Exact Speedup Factors for Linear-Time Schedulability Tests for Fixed-Priority Preemptive and Non-preemptive Scheduling

Georg von der Brüggen[1], Jian-Jia Chen[1],
Robert I. Davis[2], and Wen-Hung Kevin Huang[1]
[1]TU Dortmund University, Germany
[2]INRIA Paris-Rocquencourt, France, and University of York, UK

Abstract

In this paper, we investigate the quality of several linear-time schedulability tests for preemptive and non-preemptive fixed-priority scheduling of uniprocessor systems. The metric used to assess the quality of these tests is the resource augmentation bound commonly known as the processor speedup factor. The speedup factor of a schedulability test corresponds to the smallest factor by which the processing speed of a uniprocessor needs to be increased such that any task set that is feasible under an optimal preemptive (non-preemptive) work-conserving scheduling algorithm is guaranteed to be schedulable with preemptive (non-preemptive) fixed priority scheduling if this scheduling test is used, assuming an appropriate priority assignment. We show the surprising result that the exact speedup factors for Deadline Monotonic (DM) priority assignment combined with sufficient linear-time schedulability tests for implicit-, constrained-, and arbitrary-deadline task sets are the same as those obtained for optimal priority assignment policies combined with *exact* schedulability tests. Thus in terms of the speedup-factors required, there is no penalty in using DM priority assignment and simple linear schedulability tests.

Keywords: Speedup factors; fixed-priority real-time scheduling; non-preemptive and preemptive scheduling; schedulability tests.

1. Introduction

We consider the sporadic task model, in which a task τ_i is characterised by its worst-case execution time (WCET) C_i, its relative deadline D_i, and its period or minimum inter-arrival time T_i. The utilization U_i of task τ_i is defined as C_i/T_i.

For a task set τ, if $D_i \leq T_i$ holds for every task $\tau_i \in \tau$, the task set is said to have *constrained deadlines*. If $D_i = T_i$ holds for every task then τ is an *implicit-deadline* task set. Otherwise τ is an *arbitrary deadline* task set.

Earliest-Deadline-First Preemptive (EDF-P) scheduling is an optimal uniprocessor scheduling algorithm in the sense that if a valid schedule exists for a task set, then the schedule produced by EDF-P will also meet all

deadlines [15]. In the non-preemptive case, no *work-conserving*[1] scheduling policy is optimal, since optimality can require the presence of inserted idle time. Nevertheless, EDF Non-Preemptive (EDF-NP) scheduling is optimal among all work-conserving non-preemptive scheduling algorithms [16].

In this paper we explore both fixed priority preemptive (FP-P) and fixed priority non-preemptive (FP-NP) scheduling, where each task is assigned a unique fixed-priority which is inherited by all of its jobs. Although fixed-priority scheduling policies are not optimal with respect to schedulability, they have been widely adopted by both industry and academia for use in real-time systems due to their low scheduling overheads and simple implementation.

A number of different metrics can be used to quantify the quality of different scheduling algorithms and their schedulability tests. In this paper, we use the resource augmentation bound or *speedup factor* [18]. The speedup factor ρ for FP-P (FP-NP) scheduling is the minimum factor by which the processor speed needs to be increased to ensure that any task set that is schedulable by EDF-P (EDF-NP) is guaranteed (according to some schedulability test) to be schedulable using fixed priorities, assuming an appropriate priority assignment policy. As with prior work in this area, we assume that speeding up the processor by a factor of ρ implies that the WCET of each task τ_i is reduced to C_i/ρ.

We note that while previous work on speedup factors for uniprocessor systems has mainly focused on determining speedup factors assuming exact schedulability tests and priority assignment policies that are optimal in terms of schedulability, it is also interesting to explore how the required speedup factor changes with both the priority assignment policy and the schedulability tests used. In particular, is there a penalty in terms of a larger speedup factor for using a simple priority assignment policy such as Deadline Monotonic (DM) and sufficient schedulability tests that run in linear-time? Answering this question is the focus of the paper.

