Improved Low Time-Complexity Schedulability Test for Nonpreemptive EDF on a Multiprocessor

Seongtae Lee, Sanghyeok Park, and Jinkyu Lee, *Senior Member, IEEE*

***Abstract*—In real-time embedded systems, nonpreemptive earliest deadline first (NP-EDF) is one of the most popular scheduling algorithms to offer timing guarantees of a set of nonpreemptive real-time jobs (tasks). While most existing schedulability tests for NP-EDF on a multiprocessor platform have paid attention to improving schedulability performance at the expense of increasing time complexity, only a few studies can be used for the situation where low time complexity is critical. In this letter, based on an existing low time-complexity schedulability test for NP-EDF, we develop schedulability tests that improve schedulability performance but maintain low time complexity. Our experiments show that the proposed schedulability tests improve schedulability performance up to 592.4%, compared to the existing low time-complexity one, and they can find some additional task sets schedulable by NP-EDF, which cannot be covered by existing high time-complexity NP-EDF schedulability tests.**

***Index Terms*—Low time-complexity schedulability test, nonpreemptive earliest deadline first (NP-EDF), real-time multiprocessor scheduling.**

I. Introduction

It is the most fundamental goal in real-time embedded systems to offer timing guarantees of a set of tasks/jobs subject to timing constraints, and nonpreemptive earliest deadline first (NP-EDF) is a popular scheduling algorithm for a set of tasks/jobs, each of which is inherently nonpreemptive or incurs large overheads for preemption or migration. Since most existing schedulability tests for NP-EDF have tried to improve schedulability performance at the expense of increasing time-complexity [1]–[4], there have been only a few studies that can be used for the situation where low time-complexity matters critically [5]. For example, a real-time system where tasks/jobs are dynamically in and out, a runtime admission control is necessary, which requires a quick decision of whether a newly added task/job can be accommodated without compromising the schedulability of existing jobs/tasks and itself.

In this letter, based on an existing low time-complexity schedulability test for NP-EDF in [5], we develop two versions of schedulability tests, which maintain low time complexity but improve schedulability performance. To this end, we first investigate the effect of how a higher priority job of interest is blocked by its lower priority jobs, and incorporate the effect into the schedulability test, yielding an enhanced schedulability test with its correctness proof. Second, exploiting an interesting property from [6] (that preserves timing guarantees in case of excluding a pair of a task/job and a processor), we develop another schedulability test that further improves the proposed enhanced schedulability test. While existing studies focus on a task set that invokes a series of jobs periodically/sporadically, we make the proposed schedulability tests operate for not only the task model but also the job model in which jobs are dynamically in and out. Via experiments, we demonstrate that the proposed schedulability tests improve schedulability performance up to 592.4%, compared to the existing low time-complexity schedulability test in [5]. Also, the proposed tests can find some additional task sets schedulable by NP-EDF, which cannot be deemed schedulable by existing high time-complexity NP-EDF schedulability tests.

II. System Model and Background

In this letter, we consider a job model [7], which is a generalization of a sporadic real-time task model. A job J_i is specified as (r_i, d_i, C_i), where r_i is the release time, d_i is the absolute deadline, and C_i is the worst case execution time. Let D_i denote the relative deadline of J_i, meaning that $D_i = d_i - r_i$. Let $\mathcal{J}$ denote a set of all jobs $\{J_i\}$ in the system, and $\mathcal{J}(t)$ denote a set of jobs $\{J_i\}$ whose execution window $[r_i, d_i)$ subsumes t. Without loss of generality, we assume that the job index is sorted by its deadline, i.e., J_i is defined as the job with the ith earliest deadline among all jobs in $\mathcal{J}$. We consider a multiprocessor platform that consists of m identical processors. We assume that a job cannot be executed on more than one processor at any time slot.

