factor with respect to EDF-P.



■ **Figure 4** Theoretical comparison of SM and RM. (5) is denoted by RM (Hyperbolic), (7) is denoted by RM (Utilization Bound), and (8) is denoted by SM.

7.2 Comparing RM/SM Based on Parametric-Utilization Factors

The dominance in Theorem 6 is only due to the utilization bounds, or equivalently the augmentation factors under a given $f = \min_{\forall \tau_i} (D_i/T_i)$. If we apply more precise schedulability tests, different conclusions may be drawn.

► **Theorem 7.** Suppose that $f = \min_{\forall \tau_i} (D_i/T_i)$ where $f \geq 1$. SM is a feasible scheduling algorithm for task τ_k if $U_k \geq \frac{1+f-\sqrt{(1+f)^2-4}}{2}$ and $\sum_{i=1}^k U_i \leq 1$.

Proof. The test in (8) for SM checks whether both $\sum_{i=1}^k U_i \leq 1$ and $U_k + (1 - U_k + f) \sum_{i=1}^{k-1} U_i \leq f$ hold. The first condition constrains the utilization of the higher-priority tasks $\sum_{i=1}^{k-1} U_i$ to be at most $1 - U_k$, and the second condition constrains $\sum_{i=1}^{k-1} U_i$ to be at most $\frac{f-U_k}{1+f-U_k}$. Since $\frac{\partial(1-U_k)}{\partial U_k} = -1$ and $\frac{\partial(\frac{f-U_k}{1+f-U_k})}{\partial U_k} = \frac{-1}{(f-U_k+1)^2} \geq -1$ for $f \geq 1$ and $0 \leq U_k \leq 1$, the intersection given by $1 - U_k = \frac{f-U_k}{1+f-U_k}$ defines which of the two conditions in (8) dominates the schedulability test. Let $U^*(f)$ be such an intersection of U_k for a given f . By solving $1 - U_k = \frac{f-U_k}{1+f-U_k}$, we know that $U^*(f)$ is defined as $\frac{1+f-\sqrt{(1+f)^2-4}}{2}$. This means, $U^*(1)$ is 1, $U^*(2)$ is ≈ 0.382 , $U^*(3)$ is ≈ 0.268 , $U^*(4)$ is ≈ 0.209 , $U^*(5)$ is ≈ 0.172 , $U^*(6)$ is ≈ 0.146 , \dots , $U^*(10)$ is ≈ 0.092 , etc.

According to the above analysis, under SM scheduling, when $U_k \geq U^*(f)$, we know that $1 - U_k \leq \frac{f-U_k}{1+f-U_k}$ and therefore the condition $\sum_{i=1}^k U_i \leq 1$ dominates the condition that $U_k + (1 - U_k + f) \sum_{i=1}^{k-1} U_i \leq f$. ◀

The above analysis shows that U_k plays an important role in the schedulability analysis. For ease of comparison, we assume that f is an integer in our discussions for the rest of this section. Figure 4 provides the analytical results by comparing the conditions in (5) denoted by RM (Hyperbolic), (7) denoted by RM (Utilization Bound), and (8) denoted by SM. Figure 4 also shows that $U^*(2)$ is ≈ 0.382 , $U^*(3)$ is ≈ 0.268 , $U^*(4)$ is ≈ 0.209 ; these are the values of U_k at the corner points on the line for SM. Further, the utilization bound of SM when $U_k \geq U^*(f)$ is 100%, as shown by the part of the line for SM with a 45 degree slope. (Since the y-axis measures total utilization for higher priority tasks, a line between (0,1) and (1,0) represents 100% utilization).

The parametric utilization bound shows the importance of including U_k when testing the schedulability of task τ_k , i.e., \vec{x} in the parametric augmentation function should include U_k . As shown in Figure 4, when U_k is small, the schedulability tests for RM in (5) and (7) are better than the test for SM in (8). By contrast, for larger values of U_k the above test for SM is better than the above tests for RM. Hence conclusions on the analytical superiority of these tests for RM and SM can only be drawn when U_k is considered.

