JCSE Journal of Computing Science and Engineering

Regular Paper

Journal of Computing Science and Engineering,
Vol. 15, No. 2, June 2021, pp. 72-77

Impact of Synthetic Task Set Generation Methods on Schedulability Performance

Saehwa Kim*
Department of Information Communications Engineering, Hankuk University of Foreign Studies, Yongin, Korea
ksaehwa@hufs.ac.kr

Abstract
This paper addresses the various alternative methods of synthesizing task sets even when the continuous uniform distribution of their task utilizations is guaranteed. There are four methods that have been widely used in literature; LinearC, LinearT, LogT, and HarmonicT: C and T represent the worst-case execution times and periods, while linear, log, harmonic represent the spaces for the random generation of C or T. We have demonstrated that the schedulability performances of the task sets generated by those methods are very different. Specifically, the schedulability performance for the fixed priority scheduling is in the decreasing order of LogT, HarmonicT, LinearC, and LinearT. We have introduced notions of C-difference and T-difference, which have been used to demonstrate that the larger the value induced the better schedulability performance.

Category: Real-Time Systems

Keywords: Empirical evaluation; Fixed-priority scheduling; Real-time systems and embedded systems; Performance measurement

I. INTRODUCTION

The performance of fixed priority scheduling algorithms or schedulability attributes assignment algorithms is typically evaluated in terms of its schedulability: the ratio of schedulable task sets among all randomly generated task sets. To synthesize a task set, it is common to generate each task τ_i's utilization U_i in a uniform distribution for a given number of tasks n and a given total utilization U. Here, the UUniFast [1] algorithm is the de facto standard method for generating the utilizations of tasks. This algorithm was proposed to address the way task parameters are generated because they may significantly affect the schedulability performance result and cause bias in the judgment on the schedulability tests. To express the utilization disparity in a set of tasks, [1] introduced the following parameter called U-difference, which we denote as Δ_U.

$$\Delta_U = \frac{\max_i\{U_i\} - \min_i\{U_i\}}{\sum_{i=1}^{n} U_i} \quad (1)$$

The range of D_U is [0, 1], where 0 and 1 represent the minimum and the maximum differences, respectively. With this, the UUniFast algorithm guarantees generation of evenly distributed tasks in the utilization space.

However, we also show that there are other alternative methods of generating each task τ_i's worst-case execution time C_i and period T_i with the given $U_i = C_i/T_i$, even though the uniform distribution of U_i is guaranteed. We derive six alternative methods and compare the schedulability performance of four methods that are widely

Open Access http://dx.doi.org/10.5626/JCSE.2021.15.2.72 http://jcse.kiise.org

Received 16 February 2021; **Accepted** 02 April 2021
*Corresponding Author

 pISSN: 1976-4677 eISSN: 2093-8020

used in literature. We demonstrate that the effects of employing different methods on the schedulability performance for the fixed priority scheduling are significant. To analyze such phenomena, we introduce the notions of C-difference and T-difference, which are similar to *U*-difference [1]. With these, we demonstrate that the level of the distributions of their joint probability density functions is proportional to the schedulability performance.

The remainder of the paper is organized as follows. Section II highlights the task model and example task sets that motivate this work. Section III summarizes alternative methods of generating task sets. Section IV presents impacts of alternative methods on the schedulability performance. Section V analyzes the impacts of using the notions of C-difference and T-difference. Section VI concludes the paper.

II. TASK MODEL AND MOTIVATING EXAMPLE TASK SETS

We presume that a single processor and a system have a fixed set of tasks $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Each task τ_i has a fixed period T_i, a fixed relative deadline D_i, and a known worst-case execution time C_i. There are no restrictions such that each task's deadline should be shorter than its period. We presume that tasks are scheduled under the fully preemptive fixed priority scheduling (FPS). Since the deadline monotonic priority ordering (DMPO) is optimal in FPS, we assume that all task priorities are assigned using DMPO.