In this letter, we focus on the global, work-conserving NP-EDF algorithm, under which: 1) a job with an earlier deadline has a higher priority than a job with a later deadline; 2) each job can start its execution on any processor; and 3) every job with remaining execution should be executed if there exists an idle processor. Different from preemptive EDF, a job cannot preempt any currently executing job, and therefore, it is possible for a lower priority job to block a higher priority job if the lower priority job starts its execution before the higher priority job's release. To ease of presentation, let LHS and RHS denote the left-hand side and right-hand side, respectively.

In this letter, we target a low time-complexity NP-EDF schedulability test (denoted by Bar) as follows.

Manuscript received July 12, 2021; revised November 3, 2021; accepted December 6, 2021. Date of publication December 9, 2021; date of current version May 19, 2022. This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MSIT) under Grant 2021R1A2B5B02001758 and Grant 2021K2A9A1A01101570. This manuscript was recommended for publication by P. S. Roop. *(Corresponding author: Jinkyu Lee.)*

The authors are with the Department of Computer Science and Engineering, Sungkyunkwan University, Suwon 16419, Republic of Korea (e-mail: yuns0509@g.skku.edu; tkdgur5273@skku.edu; jinkyu.lee@skku.edu).

Digital Object Identifier 10.1109/LES.2021.3133901

Lemma 1 (Job Set Version of [5, Th. 1]): A job set $\mathcal{J}$ is schedulable by NP-EDF on m processors, if for every $t \in \{r_j\}_{J_j \in \mathcal{J}}$, the following inequality and $C_i \leq D_i - \max_{J_j \in \mathcal{J}} C_j$ hold for every $J_i \in \mathcal{J}(t)$:

$$\sum_{J_i \in \mathcal{J}(t)} V_i \leq m - (m-1) \times \max_{J_i \in \mathcal{J}(t)} V_i \tag{1}$$

where $V_i = C_i/(D_i - \max_{J_j \in \mathcal{J}} C_j)$.

III. Improved Low Time-Complexity NP-EDF Schedulability Test on Multiprocessor

In this section, we develop two versions of NP-EDF schedulability tests that improve schedulability performance without compromising low time complexity.

A. Enhancing Existing Schedulability Test

In this section, we develop an enhanced schedulability test for NP-EDF, by reducing the pessimism of Bar in calculating V_i. Bar applies $\max_{J_j \in \mathcal{J}} C_j$ to all V_i for $J_i \in \mathcal{J}$, and the physical meaning of $\max_{J_j \in \mathcal{J}} C_j$ is that all jobs, which start their execution before r_i (i.e., the release time of J_i), should be finished no later than $r_i + \max_{J_j \in \mathcal{J}} C_j$. We claim that if we apply a job-level blocking upper bound instead of the unified upper bound (i.e., $\max_{J_j \in \mathcal{J}} C_j$), we can derive a lower upper bound of the blocking duration. To this end, we derive the following three properties for the upper bound of the blocking duration for every job J_i (where its release time is denoted by r_i); recall that D_i denotes the relative deadline of J_i, not the absolute deadline.

P1: J_i cannot be blocked by its lower priority jobs after $r_i + \max_{J_j \in \mathcal{J}(r_i) \setminus \{J_i\}} C_j$.

P2: J_i cannot be blocked by its lower priority jobs after $r_i + D_i$.

P3: J_i cannot be blocked by a lower priority job J_j, if J_j's relative deadline is no larger than J_i's relative deadline (i.e., $D_j \leq D_i$).

P1 is similar to Bar, but J_i itself is excluded in the job candidates that incur the largest blocking time. P2 is straightforward since J_i's deadline is $r_i + D_i$. P3 holds because a lower priority job whose deadline is later than $r_i + D_i$ has its release time after r_i if its relative deadline is smaller than D_i. Combining P1–P3, we derive the following lemma.

Lemma 2: A job of J_i whose release time is r_i cannot be blocked by its lower priority jobs after $r_i + B_i$, where

$$B_i = \min\left(D_i, \max_{J_j \in \mathcal{J}(r_i) \setminus \{J_i\} | D_j > D_i} C_j\right). \tag{2}$$

Note that $B_i = 0$ if $\mathcal{J}(r_i) \setminus \{J_i\} | D_j > D_i = \emptyset$.