7.3 Comparing Schedulability Tests Based on Synthetic Workload

To demonstrate the impact of different distributions of U_k , we conducted the following evaluation for arbitrary-deadline sporadic task sets with k tasks, in which we are only interested in validating the schedulability of task τ_k . For target utilization levels $U_{sum} \geq 0.5$, we explored four different uniform distributions for U_k : (a) $[0, U_{sum}]$, (b) $[0, 0.5]$, (c) $[0, 0.3]$, (d) $[0, 0.1]$. In each case $\sum_{i=1}^{k-1} U_i = U_{sum} - U_k$. For each of the above configurations, we tested $f = 1, f = 2, f = 3, f = 4$. We generated 10000 task sets for each utilization level in each configuration. The metric used to compare performance is the *acceptance ratio* for the tests in (5), (7), and (8).

Figure 5 shows the results for the above configurations. The acceptance ratios of the tests are highly dependent on both f and the distribution of U_k , and hence the configuration used. Note that when the range of values that U_k can take is small, i.e. $[0, 0.1]$ then RM (Hyperbolic) and RM (Utilization Bound) have essentially the same performance, hence the line for RM (Hyperbolic) cannot be seen on the graphs in Figure 5(c), as it is under the line for RM (Utilization Bound).

Thanks to the parametric analysis, showing that U_k plays a significant role, the evaluation settings and configurations also consider such a parameter in the experimental setup. This gives a much more comprehensive picture of the performance of the different algorithms and tests, showing how they vary with critical parameters.

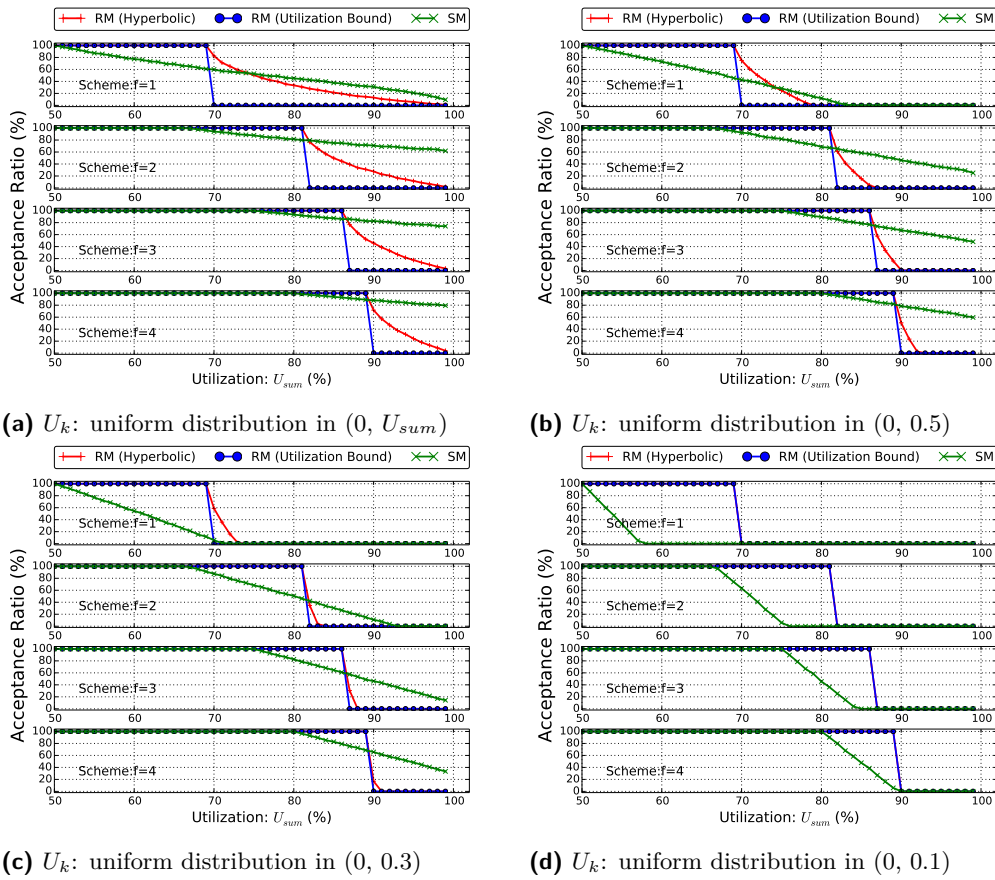
7.4 Remarks on Parametric Augmentation Functions

The concept of parametric augmentation functions can be traced back to Liu and Layland's seminal utilization bound for RM scheduling. This bound is parametric in the number of tasks: $n(2^{\frac{1}{n}} - 1) \geq \ln 2 \approx 0.693$. Similarly, work on speedup factors for DM scheduling of constrained-deadline task sets [44] explored a speedup factor upper bound that is parametric in n . Theorem A.2 in [44] made use of the hyperbolic bound to derive an expression for the speedup factor as a function of n , see also Table 5 and Figures 9 and 10 in Appendix in [44].