Contribution: We draw together results, either explicitly or only implicitly shown in previous publications [21, 8, 4, 10, 22] to build an overall picture of the exact speedup factors for linear-time schedulability tests combined with DM priority ordering. We note that since [10] and [22, 8] were developed in parallel it was not possible for the authors of those papers to see the joint implications of their work until they were published. We complete this interesting picture by deriving upper bounds on the speedup factors for preemptive and non-preemptive fixed priority scheduling of arbitrary-deadline task sets assuming linear-time schedulability tests and DM priority assignment.

2. Speedup-Optimal Priority Assignment

With Deadline-Monotonic (DM) priority assignment higher-priorities are assigned to tasks with shorter relative deadlines, with any ties broken arbitrarily to give unique priorities. DM priority assignment is what we refer to in this paper as *schedulability-optimal* for constrained-deadline task sets

[1] A scheduling algorithm is called work-conserving if it never idles the processor when there is a job ready to be executed.

under FP-P scheduling [20]. A priority assignment policy P is referred to as *schedulability-optimal* with respect to a class of task sets (e.g. those with constrained-deadlines) and a fixed priority scheduling algorithm (e.g. FP-P) if all task sets in the class that are schedulable with some other priority assignment policy are also schedulable using priority assignment policy P.

DM priority assignment is not schedulability-optimal for FP-P scheduling of arbitrary-deadline task sets [19] or for FP-NP scheduling of any of the three classes of task sets, i.e., implicit-, constrained-, and arbitrary-deadline [17]. In these cases, Audsley's algorithm [1] can be used to find a schedulability-optimal priority assignment. It is also applicable to FP-P scheduling of systems in the presence of blocking [6].

We now introduce the concept of a *speedup-optimal* priority assignment for fixed priority scheduling. We refer to a priority assignment policy as *speedup-optimal* if the speedup factor that it requires when combined with an exact schedulability test is no larger than the speedup factor required by any other priority assignment policy. Again, this can be applied to different classes of task set and different fixed priority scheduling algorithms (FP-P or FP-NP). The optimality of a priority assignment policy with respect to schedulability implies that it is also speedup-optimal for the same class of task sets; however, non-optimality with respect to schedulability does not necessarily imply non-optimality with respect to the speedup factor required. This leads to the following interesting observation.

The recent work of Davis et al. [10] and von der Brüggen et al. [22] shows that DM priority assignment is a *speedup-optimal* priority assignment policy for fixed priority scheduling in *all* of its forms, i.e. FP-P and FP-NP scheduling of implicit-, constrained-, and arbitrary deadline tasks sets.

Since DM priority assignment is schedulability-optimal for implicit and constrained deadline task sets under FP-P scheduling, this implies that it is also speedup-optimal in those cases. Somewhat surprisingly, Deadline Monotonic priority assignment is also speedup-optimal for both FP-P scheduling (Theorem 1 in [10]) and FP-NP scheduling (Theorem 7 in [10]) of task sets with arbitrary deadlines. These theorems show that the exact speedup factors are unchanged when DM priority assignment is used in place of Audsley's algorithm [1]. Similarly, DM priority assignment is also speedup-optimal for task sets with implicit or constrained deadlines under FP-NP scheduling, since the upper bounds on the speedup factors proven for DM priority assignment in those cases [22] match the lower bounds determined assuming Audsley's algorithm [11, 10]. This implies that the bounds are exact for both DM priority assignment and schedulability-optimal priority assignment using Audsley's algorithm.

In the remainder of the paper, we show that the upper bounds on the speedup factors for fixed priority scheduling using DM priority assignment are the same as the lower bounds (proven for exact tests and schedulability optimal priority assignment policies) even when simple linear-time schedulability tests are used. Hence we show that similar to the simplification from schedulability-optimal priority assignment to DM priority assignment, the simplification from exact pseudo-polynomial or exponential schedulability tests to simple linear-time sufficient tests also brings no additional penalty in terms of the speedup factors required.