Proof: The lemma holds by P1–P3. ■

Using B_i, we define V'_i and derive an enhanced schedulability analysis in the following theorem.

Theorem 1: A job set $\mathcal{J}$ is schedulable by NP-EDF on m processors, if for every $t \in \{r_j\}_{J_j \in \mathcal{J}}$, the following inequality and $C_i \leq D_i - B_i$ hold for every $J_i \in \mathcal{J}(t)$:

$$\sum_{J_i \in \mathcal{J}(t)} V'_i \leq m - (m-1) \times \max_{J_i \in \mathcal{J}(t)} V'_i \tag{3}$$

where $V'_i = C_i/(D_i - B_i)$.

From now on, we prove the correctness of the theorem. Although Theorem 1 is simple to present, it is difficult to prove its correctness, which can be achieved by comparing the amount of job executions under NP-EDF, with that under a rate-based scheduler defined as follows.

Definition 1: The R* scheduler executes every job J_i with speed of V'_i at t if the job has remaining execution.

Since $V'_i \cdot (D_i - B_i) = C_i$ holds, every job J_i under the R* scheduler finishes its execution at least B_i time units prior to its absolute deadline (if $D_i - B_i \geq C_i$ holds).

In addition to Definition 1, the proof of Theorem 1 necessitates the following definition.

Definition 2: For a scheduler S, a job J_q that belongs to a job set $\mathcal{J}$, and a time interval $[t_a, t_b]$, let $W(S, J_q, [t_a, t_b))$ denote the amount of execution of J_q done when jobs in $\mathcal{J}$ are scheduled by S during interval $[t_a, t_b)$.

Using Definitions 1 and 2, we will derive a key property that can prove Theorem 1, stated in the following lemma.

Lemma 3: If (3) holds, the following condition holds for every $p \geq 1$ and $t \geq 0$:

$$\sum_{q=1}^{p} W\big(\text{NP-EDF}, J_q, [0, t)\big) \geq \sum_{q=1}^{p} W\big(\text{R}^*, J_q, [0, t - B_q)\big). \tag{4}$$

Proof: Recall that J_q is defined as the job with the qth earliest deadline among all jobs in $\mathcal{J}$, and r_q and d_q denote J_q's release time and absolute deadline, respectively.

The proof of the lemma is proved by contradiction. Suppose that t_0 is the earliest time instant that (4) is violated, implying that

$$\sum_{q=1}^{p} W\big(\text{NP-EDF}, J_q, [0, t_0)\big) < \sum_{q=1}^{p} W\big(\text{R}^*, J_q, [0, t_0 - B_q)\big). \tag{5}$$

Then, there should be at least one job $J_j \in \{J_q\}_{q=1}^{p}$ that satisfies the following inequality at t_0 ($> r_j + B_j$) where r_j denotes the release time of J_j; otherwise, (5) cannot hold

$$W\big(\text{NP-EDF}, J_j, [0, t_0)\big) < W\big(\text{R}^*, J_j, [0, t_0 - B_j)\big). \tag{6}$$

At $r_j + B_j$ ($< t_0$), the following inequality holds because its RHS is zero

$$W\big(\text{NP-EDF}, J_j, [0, r_j + B_j)\big) \geq W\big(\text{R}^*, J_j, [0, r_j)\big). \tag{7}$$

Since $r_j + B_j < t_0$ holds, (4) holds for $t = r_j + B_j$. Then, if we subtract (4) for $t = r_j + B_j$ from (5), the following inequality holds:

$$\sum_{q=1}^{p} W\big(\text{NP-EDF}, J_q, [r_j + B_j, t_0)\big) < \sum_{q=1}^{p} W\big(\text{R}^*, J_q, [r_j + B_j - B_q, t_0 - B_q)\big). \tag{8}$$

Subtracting (7) from (6), we derive the following:

$$W\big(\text{NP-EDF}, J_j, [r_j + B_j, t_0)\big) < W\big(\text{R}^*, J_j, [r_j, t_0 - B_j)\big). \tag{9}$$