In some cases, speedup factors may be unbounded unless some parameter is controlled. For example, Davis et al. [35] used parametric speedup factors to compare non-preemptive uniprocessor scheduling (FP-NP and EDF-NP) against an optimal algorithm (EDF-P). In this case, speedup factors were obtained as a function of C_{max}/D_{min} , where C_{max} is the largest WCET of any task and D_{min} is the smallest deadline. Without using this parameter, the speedup factors $1 + C_{max}/D_{min}$ for EDF-NP and $2 + C_{max}/D_{min}$ for FP-NP would become unbounded, since C_{max}/D_{min} could take an arbitrarily large value. In some practical settings this value can be relatively small, highlighting the utility of such an approach. More recently, parametric augmentation functions have been implicitly used by Liu et al. [66] in the study of EDF-VD scheduling for mixed-criticality systems with degraded quality guarantees. They showed that the augmentation factor depends on two task set dependent constants, denoted by α and λ in [66], with the worst-case speedup factor reducing to $4/3$.

► **Observation 8.** *Parametric augmentation functions can reveal more detailed and nuanced information about the actual performance of schedulability tests or scheduling algorithms across a wide range of parameter values, including practical settings. In some cases parameterized augmentation functions are essential to avoid singularities and hence unbounded results due to unrealistic combinations of parameter values.*

Finally, as shown in Section 7.3 parametric results can be helpful in the design of empirical evaluations and workload generators aimed at providing a comprehensive comparison between different scheduling algorithms and schedulability tests.



■ **Figure 5** Experimental comparison of SM and RM scheduling. (5) is denoted as RM (Hyperbolic), (7) is denoted as RM (Utilization Bound), and (8) is denoted as SM.

8 Summary and Conclusions

In this paper, we studied the use of speedup factors, utilization bounds, and capacity augmentation bounds. Through a series of worked examples and studies, we showed that:

- The metrics often lack the power to discriminate between the performance of different scheduling algorithms and schedulability tests even though their performance may be very different when viewed from the perspective of empirical evaluation.
- The metrics should only be considered for their negative implications, since they only provide information on performance in corner cases.
- Proving that an algorithm or test has an optimal speedup factor or bound for a class of algorithms or problems does not imply that the algorithm or test cannot be substantially improved upon. Further, an algorithm or test with a worse speedup factor or bound may perform much better in practice, and thus conclusions based on speedup factors may directly contradict those drawn from empirical evaluation.
- Identifying regions of dominance, in terms of schedulability, between scheduling algorithms or schedulability tests provides valuable information in addition to speedup factors or bounds, and empirical evaluations in terms of acceptance ratios.
- Adding enforcements tailoring the design of an algorithm or test to facilitate the derivation of a bounded speedup factor can be counterproductive; it may severely compromise performance in practical settings.

- When relative speedup factors are derived in relation to a non-optimal algorithm or class of algorithms, great care needs to be taken in interpreting the results. They can be undermined if the reference algorithm has an unbounded speedup factor with respect to optimal solutions.

We recommend parametric augmentation functions as a theoretical method capable of revealing more detailed and nuanced information about the actual performance of schedulability tests or scheduling algorithms across a wide range of parameter values, including practical settings. We illustrated this technique by deriving such functions for two scheduling algorithms and schedulability tests for uniprocessor scheduling. We also showed that in some cases, parameterized augmentation functions are essential to avoid singularities and hence unbounded results due to unrealistic combinations of parameter values.

Based on our studies of speedup factors, utilization bounds, and capacity augmentation bounds, our considered view is *handle with care*. These theoretical metrics can provide useful information; however, there are also pitfalls in their use. Problems can occur when algorithms are designed with speedup factors in mind, or conclusions are drawn taking a positive perspective solely on the basis of these results. We welcome the extra information that theoretical metrics, particularly parametric augmentation functions, can provide; however, we also recommend that any judgement on the practical utility or otherwise of scheduling algorithms or schedulability tests is backed up by a thorough performance evaluation studying practical settings.