3. Linear-Time Schedulability Tests

We focus only on linear-time schedulability tests for fixed priority scheduling with DM priority assignment (for brevity referred to as *DM scheduling*). We assume that there are n sporadic tasks and that those tasks are indexed in order of non-decreasing relative deadlines, i.e., $D_1 \leq D_2 \leq D_3 \leq \cdots \leq D_n$. Suppose that the first $k-1$ tasks are already verified to be schedulable under DM scheduling. We focus on testing the schedulability of task τ_k in linear time. We note that all of the tests *for fixed-priority scheduling* in this section can be efficiently implemented by using appropriate data structures to amortize the overall time complexity to $O(n)$ for testing *all* n tasks. This is discussed in the appendix.

3.1. Speedup Factors

We first introduce the technique that we will use to prove the speedup factors in the case of arbitrary-deadline task sets.

Baruah et al. [2] derived an exact test for arbitrary deadline task sets scheduled under EDF-P based on the concept of the *demand bound function*. We refer to

$$dbf_i(t) \;=\; max\left\{0, \left\lfloor \frac{t-D_i}{T_i} \right\rfloor + 1\right\} C_i \tag{1}$$

as the demand bound function of task τ_i for a time interval of length t. Baruah et al. [2] showed that a task set is schedulable under EDF-P *if and only if* 1) the total utilisation is no greater than 1 (i.e. $\sum_{i=1}^{n} U_i \leq 1$) and 2) for any time interval of length t, the total processor demand $dbf(t)$ requested by the task set is no greater than the length of the interval.

$$dbf(t) = \sum_{i=1}^{n} dbf_i(t) \leq t \tag{2}$$

George et al. [17] extended this demand bound test to the non-preemptive case, introducing a blocking factor $B(t)$. Hence an arbitrary deadline task set is schedulable under EDF-NP *if and only if* 1) the task set utilisation is no greater than 1, and 2) for any interval of length $t \geq D_1$ (where D_1 is the smallest task deadline)

$$dbf(t) + B(t) \leq t \tag{3}$$

Here, the maximum blocking time $B(t)$ for a time interval of length t is defined as

$$B(t) = \max_{\forall i, D_i > t} (C_i - \Delta) \tag{4}$$

with $\Delta > 0$ but infinitesimally small [10].

In contrast, the maximum blocking time B_k for task τ_k under non-preemptive DM scheduling is defined as

$$B_k = \max_{\forall \tau_i \in lp(k)} (C_i - \Delta) \tag{5}$$

where $lp(k)$ is the set of tasks with priorities lower than that of task τ_k. Note, as the tasks are in DM priority order, none of these lower priority tasks can have a deadline that is shorter than that of task τ_k. Assuming DM priority order, then

it follows that $B(D_k) \geq B_k$ with the two quantities being equal unless there is some lower priority task with the same deadline and a long execution time.

In the following, we assume the more general form (3) of the demand bound test for both preemptive and non-preemptive scheduling, assuming in the preemptive case that $B(D_k) = B_k = 0$. If we can prove that any task set ϕ that is *unschedulable* according to some schedulability test for FP-P (FP-NP) scheduling is also unschedulable under EDF-P (EDF-NP) on a processor whose speed has been reduced, i.e., scaled by a factor of $1/\rho$, then that suffices to show that ρ is an upper bound on the speedup factor for the test. Therefore, to prove an upper bound on the speedup factor ρ for a linear-time schedulability test for preemptive or non-preemptive DM scheduling, we only need to show that failure of the schedulability test implies either

$$\sum_{i=1}^{n} U_i > \frac{1}{\rho} \tag{6}$$

or

$$\frac{dbf(D_k) \ + \ B(D_k)}{D_k} > \frac{1}{\rho} \tag{7}$$

which in turn implies that the task set cannot be scheduled under EDF-P (EDF-NP) on a processor of speed $1/\rho$. Assuming DM priority order, we observe the following relationship which we use later.

$$\frac{dbf(D_k) \ + \ B(D_k)}{D_k} \geq \frac{C_k + \sum_{i=1}^{k-1} C_i + B_k}{D_k} \tag{8}$$

We note that any lower bound speedup factor for FP-P (FP-NP) versus EDF-P (EDF-NP) proven for an exact test for FP-P (FP-NP) is also valid for any sufficient test for the same scheduling algorithm. This is the case because by definition, an exact test dominates any sufficient test for the same algorithm, since there are no task sets which are deemed schedulable by the sufficient test that are not also schedulable according to the exact test. In the following, we provide a set of speedup factor upper bounds for simple linear-time sufficient schedulability tests. Since these upper bounds are the same as the lower bounds (and exact values) previously published for exact tests, the speedup factors are also tight (exact) for the linear-time tests.