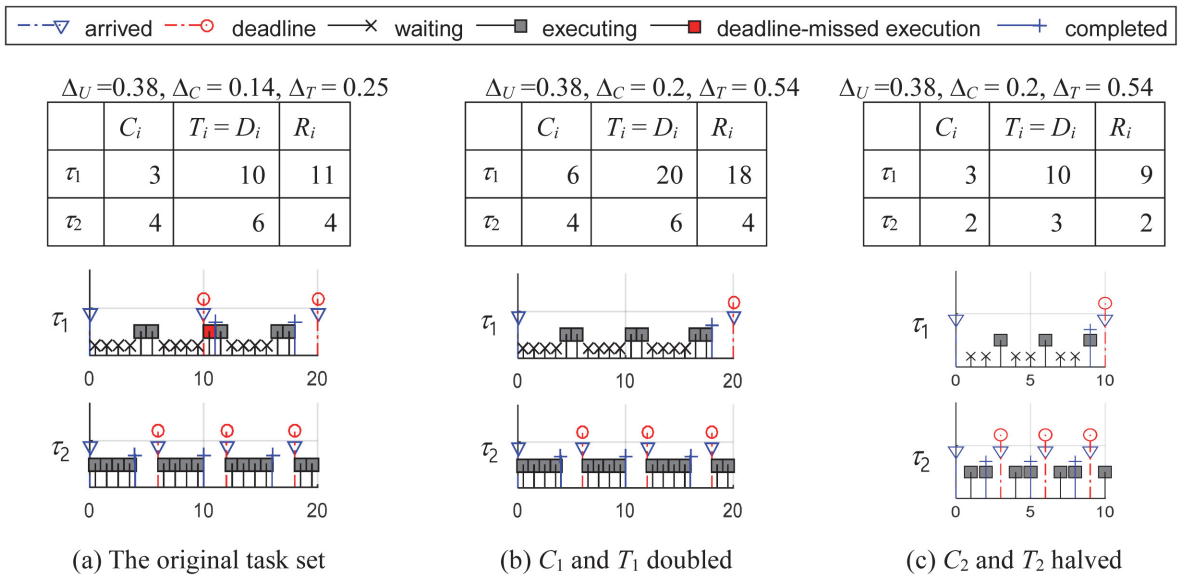
All of the three task sets in Fig. 1 have the same *U*-difference of $\Delta_U = 0.38$ where $U_1 = 3/10$ and $U_2 = 2/3$, taking task set as the original task set. Then, C_1 and T_1 of task set are doubled, while C_2 and T_2 of task set are halved in comparison to those of the original task set. As Fig. 1 shows, task set (a) is not schedulable while task sets (b) and (c) are schedulable. It shows that the schedulability results of task sets of the same *U*-difference with the same task utilizations may be different, which motivated our work.

III. ALTERNATIVE METHODS FOR TASK SET SYNTHESIS

We presume a single processor. As the example task sets of Fig. 1 show, same *U*-difference with same utilization of each task does not give the same schedulability. This section presents alternative methods in synthesizing task sets even if we generate task sets in the uniform distribution of utilizations. Specifically, after getting the uniform distribution of utilizations, there are two aspects to consider: 1) (timing attribute) targets to randomly generate and 2) space where the random generation of targets is performed.

For the first aspect of the target, there are two alternatives: 1) the worst case execution time C_i or 2) the period of a task T_i to be generated first should be decided while the other one is derived from the equation $C_i = T_i \times U_i$ or $T_i = C_i/U_i$. For the second aspect of the space, there are three alternatives—linear, log, and harmonic, the chosen target values of the first aspect (C_i or T_i) to be generated in a uniform distribution.