References

- 1 Tarek F. Abdelzaher, Vivek Sharma, and Chenyang Lu. A utilization bound for aperiodic tasks and priority driven scheduling. *IEEE Trans. Computers*, 53(3):334–350, 2004. doi:10.1109/TC.2004.1261839.
- 2 Björn Andersson, Sanjoy K. Baruah, and Jan Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium (RTSS)*, pages 193–202, 2001. doi:10.1109/REAL.2001.990610.
- 3 Björn Andersson and Arvind Easwaran. Provably good multiprocessor scheduling with resource sharing. *Real-Time Systems*, 46(2):153–159, 2010. doi:10.1007/s11241-010-9105-6.
- 4 Björn Andersson and Gurulingesh Raravi. Real-time scheduling with resource sharing on heterogeneous multiprocessors. *Real-Time Systems*, 50(2):270–314, 2014. doi:10.1007/s11241-013-9195-z.
- 5 Björn Andersson and Eduardo Tovar. The utilization bound of non-preemptive rate-monotonic scheduling in controller area networks is 25%. In *IEEE Fourth International Symposium on Industrial Embedded Systems – SIES*, pages 11–18, 2009. doi:10.1109/SIES.2009.5196186.
- 6 N. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993. doi:10.1049/sej.1993.0034.
- 7 Neil C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, May 2001. doi:10.1016/S0020-0190(00)00165-4.
- 8 Sanjoy Baruah. The federated scheduling of constrained-deadline sporadic DAG task systems. In *Proceedings of the Design, Automation & Test in Europe Conference & Exhibition, DATE*, pages 1323–1328, 2015. doi:10.7873/DATE.2015.0200.

- 9 Sanjoy Baruah. Federated scheduling of sporadic DAG task systems. In *IEEE International Parallel and Distributed Processing Symposium, IPDPS*, pages 179–186, 2015. doi:10.1109/IPDPS.2015.33.
- 10 Sanjoy Baruah. The federated scheduling of systems of conditional sporadic DAG tasks. In *Proceedings of the 15th International Conference on Embedded Software (EMSOFT)*, 2015. doi:10.1109/EMSOFT.2015.7318254.
- 11 Sanjoy Baruah. Schedulability analysis for a general model of mixed-criticality recurrent real-time tasks. In *IEEE Real-Time Systems Symposium, RTSS*, pages 25–34, 2016. doi:10.1109/RTSS.2016.012.
- 12 Sanjoy Baruah. Schedulability analysis of mixed-criticality systems with multiple frequency specifications. In *International Conference on Embedded Software, EMSOFT*, pages 24:1–24:10, 2016. doi:10.1145/2968478.2968488.
- 13 Sanjoy K. Baruah, Vincenzo Bonifaci, Gianlorenzo D’Angelo, Haohan Li, Alberto Marchetti-Spaccamela, Suzanne van der Ster, and Leen Stougie. The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems. In *24th Euromicro Conference on Real-Time Systems, ECRTS*, pages 145–154, 2012. doi:10.1109/ECRTS.2012.42.
- 14 Sanjoy K. Baruah, Vincenzo Bonifaci, Gianlorenzo D’Angelo, Haohan Li, Alberto Marchetti-Spaccamela, Suzanne van der Ster, and Leen Stougie. Preemptive uniprocessor scheduling of mixed-criticality sporadic task systems. *J. ACM*, 62(2):14:1–14:33, 2015. doi:10.1145/2699435.
- 15 Sanjoy K. Baruah, Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, and Sebastian Stiller. Implementation of a speedup-optimal global EDF schedulability test. In *21st Euromicro Conference on Real-Time Systems, ECRTS*, pages 259–268, 2009. doi:10.1109/ECRTS.2009.31.
- 16 Sanjoy K. Baruah, Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, and Sebastian Stiller. Improved multiprocessor global schedulability analysis. *Real-Time Systems*, 46(1):3–24, 2010. doi:10.1007/s11241-010-9096-3.
- 17 Sanjoy K. Baruah and Nathan Fisher. The partitioned multiprocessor scheduling of sporadic task systems. In *RTSS*, pages 321–329, 2005. doi:10.1109/RTSS.2005.40.
- 18 Sanjoy K. Baruah and Nathan Fisher. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. *IEEE Trans. Computers*, 55(7):918–923, 2006. doi:10.1109/TC.2006.113.
- 19 Sanjoy K. Baruah and Nathan Wayne Fisher. The partitioned dynamic-priority scheduling of sporadic task systems. *Real-Time Syst.*, 36(3):199–226, August 2007. doi:10.1007/s11241-007-9022-5.
- 20 Marko Bertogna and Michele Cirinei. Response-time analysis for globally scheduled symmetric multiprocessor platforms. In *Real-Time Systems Symposium (RTSS)*, pages 149–160, 2007. doi:10.1109/RTSS.2007.31.
- 21 E. Bini, G. C. Buttazzo, and G. M. Buttazzo. A hyperbolic bound for the rate monotonic algorithm. In *Real-Time Systems, 13th Euromicro Conference on, 2001.*, pages 59–66, 2001. doi:10.1109/EMRTS.2001.934000.
- 22 Enrico Bini and Giorgio C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005. doi:10.1007/s11241-005-0507-9.
- 23 Enrico Bini, Giorgio C. Buttazzo, and Giuseppe M Buttazzo. Rate monotonic analysis: the hyperbolic bound. *Computers, IEEE Transactions on*, 52(7):933–942, 2003. doi:10.1109/TC.2003.1214341.
- 24 Enrico Bini, Thi Huyen Chau Nguyen, Pascal Richard, and Sanjoy K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Trans. Computers*, 58(2):279–286, 2009. doi:10.1109/TC.2008.167.