While Bar calculates the latest time instant when a lower priority job of J_j blocks J_j, as $r_j + \max_{J_i \in \mathcal{J}} C_i$, we calculate the time instant as $r_j + B_j$. Then, if J_j is not executing in $[r_j + B_j, t_0)$, all processors should be occupied by jobs whose priority is higher than J_j. Let x denote the cumulative length in $[r_j+B_j, t_0)$ such that all processors are occupied by jobs in $\{J_q\}_{q=1}^{p}$. Also, let y denote the cumulative length in the same interval such that at least one processor is idle or occupied by a job not in $\{J_q\}_{q=1}^{p}$. Note that J_j does not finish its execution until t_0 because of (6). The LHS of (8) is at least $m\cdot x+y$, while the RHS of the inequality is at most $(x+y)\cdot\sum_{J_i\in\mathcal{J}(t^*)} V_i'$, where t^* maximizes $(x+y)\cdot\sum_{J_i\in\mathcal{J}(t)} V_i'$ among all $t\in\{r_k\}_{J_k\in\mathcal{J}}$ where $r_j\le r_k<t_0-B_j$. Therefore, the following inequality holds:

$$m\cdot x+y<(x+y)\cdot\sum_{J_i\in\mathcal{J}(t^*)} V_i'. \tag{10}$$

Also, the LHS of (9) is at least y, while the RHS of the inequality is at most $(x+y)\cdot V_j'$, yielding the following inequality:

$$y<(x+y)\cdot\max_{J_i\in\mathcal{J}(t^*)} V_i'. \tag{11}$$

Then, (3) for $t=t^*$ contradicts, if we multiply (11) and $(m-1)$ and add it to (10). This proves the lemma. ■

Proof of Theorem 1: We can prove Theorem 1 using Lemma 3 as follows.

Base Case: If we focus on Lemma 3 with $p=1$ and $t=d_1$, the RHS of (4) is equal to C_1. Hence, the LHS is at least C_1, meaning that J_1 finishes its execution until d_1 under NP-EDF.

Inductive Case: Suppose that every job in $\{J_q\}_{q=1}^{x-1}$ finishes its execution within its deadline. We focus on (4) for given $p=x$ and $t=d_x$. Then, the RHS of (4) implies the summation of the execution time of $\{J_q\}_{q=1}^{x}$, since every job J_q finishes its execution until d_q-B_q, which is no later than d_x-B_q. On the other hand, the LHS implies the summation of the amount of execution of jobs in $\{J_q\}_{q=1}^{x}$ performed until d_x. Considering the LHS is no smaller than the RHS in (4), we conclude that every job in $\{J_q\}_{q=1}^{x}$ finishes its execution until d_x. By the supposition, this implies that J_x finishes its execution until its deadline d_x.

By the base and inductive cases, the theorem holds.

Theorem 1 enhances Lemma 1 (i.e., Bar) in terms of providing a tighter schedulability condition, which is possible by incorporating the properties of nonpreemptive scheduling. Also, while the original version of Bar cannot, Theorem 1 can be used for the job model in which jobs are dynamically in and out.

B. Further Improving Schedulability Test

While Theorem 1 is successful in developing an enhanced schedulability test for NP-EDF, we can further improve the proposed schedulability test. To this end, we will exploit a property from [6], which is originally derived for a task set (not for a job set), as follows. Focusing on (3), we investigate how excluding a job J_x and a processor affects its LHS and RHS. Once we exclude a job J_x and a processor, the LHS of (3) decreases by V_x' and the RHS decreases by $(1-\max_{J_i\in\mathcal{J}(t)} V_i')$. This implies that if V_x' is larger than $(1-\max_{J_i\in\mathcal{J}(t)} V_i')$, it is possible that (3), which does not hold for $\mathcal{J}(t)$ on an m-processor platform, can hold for $\mathcal{J}(t)\setminus\{J_x\}$ on an $(m-1)$-processor platform. By carefully utilizing the property, we can derive the following schedulability test from Theorem 1.