From these two aspects and their alternatives, six



	C_i	$T_i = D_i$	R_i
τ_1	3	10	11
τ_2	4	6	4

	C_i	$T_i = D_i$	R_i
τ_1	6	20	18
τ_2	4	6	4

	C_i	$T_i = D_i$	R_i
τ_1	3	10	9
τ_2	2	3	2

(a) The original task set (b) C_1 and T_1 doubled (c) C_2 and T_2 halved

Fig. 1. Schedules produced for task sets whose *U*-differences are the same ($\Delta_U = 0.38$) with $U_1 = 3/10$ and $U_2 = 2/3$). (a) is the original task set, (b) is a task set whose C_1 and T_1 were doubled, and (c) is a task set whose C_2 and T_2 were halved compared to the original task set (a). R_i is the worst-case response time of task τ_i. Note that while task set (a) is not schedulable (since R_1 (=11) > D_1 (=10)), task sets of (b) and (c) are schedulable.

Table 1. Alternative methods for generating task sets with the given $U_i = C_i/T_i$

Random generation		Target	
		C_i	T_i
Space	Linear	LinearC	LinearT
		-linearC1: [100, 500] in [2]	-linearT1: [1, 10^6] in [3] -linearT2: [10, 10^3] in [4, 5] -linearT2′: [25, 10^3] in [6] -linearT2″: [50, 10^3] in [7] -linearT2‴: [50, 200], [200, 500], [500, $2{\cdot}10^3$] in [8] -linearT3: [10^2, 10^5] in [9]
	Log	LogC	LogT
		-N/A	-logT1: [10^3, 10^5] in [10] -logT1′: [20, 200] in [11] -logT1″: [1, 10^3] in [12] -logT2: [10^3, 10^7] in [10] -logT3: [10^3, 10^9] in [10]
	Harmonic	HarmonicC	HarmonicT
		-N/A	-harmonicT1: randomly drawn from {5, 10, 20, 40, 50, 100, 200, 400, 500, 1000} in [13] -harmonicT1′: randomly drawn from {25, 50, 75, 100} in [14] -harmonicT1″: randomly drawn while minimizing hyper-period in [15, 16] -harmonicT2: product of 1–3 factors, each of which randomly drawn from harmonic sets (2, 4), (5, 10), (6, 12) in [17]

alternative methods are possible as shown in Table 1: LinearC, LinearT, LogC, LogT, HarmonicC, and HarmonicT. Table 1 also shows example usages of these alternatives. Specifically, LinearC has been adopted in [2] with duration of [100, 500]. LinearT is the most widely used method in literature as it has been adopted in various durations: [1, 10^6] in [3], [10, 10^3] in [4, 5], [25, 10^3] in [6], [50, 10^3] in [7], [50, 200], [200, 500], [500, $2{\cdot}10^3$] in [8], and [10^2, 10^5] in [9], while LogT has also been adopted in various durations: [10^3, 10^5] in [10], [20, 100] in [11], [1, 10^3] in [12], and [10^3, 10^7], [10^3, 10^9] in [10]. HarmonicT has also been adopted in various combinations: randomly drawn from {5, 10, 20, 40, 50, 100, 200, 400, 500, 1000} in [13], {25, 50, 75, 100} in [14], randomly drawn while minimizing hyperperiod in [15, 16], or the product of one to three factors, each randomly drawn from three harmonic sets (2, 4), (5, 10), (6, 12) in [17]. To the best of our knowledge, there have been no example usages for LogC and HarmonicC. We named specific examples of LinearC, LinearT, LogT, and HamonicT as linearC1, linearT1/2/3, logT1/2/3, and harmonicT1/2 as shown in Table 1. In the next section, we empirically compare the schedulability performance of all of the example methods in Table 1.

IV. SCHEDULABILITY COMPARISON

The experiments were conducted with MATLAB R2016b on an Intel Core i7-4790, 3.60 GHz system with 16 GB of RAM. For a specific given total utilization U, we generated utilization U_i for each task τ_i using UUniFast [1] algorithm. After that, for linearC1, we generated C_i as a random integer uniformly distributed in the linear interval [100, 500] and got $T_i = C_i/U_i$ as in [2]. For the other methods (linearT1/2/3, logT1/2/3, and harmonicT1/2), we generated T_i as a random integer uniformly distributed in the linear or log intervals in Table 1, and got $C_i = T_i \times U_i$. For any method, we generated D_i as a random integer uniformly distributed in the interval $[C_i + 0.5 \times (T_i - C_i), T_i]$. For each configuration, we generated 10,000 task sets. Fig. 2 shows MATLAB code for synthesizing task sets with linearC1, linearT3, logT3, and harmonicT2.