3.2. Preemptive DM Scheduling

Implicit-Deadlines. The well-known Liu and Layland bound [21] provides a schedulability test by verifying that

$$\sum_{i=1}^{k} U_i \leq k(2^{1/k} - 1) \tag{9}$$

Alternatively, the hyperbolic bound by Bini et al. [3] can be used to verify whether

$$\prod_{i=1}^{k} (1 + U_i) \leq 2 \tag{10}$$

Since EDF-P can schedule all implicit-deadline tasks sets with utilisation not exceeding 1 [21], both (9) and (10) lead directly to linear-time schedulability tests with an exact speedup factor of $1/\ln 2$.

Constrained-Deadlines. The linear-time test provided by Chen et al. in Section 5.1 of [8] can be used to determine schedulability by verifying the hyperbolic form

$$\left(\frac{C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i}{D_k} + 1\right) \prod_{\tau_i \in hp_1(\tau_k)} (1 + U_i) \leq 2 \tag{11}$$

where $hp_1(k)$ is the set of tasks with higher-priority than τ_k that have periods less than D_k, and $hp_2(\tau_k)$ is the set of tasks with higher-priority than τ_k that have periods greater than or equal to D_k. Theorem 2 in [8] proves that the upper bound speedup factor for this test is $1/\Omega \approx 1.76322$. Since this is the same as the lower bound proven in [13] it is also exact.

Arbitrary-Deadline. The linear-time test we use in this case is a weaker form of the response time upper bounds provided by Davis and Burns in Eq. (26) in [9] and Bini et al. in Eq. (11) in [4]. Schedulability can be verified by checking if

$$D_k \geq \frac{C_k + \sum_{i=1}^{k-1} C_i}{1 - \sum_{i=1}^{k-1} U_i} \tag{12}$$

If task τ_k cannot pass the above linear-time test, then re-arranging (12), it means that

$$1 < \frac{C_k + \sum_{i=1}^{k-1} C_i}{D_k} + \sum_{i=1}^{k-1} U_i \tag{13}$$

Therefore, it must be the case that either or both of (i) $\frac{C_k + \sum_{i=1}^{k-1} C_i}{D_k} > 0.5$ and (ii) $\sum_{i=1}^{k-1} U_i > 0.5$ hold. From (6), (7), and (8) this implies that the task set cannot be scheduled on a processor of speed 0.5 under EDF-P, and hence an upper bound on the speedup factor is 2. Since this is the same as the lower bound proven in [10] it is also exact.

3.3. Non-Preemptive DM Scheduling

Implicit-Deadlines and Constrained-Deadlines. The linear-time schedulability test is provided by von der Brüggen et al. in Section 4 in [22]. Schedulability is determined by verifying whether

$$\left(\frac{B_k + C_k + \sum_{\tau_i \in hp_2(k)} C_i}{D_k} + 1\right) \prod_{\tau_i \in hp_1(k)} (1 + U_i) \leq 2 \tag{14}$$

where $hp_1(k)$ is the set of tasks with higher-priority than τ_k that have periods less than D_k, and $hp_2(\tau_k)$ is the set of tasks with higher-priority than τ_k that have periods greater than or equal to D_k. Theorem 5 in [22] proves that the upper bound speedup factor for the above test is $1/\Omega \approx 1.76322$, compared to EDF-NP. Since this is the same as the lower bound proven in [11] and [10] it is also exact.