Theorem 2: Let $J_*(t)$ denote a job $J_i\in\mathcal{J}(t)$ whose V_i' is the largest among $\mathcal{J}(t)$; we choose only one job for $J_*(t)$ if there are multiple jobs with the largest V_i'. We also let $\mathcal{J}'(t)$ denote a set of jobs $J_j\in\mathcal{J}(t)\setminus\{J_*(t)\}$ whose V_j' is larger than $(1-\max_{J_i\in\mathcal{J}(t)} V_i')$, and $m'(t)$ denotes the number of jobs in $\mathcal{J}'(t)$. Then, a job set $\mathcal{J}$ is schedulable by NP-EDF on m processors, if for every $t\in\{r_j\}_{J_j\in\mathcal{J}}$, the following inequality and $C_i\le D_i-B_i$ hold for every $J_i\in\mathcal{J}(t)$ and $m'(t)<m$ holds:

$$\sum_{J_i\in\mathcal{J}(t)\setminus\mathcal{J}'(t)} V_i'\le(m-m'(t))-(m-m'(t)-1)\times\max_{J_i\in\mathcal{J}(t)\setminus\mathcal{J}'(t)} V_i'. \tag{12}$$

Proof: From Theorem 1, (12) implies $\mathcal{J}(t)\setminus\mathcal{J}'(t)$ is schedulable by NP-EDF on an $(m-m'(t))$-processor platform. Since a job cannot occupy more than one processor at any time, adding jobs in $\mathcal{J}'(t)$ and $m'(t)$ processors yields the following: if $\mathcal{J}(t)$ is scheduled by NP-EDF on an m-processor platform, any job in $\mathcal{J}(t)\setminus\mathcal{J}'(t)$ does not yield any job deadline miss.

Then, the remaining step is to guarantee no deadline miss for jobs in $\mathcal{J}'(t)$, which can hold by switching a job in $\mathcal{J}'(t)$ [denoted by $J_{**}(t)$] and $J_*(t)$. Let $\mathcal{J}''(t)$ denote $\mathcal{J}'(t)\cup\{J_*(t)\}\setminus\{J_{**}(t)\}$. Then, if we replace $\mathcal{J}'(t)$ with $\mathcal{J}''(t)$ in (12), the new inequality also holds, due to the following reasons.

The LHS of the new inequality decreases by V_i' for $J_*(t)$ minus V_i' for $J_{**}(t)$ (denoted by α that is nonnegative), compared to the LHS of (12). The RHS of the new inequality increases by at most $(m-m'(t)-1)\cdot\alpha\ge 0$, compared to the RHS of (12). Therefore, we can guarantee that when $\mathcal{J}(t)$ is scheduled by NP-EDF on an m-processor platform, any job in $\mathcal{J}(t)\setminus\mathcal{J}''(t)$ does not yield any job deadline miss. Since $\mathcal{J}(t)\setminus\mathcal{J}''(t)$ subsumes $J_{**}(t)$, we can guarantee no job deadline miss for $J_{**}(t)$.

Likewise, we can guarantee no job deadline miss for every job in $\mathcal{J}'(t)$, which proves the theorem. ■

C. Usage and Time Complexity

We can use Theorems 1 and 2 for a runtime admission control for a job set. Whenever a new job is released, we test the theorems for $\mathcal{J}$ by adding the new job to $\mathcal{J}$; if (3) [or (12)] holds, the system can accommodate the new job. This needs $O(n(t))$ operations to calculate B_i (and therefore, V_i') for each $J_i\in\mathcal{J}(t)$, where $n(t)$ is the number of jobs in $J_i\in\mathcal{J}(t)$, yielding $O(n(t)^2)$ time complexity.