Fig. 3 shows the ratios of feasible task sets that were generated by each of the example methods of Table 1: Fig. 3(a) for varying utilization with $n = 20$ and Fig. 3(b) for varying number of tasks with $U = 0.9$. As shown in both Fig. 3(a) and 3(b), the schedulability ratios of example methods of LinearT (linearT1, linearT2, and linearT3) are almost the same and are also the worst. Moreover, they are convex (downward) in both Fig. 3(a) and 3(b) the decreasing ratios (derivatives/slopes) decrease with the larger n and U. On the other hand, the ratio results of linearC1 are almost the same as those of logT1. While they are much larger than the schedulability ratios of all example methods of LinearT, they are all smaller than those of the Harmonic example methods (harmonicT1 and harmonicT2), which are also smaller than those of logT2 and logT3.

```
C = randi([100, 500], 1, n);
T = round(C ./ vectU, 0);
```
(a)

```
T = randi([100, 100000], 1, n);
C = T.*vectU;
```
(b)

```
T = round(10.^(3 + 3 * rand(1, n)), 0);
C = T.*vectU;
```
(c)

```
Tcand = [1, 2, 4; 1, 5, 10; 1, 6, 12];
for i = 1:n
  while (1)
    T(i) = Tcand(1, randi([1,3], 1))*...
           Tcand(2, randi([1,3], 1))*...
           Tcand(3, randi([1,3], 1));
    if T(i) >= 3
      break;
    end
  end % end while
end % end for i
C = T.*vectU;
```
(d)

Fig. 2. MATLAB code for task set synthesis for (a) linearC1, (b) linearT3, (c) logT3, and (d) harmonicT2, where n is the number of tasks in a task set and `vectU` is a vector of which each element is the utilization of each task.

V. C-DIFFERENCE AND T-DIFFERENCE

To analyze the impacts of task set generation methods on schedulability, we introduced another set of parameters in addition to *U*-difference; *C*-difference and *T*-difference, and expressed the execution time and period disparities, respectively, in a set of tasks. They were denoted as Δ_C and Δ_T, respectively, and we defined them like *U*-difference as follows:

$$\Delta_C = \frac{\max_i\{C_i\} - \min_i\{C_i\}}{\sum_{i=1}^{n} C_i} \tag{2}$$

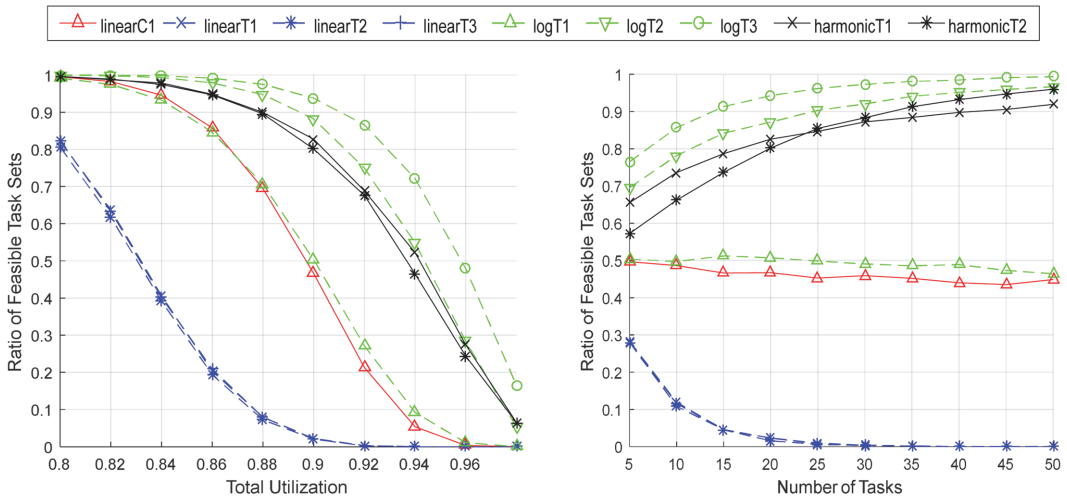
$$\Delta_T = \frac{\max_i\{T_i\} - \min_i\{T_i\}}{\sum_{i=1}^{n} T_i} \tag{3}$$