- 25 Enrico Bini, Andrea Parri, and Giacomo Dossena. A quadratic-time response time upper bound with a tightness property. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 13–22, 2015. doi:10.1109/RTSS.2015.9.
- 26 Almut Burchard, Jörg Liebeherr, Yingfeng Oh, and Sang Hyuk Son. New strategies for assigning real-time tasks to multiprocessor systems. *IEEE Trans. Computers*, 44(12):1429–1442, 1995. doi:10.1109/12.477248.
- 27 Jian-Jia Chen. Computational complexity and speedup factors analyses for self-suspending tasks. In *Real-Time Systems Symposium (RTSS)*, pages 327–338, 2016. doi:10.1109/RTSS.2016.039.
- 28 Jian-Jia Chen. Federated scheduling admits no constant speedup factors for constrained-deadline dag task systems. *Real-Time Systems*, 52(6):833–838, November 2016. doi:10.1007/s11241-016-9255-2.
- 29 Jian-Jia Chen. Partitioned multiprocessor fixed-priority scheduling of sporadic real-time tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 251–261, 2016. doi:10.1109/ECRTS.2016.26.
- 30 Jian-Jia Chen, Wen-Hung Huang, and Cong Liu. Evaluate and compare two utilization-based schedulability-test frameworks for real-time systems. *CoRR*, 2015. URL: <https://arxiv.org/abs/1505.02155>.
- 31 Jian-Jia Chen, Wen-Hung Huang, and Cong Liu. k2U: A general framework from k-point effective schedulability analysis to utilization-based tests. In *Real-Time Systems Symposium (RTSS)*, pages 107–118, 2015. doi:10.1109/RTSS.2015.18.
- 32 Jian-Jia Chen, Wen-Hung Huang, and Cong Liu. k2Q: A quadratic-form response time and schedulability analysis framework for utilization-based analysis. In *2016 IEEE Real-Time Systems Symposium, RTSS*, pages 351–362, 2016. doi:10.1109/RTSS.2016.041.
- 33 Jian-Jia Chen and Cong Liu. Fixed-relative-deadline scheduling of hard real-time tasks with self-suspensions. In *Real-Time Systems Symposium (RTSS)*, pages 149–160, 2014. doi:10.1109/RTSS.2014.31.
- 34 R. I. Davis and A. Burns. Response time upper bounds for fixed priority real-time systems. In *Real-Time Systems Symposium, 2008*, pages 407–418, Nov 2008. doi:10.1109/RTSS.2008.18.
- 35 R. I. Davis, A. Thekkilakattil, O. Gettings, R. Dobrin, and S. Punnekkat. Quantifying the exact sub-optimality of non-preemptive scheduling. In *Real-Time Systems Symposium, 2015 IEEE*, pages 96–106, Dec 2015. doi:10.1109/RTSS.2015.17.
- 36 R.I. Davis. On the evaluation of schedulability tests for real-time scheduling algorithms. In *WATERS*, July 2016. URL: <https://www-users.cs.york.ac.uk/~robdavis/papers/WATERS2016EvalSchedTests.pdf>.
- 37 R.I. Davis, T. Rothvoß, S.K. Baruah, and A. Burns. Quantifying the sub-optimality of uniprocessor fixed priority pre-emptive scheduling for sporadic tasksets with arbitrary deadlines. In *Real-Time and Network Systems (RTNS)*, pages 23–31, 2009. URL: <https://hal.inria.fr/inria-00441952>.
- 38 Robert I. Davis and Alan Burns. Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. *Real-Time Systems*, 47(1):1–40, 2011. doi:10.1007/s11241-010-9106-5.
- 39 Robert I. Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35, 2011. doi:10.1145/1978802.1978814.
- 40 Robert I. Davis, Alan Burns, Sanjoy Baruah, Thomas Rothvoß, Laurent George, and Oliver Gettings. Exact comparison of fixed priority and EDF scheduling based on speedup factors for both pre-emptive and non-pre-emptive paradigms. *Real-Time Systems*, 51(5):566–601, 2015. doi:10.1007/s11241-015-9233-0.