Theorems 1 and 2 can be used for testing the schedulability of a task set, by replacing $\mathcal{J}(t)$ in (3) [or (12)] with τ, where a task $\tau_i\in\tau$ is specified as the relative deadline D_i, the worst case execution time C_i, and the period T_i ($\ge D_i$). This also yields $O(n^2)$ time complexity where n is the number of tasks. This time complexity is still very low, compared to existing schedulability tests that yield high schedulability performance at the expense of high time-complexity [1]–[4]. For example, one of NP-EDF tests with the highest schedulability performance in [3], [4] yields $O(n^3\cdot\max_{\tau_i\in\tau} D_i^2)$, which is pseudopolynomial time complexity.

TABLE I
NUMBER OF SCHEDULABLE TASK SETS BY EACH TEST, NORMALIZED BY THAT BY BAR, WHEN $m = 2$

Task utilization distribution (average # of tasks)	Bar	Ours1	Ours2
Bino. 0.1 (5.05)	100.0%	123.8%	172.7%
Bino. 0.3 (4.30)	100.0%	130.6%	189.0%
Bino. 0.5 (3.79)	100.0%	145.2%	218.5%
Bino. 0.7 (3.40)	100.0%	171.7%	273.3%
Bino. 0.9 (3.11)	100.0%	377.3%	692.4%
Exp. 0.1 (11.62)	100.0%	102.8%	108.7%
Exp. 0.3 (5.74)	100.0%	116.1%	142.4%
Exp. 0.5 (4.79)	100.0%	124.8%	163.5%
Exp. 0.7 (4.45)	100.0%	128.7%	175.8%
Exp. 0.9 (4.27)	100.0%	132.6%	183.0%
Total (5.05)	100.0%	119.6%	153.0%

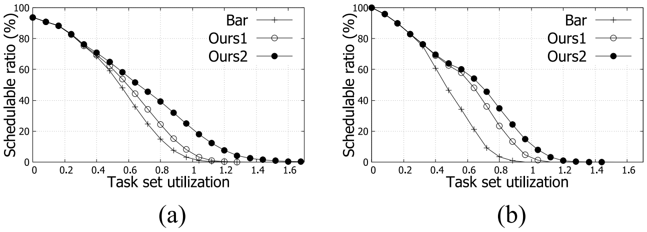
IV. EVALUATION

In this section, we evaluate the schedulability performance of the proposed schedulability tests. To this end, we generate a number of task sets and apply them to test Theorems 1 and 2 according to Section III-C.

We follow a well-known task set generation method [8], which has been widely used for many real-time scheduling studies, e.g., [3], [4], and [6]. We generate 100 000 task sets for every pair of: 1) the number of processors m (2, 4, and 8) and 2) the task utilization C_i/D_i (the bimodal distribution with 0.1, 0.3, 0.5, 0.7, and 0.9, and the exponential distribution with 0.1, 0.3, 0.5, 0.7, and 0.9). Different choices in 2) yield different average values for the number of tasks in each task set as shown in Table I, which shows the number of schedulable task sets by each test normalized by that by Bar, when $m = 2$.

We now compare the schedulability performance of Theorems 1 and 2 (denoted by Ours1 and Ours2, respectively), with that of the existing low time-complexity schedulability test (i.e., Lemma 1, denoted by Bar). Overall, Ours1 and Ours2 significantly improve the schedulability performance of Bar. As shown in Table I, Ours1 and Ours2, respectively, find 19.6% and 53.0% more task sets for $m = 2$ than Bar. Although not as large as the amount of improvement for $m = 2$, that for $m = 4$ and $m = 8$ is still significant in that Ours1 and Ours2 yield 7.7% and 37.6% improvement for $m = 4$ and 11.2% and 28.9% improvement for $m = 8$, respectively.