For Fig. 1(a), Δ_C = 0.14, Δ_T = 0.25, while for both Fig. 1(b) and 1(c), Δ_C = 0.2, Δ_T = 0.54; both *C*-difference and *T*-difference in Fig. 1(b) and 1(c) are larger than in Fig. 1(a). These example task sets infer that the larger the *C*-difference and the *T*-difference are, the greater the possibility of schedulability of the task sets. Note that all task sets in Fig. 1(a) have the same *U*-difference of Δ_U = 0.38.

Fig. 4 shows contour graphs for the joint probability density function (pdf) of Δ_C and Δ_T for linearC1, linearT3, logT3, and harmonicT2, each of which is a representative example of LinearC, LinearT, LogT, and HarmonicT. These are derived from the synthesized task sets described in Section IV.

Note that utilizations for all task sets were uniformly generated with UUniFast [1]. As shown in Fig. 4, the degrees of the distributions of the joint pdf of Δ_C and Δ_T are in the decreasing order of logT3, hamonitcT2, linearC1, and linearT3, which is the same as the order of the schedulability ratio performance in Fig. 3.

Specifically, Fig. 4(a) shows that linearC1 generates C_i that is characterized by values of *C*-difference that are very close to zero, which means that C_i values tend to be



(a) $n = 20$ (b) $U = 0.9$

Fig. 3. Ratio of feasible task sets with (a) varying utilization and (b) varying number of tasks.