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	$1/ln(2) \approx 1.44269$ [21]			$1/\Omega \approx$ 1.76322 [11]	$1/\Omega \approx$ 1.76322 [22]	$1/\Omega \approx$ 1.76322 [22]
constrained-deadline	$1/\Omega \approx$ 1.76322 [13]	$1/\Omega \approx$ 1.76322 [8]	$1/\Omega \approx$ 1.76322 [13]	$1/\Omega \approx$ 1.76322 [11]	$1/\Omega \approx$ 1.76322 [22]	$1/\Omega \approx$ 1.76322 [22]
arbitrary-deadline	2 [10]	2 (this paper)	2 [12]	2 [10]	2 (this paper)	2 [11]

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

Arbitrary-Deadlines. The linear-time test we use is a weaker form of the response time upper bounds provided by Davis and Burns in Eq. (33) in [9] and Bini et al. in Eq. (14) in [4]. Here, schedulability may be verified by checking:

$$D_k \geq \frac{B_k + C_k + \sum_{i=1}^{k-1} C_i}{1 - \sum_{i=1}^{k-1} U_i} \tag{15}$$

Therefore, if task τ_k cannot pass the above linear-time test, it means that

$$1 < \frac{B_k + C_k + \sum_{i=1}^{k-1} C_i}{D_k} + \sum_{i=1}^{k-1} U_i \tag{16}$$

Following the same logic as before, it must be the case that either (i) $\frac{B_k+C_k+\sum_{i=1}^{k-1} C_i}{D_k} > 0.5$ or (ii) $\sum_{i=1}^{k-1} U_i > 0.5$ or both (i) and (ii) hold. From (6), (7), and (8) this implies that the task set cannot be schedulable on a processor of speed 0.5 under EDF-NP, and hence an upper bound on the speedup factor is 2. Since this is the same as the lower bound proven in [10] it is also exact.

4. Summary and Conclusions

In this paper, we showed the somewhat surprising result that Deadline Monotonic (DM) priority assignment combined with simple linear-time sufficient schedulability tests for fixed priority scheduling results in *exact* speedup factors with respect to EDF that are identical to those obtained for optimal priority assignment and exact pseudo-polynomial- or exponential-time schedulability tests. This result holds across all three classes of task set (implicit-, constrained- and arbitrary-deadline) as well as for both preemptive and non-preemptive scheduling (FP-P and FP-NP).

We note that there are other linear-time schedulability tests for fixed-priority scheduling. Several of which are superior to the very simple linear-time tests we used in this paper. These include the more precise quadratic bound by Bini and Parri [5] and Chen et al. [7] for arbitrary-deadline task sets and the more precise hyperbolic bound for constrained-deadline task sets under non-preemptive fixed-priority scheduling by von der Brüggen et al. [22]. Since these tests dominate the ones used in this paper, they also have speedup factors that are the same as those for exact tests.

Table 1 summarises the results, giving the speedup factor lower bounds (previously proven for exact tests and thus applying to all valid sufficient tests), the upper bounds for linear-time tests assuming DM priority

assignment (shown in this paper) and the upper bounds for exact pseudo-polynomial or exponential time schedulability tests assuming DM priority assignment. Note the values for these upper bounds are the same as those for exact tests using Audsley's schedulability-optimal priority assignment algorithm [1], which are not separately shown.

Finally, we note that the exact speedup factors for arbitrary deadline task sets derived for the comparisons between FP-NP and FP-P and between FP-NP and EDF-P by Davis et al. in [14] also hold in the case where (non-optimal) DM priority assignment and a linear time sufficient schedulability test are used for FP-NP. Thus DM priority assignment combined with a linear-time sufficient test is also speedup optimal when comparing non-preemptive versus preemptive scheduling paradigms.

Acknowledgments

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

References

References

[1] N. C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, 79(1):39–44, May 2001.

[2] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *proceedings Real-Time Systems Symposium (RTSS)*, pages 182–190, Dec 1990.

[3] E. Bini, G. Buttazzo, and G. Buttazzo. Rate monotonic analysis: the hyperbolic bound. *Computers, IEEE Transactions on*, 52(7):933–942, Jul 2003.

[4] E. Bini, T. H. C. Nguyen, P. Richard, and S. K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. *IEEE Trans. Computers*, 58(2):279–286, 2009.

[5] E. Bini, A. Parri, and G. Dossena. A quadratic-time response time upper bound with a tightness property. In *IEEE Real-Time Systems Symposium (RTSS)*, 2015.