As shown in Table I, Ours1 and Ours2 yield a greater improvement with a smaller value for the average number of tasks in each task set (denoted by $\bar{n}$). For example, while a binomial distribution with 0.1 ($\bar{n} = 5.05$) yields only 23.8% and 72.7% improvement for Ours1 and Ours2, respectively, that with 0.9 ($\bar{n} = 3.11$) yields 277.3% and 592.4% improvement. This is because, a smaller number of tasks in a task set tends to yield a larger $\max_{\tau_i \in \tau} V_i$, which emphasizes the pessimism of Bar, while Ours1 and Ours2 effectively reduce the pessimism. Note that the improvement does not significantly depend on the task utilization distribution type (i.e., binomial or exponential), as we observe a similar $\bar{n}$ yields a similar improvement regardless of the type; for example, a binomial distribution with 0.3 ($\bar{n} = 4.30$) and an exponential distribution with 0.9 ($\bar{n} = 4.27$) yield a similar improvement.



(a)

(b)

Fig. 1. Ratio of task sets deemed schedulable by each test with $m = 2$. (a) Bimodal distribution with 0.1. (b) Bimodal distribution with 0.9.

We also observe that the performance gap between Bar and Ours1 and that between Ours1 and Ours2 varies with the task utilization distribution. As shown in Table I and Fig. 1, the former is larger than the latter for a binomial distribution with 0.9, while the converse holds for other distributions. This means the improvement by Theorem 1 stands out when the number of tasks in each task set is small.

Since the proposed schedulability tests are designed for exhibiting low time complexity, they cannot yield the schedulability performance comparable to high time-complexity schedulability tests. However, the proposed schedulability tests can find some additional task sets schedulable by NP-EDF, which cannot be covered by existing high time-complexity schedulability tests. That is, among 89 509 task sets deemed schedulable by Ours2 for $m = 2$, 2459 task sets are not covered by the existing high time-complexity schedulability tests [3], [4], which are known to exhibit the highest schedulability performance for NP-EDF.

V. CONCLUSION

In this letter, we developed two versions of schedulability test for NP-EDF, which not only improve the schedulability performance but also maintain low time complexity, and we demonstrated their effectiveness via experiments. While this letter focused on work-conserving NP-EDF scheduling, there exist some recent studies that deal with nonwork-conserving NP-EDF, e.g., [9]; it would be interesting to develop a method that extends the techniques proposed in this letter to nonwork-conserving NP-EDF.

REFERENCES

[1] N. Guan, W. Yi, Z. Gu, Q. Deng, and G. Yu, “New schedulability test conditions for non-preemptive scheduling on multiprocessor platforms,” in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2008, pp. 137–146.

[2] H. Leontyev and J. H. Anderson, “A hierarchical multiprocessor bandwidth reservation scheme with timing guarantees,” in *Proc. Euromicro Conf. Real-Time Syst. (ECRTS)*, 2008, pp. 191–200.

[3] J. Lee and K. G. Shin, “Controlling preemption for better schedulability in multi-core systems,” in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2012, pp. 29–38.

[4] J. Lee and K. G. Shin, “Improvement of real-time multi-core schedulability with forced non-preemption,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 5, pp. 1233–1243, May 2014.

[5] S. Baruah, “The non-preemptive scheduling of periodic tasks upon multiprocessors,” *Real-Time Syst.*, vol. 32, nos. 1–2, pp. 9–20, 2006.

[6] J. Lee, K. G. Shin, I. Shin, and A. Easwaran, “Composition of schedulability analyses for real-time multiprocessor systems,” *IEEE Trans. Comput.*, vol. 64, no. 4, pp. 941–954, Apr. 2015.

[7] S. Baruch and G. Lipari, “Executing aperiodic jobs in a multiprocessor constant-bandwidth server implementation,” in *Proc. Euromicro Conf. Real-Time Syst. (ECRTS)*, 2004, pp. 109–116.

[8] T. P. Baker, “Comparison of empirical success rates of global vs. paritioned fixed-priority and EDF scheduling for hand real time,” Dept. Comput. Sci., Florida State Univ., Tallahassee, FL, USA, Rep. TR-050601, 2005.

[9] H. Baek, J. Kwak, and J. Lee, “Non-preemptive real-time multiprocessor scheduling beyond work-conserving,” in *Proc. IEEE Real-Time Syst. Symp. (RTSS)*, 2020, pp. 102–114.