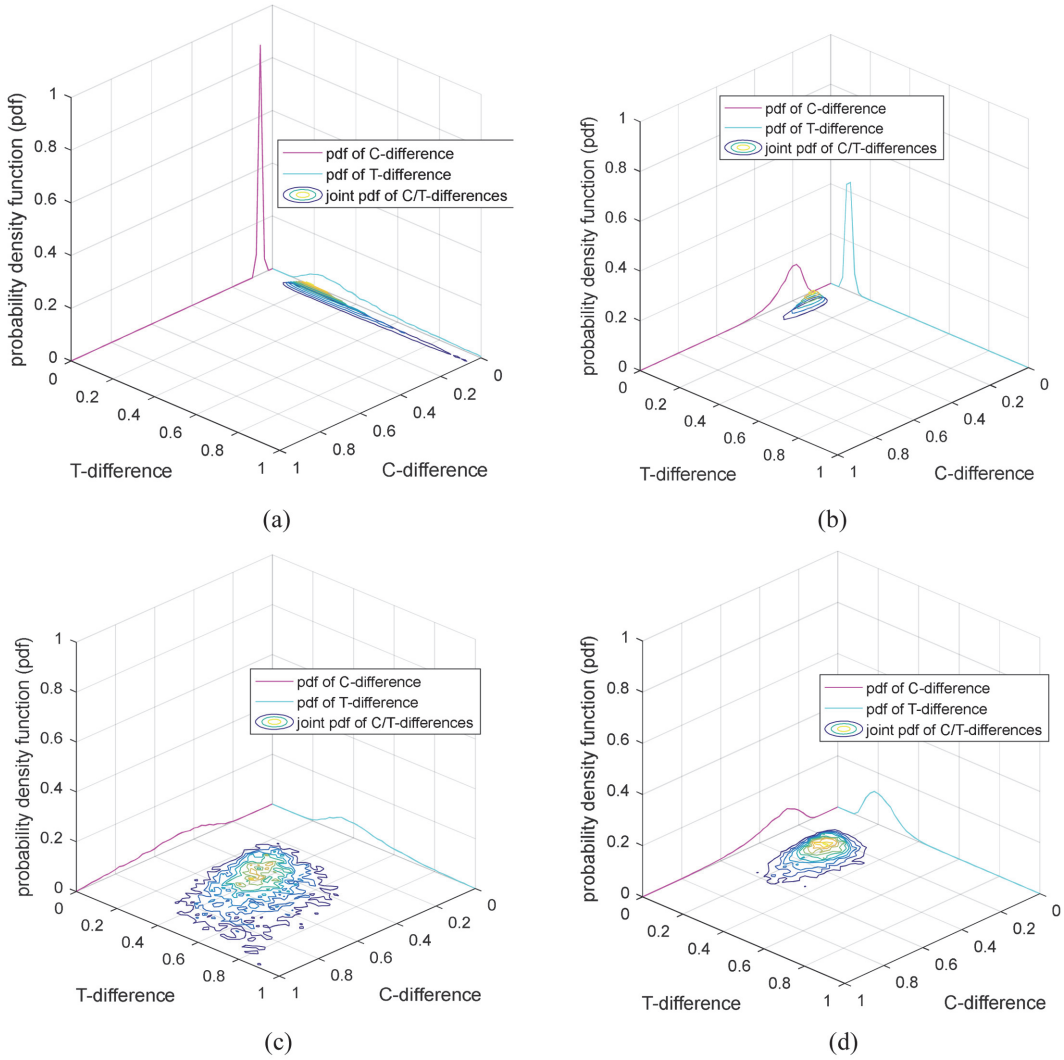


Fig. 4. Contour graphs for the joint probability density function (pdf) of C-difference and T-difference for (a) linearC1, (b) linearT3, (c) logT3, and (d) harmonicT2.

similar. Fig. 4(b) shows that linearT3 generates T_i that is characterized by values of *T*-difference that are close to zero, which also means that T_i values tend to be similar. Fig. 4(c) shows that logT3 generates C_i and T_i that are characterized by widely spread values of *C*-difference and *T*-difference. Finally, Fig. 4(d) shows that harmonicT2 generates C_i and T_i that are characterized by *C*-difference and *T*-difference values that are less spread than those of logT3.

VI. CONCLUSION

This paper has shown that the synthetic task set generation methods can generate varying results; however, uniform distribution of task utilizations is guaranteed. Specifically, after getting the uniform distribution of task utilizations, there are two aspects to consider: 1) timing attribute which targets to randomly generate and 2) space where the random generation of targets is performed. With this, we showed that six alternative methods are derivable, among which we noted that four methods have been widely used in literature: LinearC, LinearT, LogT, and HarmonicT. With empirical experiments, we showed that the schedulability performances of task sets generated by these methods are significantly different in respects to not only the schedulability performance but also the varying tendencies/derivatives with varying utilizations or varying number of tasks. To analyze such effects, we introduced the notions of C-difference and T-difference. We also demonstrated that the larger the values are, the better the schedulability is. Specifically, we showed that their joint probability density functions are in the decreasing order of LogT, HarmonicT, LinearC, and LinearT, which is the same order for the schedulability performance.

ACKNOWLEDGMENTS

This work was supported by Hankuk University of Foreign Studies Research Fund. This research was also supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2017R1A2B1001824).