- 41 Robert I. Davis, Alan Burns, Reinder J. Bril, and Johan J. Lukkien. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35(3):239–272, 2007. doi:10.1007/s11241-007-9012-7.
- 42 Robert I. Davis, Liliana Cucu-Grosjean, Marko Bertogna, and Alan Burns. A review of priority assignment in real-time systems. *Journal of systems architecture*, 65:64–82, 2016. doi:10.1016/j.sysarc.2016.04.002.
- 43 Robert I. Davis, Laurent George, and Pierre Courbin. Quantifying the sub-optimality of uniprocessor fixed priority non-pre-emptive scheduling. In *International Conference on Real-Time and Network Systems (RTNS'10)*, 2010. URL: <https://hal.inria.fr/inria-00536363/document>.
- 44 Robert I. Davis, Thomas Rothvoß, Sanjoy K. Baruah, and Alan Burns. Exact quantification of the sub-optimality of uniprocessor fixed priority pre-emptive scheduling. *Real-Time Systems*, 43(3):211–258, 2009. doi:10.1007/s11241-009-9079-4.
- 45 Michael L. Dertouzos. Control robotics: The procedural control of physical processes. In *IFIP Congress'74*, pages 807–813, 1974.
- 46 M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-completeness*. W.H. Freeman and Co., 1979.
- 47 Laurent George, Nicolas Rivierre, and Marco Spuri. Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling. Research report, INRIA, 1996. URL: <https://hal.inria.fr/inria-00073732>.
- 48 Nan Guan, Martin Stigge, Wang Yi, and Ge Yu. New response time bounds for fixed priority multiprocessor scheduling. In *IEEE Real-Time Systems Symposium (RTSS)*, pages 387–397, 2009. doi:10.1109/RTSS.2009.11.
- 49 Nan Guan, Martin Stigge, Wang Yi, and Ge Yu. Fixed-priority multiprocessor scheduling with Liu and Layland's utilization bound. In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 165–174, 2010. doi:10.1109/RTAS.2010.39.
- 50 Ching-Chih Han and Hung-Ying Tyan. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. In *Real-Time Systems Symposium (RTSS)*, pages 36–45, 1997. doi:10.1109/REAL.1997.641267.
- 51 Wen-Hung Huang and Jian-Jia Chen. Techniques for schedulability analysis in mode change systems under fixed-priority scheduling. In *Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 176–186, 2015. doi:10.1109/RTCSA.2015.36.
- 52 Wen-Hung Huang, Jian-Jia Chen, Husheng Zhou, and Cong Liu. PASS: Priority assignment of real-time tasks with dynamic suspending behavior under fixed-priority scheduling. In *Proceedings of the 52nd Annual Design Automation Conference (DAC)*, pages 154:1–154:6, 2015. doi:10.1145/2744769.2744891.
- 53 Wen-Hung Huang, Maolin Yang, and Jian-Jia Chen. Resource-oriented partitioned scheduling in multiprocessor systems: How to partition and how to share? In *Real-Time Systems Symposium (RTSS)*, pages 111–122, 2016. doi:10.1109/RTSS.2016.020.
- 54 Xu Jiang, Xiang Long, Nan Guan, and Han Wan. On the decomposition-based global EDF scheduling of parallel real-time tasks. In *Real-Time Systems Symposium (RTSS)*, pages 237–246, 2016. doi:10.1109/RTSS.2016.031.
- 55 M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, May 1986. doi:10.1093/comjnl/29.5.390.
- 56 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of ACM*, 47(4):617–643, July 2000. doi:10.1145/347476.347479.
- 57 Andreas Karrenbauer and Thomas Rothvoß. A 3/2-approximation algorithm for rate-monotonic multiprocessor scheduling of implicit-deadline tasks. In *International Workshop on Approximation and Online Algorithms*, pages 166–177. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-18318-8_15.