[6] K. Bletsas and N. Audsley. Optimal priority assignment in the presence of blocking. *Inf. Process. Lett.*, 99(3):83–86, Aug. 2006.

[7] J.-J. Chen, W.-H. Huang, and C. Liu. k^2Q: A quadratic-form response time and schedulability analysis framework for utilization-based analysis. *Computing Research Repository (CoRR)*, abs/1505.03883, 2015.

[8] J.-J. Chen, W.-H. Huang, and C. Liu. k2U: A general framework from k-point effective schedulability analysis to utilization-based tests. In *Real-Time Systems Symposium (RTSS)*, 2015.

[9] 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.

[10] R. I. Davis, A. Burns, S. Baruah, T. Rothvoß, L. George, and O. 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.

[11] R. I. Davis, L. George, and P. 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.

[12] R. I. Davis, T. Rothvoß, S. Baruah, and A. Burns. Quantifying the sub-optimality of uniprocessor fixed priority pre-emptive scheduling for sporadic tasksets with arbitrary deadlines. In *Real-Time Networks and Systems Conference*, 2009.

[13] R. I. Davis, T. Rothvoß, S. K. Baruah, and A. Burns. Exact quantification of the suboptimality of uniprocessor fixed priority pre-emptive scheduling. *Real-Time Systems*, 43, 2009.

[14] 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.

[15] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *IFIP Congress'74*, pages 807–813, 1974.

[16] L. George, P. Muhlethaler, and N. Rivierre. Optimality and non-preemptive real-time scheduling revisited. Research Report RR-2516, INRIA, 1995. https://hal.inria.fr/inria-00074162.

[17] L. George, N. Rivierre, and M. Spuri. Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling. Research report, INRIA, 1996.

[18] B. Kalyanasundaram and K. Pruhs. Speed is as powerful as clairvoyance. *Journal of ACM*, 47(4):617–643, July 2000.

[19] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *proceedings Real-Time Systems Symposium (RTSS)*, pages 201–209, Dec 1990.

[20] J. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic real-time tasks. *Performance Evaluation*, 2:237–250, 1982.

[21] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[22] G. von der Bruggen, J.-J. Chen, and W.-H. Huang. Schedulability and optimization analysis for non-preemptive static priority scheduling based on task utilization and blocking factors. In *27th Euromicro Conference on Real-Time Systems, ECRTS*, pages 90–101, 2015.

Appendix: Linear-Time Schedulability Tests

Here, we explain how (9), (10), (11), (12), (14), and (15) can be efficiently implemented by using appropriate data structures to amortize the overall time complexity to $O(n)$ for testing *all* n tasks, i.e., $k = 1, 2, \ldots, n$. The blocking time for all tasks can be computed in $O(n)$ time if we compute it starting with the lowest priority task and save the values in an array. The linear-time complexity for (9), (10), (12), (15) is obvious since we just need to incrementally include a constant

number of additional terms when moving from testing task τ_k to task τ_{k+1}. For example, in (15), we simply have to store $\sum_{i=1}^{k-1} U_i$ and $\sum_{i=1}^{k-1} C_i$ in two variables and increase them by U_k and C_k, respectively, when we test τ_{k+1}.

The tests in (11) and (14) are also of linear time complexity, provided that the orders of tasks by their periods and their relative deadlines (from the smallest to the largest) are given. The details were already discussed in [22]. Here, we summarize the reasons. For FP-NP and FP-P, when we move from τ_k to τ_{k+1}, we want to analyze the changes in $hp_1(k)$ to $hp_1(k+1)$ and $hp_2(k)$ to $hp_2(k+1)$ for this step. As the relative deadline is increasing with the task priority, no task can ever move from $hp_1(k)$ to $hp_2(k+1)$. Assume τ_k is placed in $hp_2(k)$. Then all tasks $\tau_i \in hp_2(k)$ with $D_k \leq T_i < D_{k+1}$ will be moved to $hp_1(k+1)$. This can be determined in $O(1)$ time for each task (in an amortized manner) by using a proper data structure, e.g., another array sorted according to the periods and some variables to store the internal information.