REFERENCES

1. E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Systems*, vol. 30, no. 1-2, pp. 129-154, 2005.
2. G. C. Buttazzo, M. Bertogna, and G. Yao, "Limited preemptive scheduling for real-time systems: a survey," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 3-15, 2013.
3. E. Bini and G. C. Buttazzo, "Schedulability analysis of periodic fixed priority systems," *IEEE Transactions on Computers*, vol. 53, no. 11, pp. 1462-1473, 2004.
4. R. J. Bril, S. Altmeyer, M. M. van den Heuvel, R. I. Davis, and M. Behnam, "Integrating cache-related pre-emption delays into analysis of fixed priority scheduling with pre-emption thresholds," in *Proceedings of 2014 IEEE Real-Time Systems Symposium*, Rome, Italy, 2014, pp. 161-172.
5. N. Min-Allah, S. U. Khan, N. Ghani, J. Li, L. Wang, and P. Bouvry, "A comparative study of rate monotonic schedulability tests," *The Journal of Supercomputing*, vol. 59, no. 3, pp. 1419-1430, 2012.
6. B. Hu, L. Thiele, P. Huang, K. Huang, C. Griesbeck, and A. Knoll, "FFOB: efficient online mode-switch procrastination in mixed-criticality systems," *Real-Time Systems*, vol. 55, no. 3, pp. 471-513, 2019.
7. J. J. Han, Z. Wang, S. Gong, T. Miao, and L. T. Yang, "Resource-aware scheduling for dependable multicore real-time systems: utilization bound and partitioning algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 12, pp. 2806-2819, 2019.
8. J. J. Han, S. Gong, Z. Wang, W. Cai, D. Zhu, and L. T. Yang, "Blocking-aware partitioned real-time scheduling for uniform heterogeneous multicore platforms," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 19, no. 1, pp. 1-25, 2020.
9. N. Min-Allah, S. U. Khan, X. Wang, and A. Y. Zomaya, "Lowest priority first based feasibility analysis of real-time systems," *Journal of Parallel and Distributed Computing*, vol. 73, no. 8, pp. 1066-1075, 2013.
10. R. I. Davis, A. Zabos, and A. Burns, "Efficient exact schedulability tests for fixed priority real-time systems," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1261-1276, 2008.
11. M. A. Awan, P. F. Souto, B. Akesson, K. Bletsas, and E. Tovar, "Uneven memory regulation for scheduling IMA applications on multi-core platforms," *Real-Time Systems*, vol. 55, no. 2, pp. 248-292, 2019.
12. S. Zhao, J. Garrido, R. Wei, A. Burns, A. Wellings, and A. Juan, "A complete run-time overhead-aware schedulability analysis for MrsP under nested resources," *Journal of Systems and Software*, vol. 159, article no. 110449, 2020. https://doi.org/10.1016/j.jss.2019.110449
13. H. Zeng, M. D. Natale, and Q. Zhu, "Minimizing stack and communication memory usage in real-time embedded applications," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 13, no. 5S, pp. 1-25, 2014.
14. C. Deutschbein, T. Fleming, A. Burns, and S. Baruah, "Multi-core cyclic executives for safety-critical systems," *Science of Computer Programming*, vol. 172, pp. 102-116, 2019.
15. A. Guasque, P. Balbastre, A. Crespo, and S. Peiro, "Energy efficient partition allocation in mixed-criticality systems," *PLoS One*, vol. 14, no. 3, article no. e0213333, 2019. https://doi.org/10.1371/journal.pone.0213333
16. M. Shirazi, M. Kargahi, and L. Thiele, "Performance maximization of energy-variable self-powered (m, k)-firm real-time systems," *Real-Time Systems*, vol. 56, no. 1, pp. 64-111, 2020.
17. H. Zeng and M. Di Natale, "An efficient formulation of the real-time feasibility region for design optimization," *IEEE Transactions on Computers*, vol. 62, no. 4, pp. 644-661, 2013.

Saehwa Kim https://orcid.org/0000-0003-3303-4218

Saehwa Kim received her B.S., M.S., and Ph.D. degrees in electrical and computer science engineering from Seoul National University, Seoul, Korea, in 1997, 2000, and 2006, respectively. In 1997 and 2006, she was a researcher and a senior researcher, respectively, in the Automation and Systems Research Institute, Seoul, Korea, where she was engaged in developing intelligent robot software platform and automotive operating systems. In 2007 and 2008, she was a senior research engineer at the Software & Solution Center of LG Electronics Company, Ltd., Seoul, Korea, where she was engaged in the development of mobile software platforms. She is currently a professor at the Department of Information Communications Engineering, Hankuk University of Foreign Studies, Korea. Her research interest is in embedded software platforms, edge computing, and deep reinforcement learning that are specialized for various industrial domains such as automotive vehicles and intelligent robots.