- 58 Tei-Wei Kuo, Li-Pin Chang, Yu-Hua Liu, and Kwei-Jay Lin. Efficient online schedulability tests for real-time systems. *IEEE Trans. Software Eng.*, 29(8):734–751, 2003. doi:10.1109/TSE.2003.1223647.
- 59 Sylvain Lauzac, Rami G. Melhem, and Daniel Mossé. An efficient RMS admission control and its application to multiprocessor scheduling. In *IPPS/SPDP*, pages 511–518, 1998. doi:10.1109/IPPS.1998.669964.
- 60 John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *proceedings Real-Time Systems Symposium (RTSS)*, pages 201–209, Dec 1990. doi:10.1109/REAL.1990.128748.
- 61 John P. Lehoczky, Lui Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *IEEE Real-Time Systems Symposium '89*, pages 166–171, 1989. doi:10.1109/REAL.1989.63567.
- 62 Joseph Y.-T. Leung and Jennifer Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Perform. Eval.*, 2(4):237–250, 1982. doi:10.1016/0166-5316(82)90024-4.
- 63 Jing Li, Kunal Agrawal, Chenyang Lu, and Christopher D. Gill. Analysis of global EDF for parallel tasks. In *Euromicro Conference on Real-Time Systems (ECRTS)*, pages 3–13, 2013. doi:10.1109/ECRTS.2013.12.
- 64 Jing Li, Jian-Jia Chen, Kunal Agrawal, Chenyang Lu, Christopher D. Gill, and Abusayeed Saifullah. Analysis of federated and global scheduling for parallel real-time tasks. In *26th Euromicro Conference on Real-Time Systems, ECRTS*, pages 85–96, 2014. doi:10.1109/ECRTS.2014.23.
- 65 C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973. doi:10.1145/321738.321743.
- 66 Di Liu, Jelena Spasic, Nan Guan, Gang Chen, Songran Liu, Todor Stefanov, and Wang Yi. EDF-VD scheduling of mixed-criticality systems with degraded quality guarantees. In *IEEE Real-Time Systems Symposium, RTSS*, pages 35–46, 2016. doi:10.1109/RTSS.2016.013.
- 67 Wei Liu, Jian-Jia Chen, Anas Toma, Tei-Wei Kuo, and Qingxu Deng. Computation offloading by using timing unreliable components in real-time systems. In *Design Automation Conference (DAC)*, pages 39:1–39:6, 2014. doi:10.1145/2593069.2593109.
- 68 C. Phillips, C. Stein, E. Torng, and J. Wein. Optimal time-critical scheduling via resource augmentation. In *ACM Symposium on Theory of Computing*, pages 140–149, 1997. doi:10.1145/258533.258570.
- 69 Georg von der Brüggen, Jian-Jia Chen, Robert I. Davis, and Wen-Hung Huang. Exact speedup factors for linear-time schedulability tests for fixed-priority preemptive and non-preemptive scheduling. *Information Processing Letters (IPL)*, 2016. doi:10.1016/j.ipl.2016.08.001.
- 70 Georg von der Brüggen, Jian-Jia Chen, and Wen-Hung Huang. Schedulability and optimization analysis for non-preemptive static priority scheduling based on task utilization and blocking factors. In *Euromicro Conference on Real-Time Systems, ECRTS*, pages 90–101, 2015. doi:10.1109/ECRTS.2015.16.
- 71 Georg von der Brüggen, Wen-Hung Huang, Jian-Jia Chen, and Cong Liu. Uniprocessor scheduling strategies for self-suspending task systems. In *International Conference on Real-Time Networks and Systems, RTNS'16*, pages 119–128, 2016. doi:10.1145/2997465.2997497.
- 72 Gang Yao, Giorgio Buttazzo, and Marko Bertogna. Feasibility analysis under fixed priority scheduling with fixed preemption points. In *Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 71–80, 2010. doi:10.1109/RTCSA.2010.40.