

# Priority Point Exploration Using MBO

Minh Duy Truong, Junjie Shi, Jian-Jia Chen, Jörg Rahnenführer, and Mario Günzel

TU Dortmund University

Email: {minhduy.truong, junjie.shi, jian-jia.chen, joerg.rahnenuhrer, mario.guenzel}@tu-dortmund.de

**Abstract**—While for ordinary sporadic real-time task sets (without self-suspension) the preemptive Earliest-Deadline-First (EDF) scheduler minimizes the maximum lateness and hence optimizes the task set for schedulability, for self-suspending tasks (i.e., tasks that suspend their execution on the processor voluntarily) this analytical result is no longer applicable. Therefore, when studying self-suspending tasks, exploring alternative scheduling algorithms is a great opportunity to improve schedulability and minimize the maximum lateness. One way to explore different scheduling algorithms is to apply EDF-Like (EL) scheduling, where a large number of scheduling algorithms can be described by setting a relative priority point for each task. Previous work in that regard is limited to two approaches, which either rely on training or can only optimize the schedulability.

In this work, we propose the usage of Model-Based Optimization (MBO), which is a method to optimize arbitrary black-box objective functions by using a surrogate to guide the search, for finding relative priority points for EL scheduling. First evaluations demonstrate that MBO simultaneously improves schedulability and reduces maximum lateness, thereby overcoming key limitations of previous approaches.

## I. INTRODUCTION

Common timing metrics for real-time systems encompass *schedulability*, i.e., all jobs finish their execution no later than the deadline, and *maximum lateness*, i.e., the highest value of finishing time minus the absolute deadline. To optimize such timing metrics, the choice of the scheduling algorithm is a major driver. To that end, the preemptive Earliest-Deadline-First (EDF) scheduler is shown to minimize the maximum lateness and hence to optimize the task set for schedulability for ordinary sporadic real-time task sets [7]. However, when tasks are allowed to suspend themselves, i.e., suspending their execution on the processor voluntarily before completing their execution, such results for EDF are not applicable [3]. Hence, when studying self-suspending tasks, exploring alternative scheduling algorithms is a great opportunity to significantly improve schedulability and maximum lateness. One example is depicted in Figure 1, where a different priority ordering can achieve schedulability while EDF produces a deadline miss.

One direction to approach the scheduler optimization is the study of EDF-Like (EL) scheduling in the context of self-suspending tasks [4], [8]. Specifically, EL generalizes a large variety of scheduling algorithms (such as EDF, Fixed-Priority (FP), and First-In-First-Out (FIFO) [4, Section 3]) by choosing relative priority points  $\Pi_i$  for each task  $\tau_i$ . Building upon existing response time analyses and schedulability tests provided in [4], [8], the relative priority points can be optimized with respect to different timing metrics.

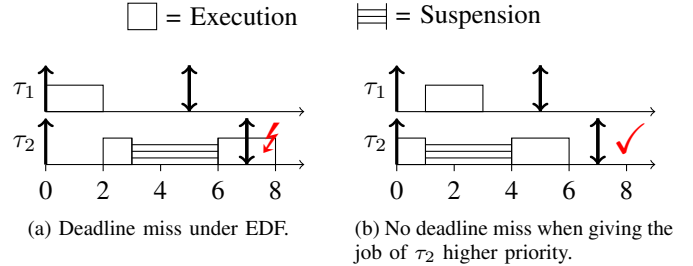


Figure 1. Example where a priority ordering different from EDF leads to a feasible schedule, while EDF produces a deadline miss.

Previous work [5] proposes two approaches to determine relative priority points based on the schedulability test provided by Günzel et al. [4]:

- **Iterative Approach:** Increasing the WCRT of schedulable tasks (by increasing their relative priority point). This potentially reduces the relative priority points of other tasks.
- **Genetic Algorithm (GA):** Train the parameters of a formula [5, Equation (3)] that translates task parameters into relative priority points.

However, these approaches have some drawbacks:

- The iterative approach just implicitly tries to obtain schedulability. Furthermore, it can not be used to optimize other metrics such as the maximum lateness.
- The performance of GA relies on test set generation. That is, the formula has to be trained on a set of similar task sets (e.g., similar number of tasks and suspension range) to obtain a good result.

Due to these limitations, in this paper we explore the possibility to use **Model-based optimization (MBO)**, based on *Bayesian Optimization* [6], which employs a Gaussian-process surrogate to model the objective and guides the search by maximizing an acquisition criterion. This can be applied here to search for the optimal value within a given response time analysis such as [4].

## Contributions:

- We present a novel approach to find relative priority points for task sets with dynamic self-suspending behavior using MBO to optimize timing-related metrics such as the schedulability and the maximum lateness.
- A preliminary evaluation shows that the MBO approach can effectively overcome the limitations of GA since it shows better schedulability ratio for a wide range of

task sets parameters without requiring training. Furthermore, MBO outperforms the iterative approach not only in schedulability ratio but also reduces the maximum lateness significantly.

## II. SYSTEM MODEL

In this work, we consider a set  $\mathbb{T} = \{\tau_1, \dots, \tau_n\}$  of sporadic real-time tasks with dynamic self-suspending behavior. That is, each task  $\tau_i$  is described by a tuple  $(C_i, S_i, D_i, T_i)$ , where  $C_i > 0$  is the worst-case execution time (WCET),  $S_i \geq 0$  is the maximum suspension time,  $D_i > 0$  is the relative deadline, and  $T_i > 0$  is the minimum inter-arrival time. Each task  $\tau_i$  releases jobs  $\tau_{i,j}$  with  $j \in \mathbb{N} = \{1, 2, 3, \dots\}$  recurrently at times  $r_{i,j}$ , respecting the minimum inter-arrival time, i.e.,  $r_{i,j+1} \geq r_{i,j} + T_i$  for all  $j \in \mathbb{N}$ . After its release, each job  $\tau_{i,j}$  has to be executed for a certain amount of time  $c_{i,j} \leq C_i$  until its absolute deadline  $d_{i,j} = r_{i,j} + D_i$  is reached. During its execution, the job  $\tau_{i,j}$  can suspend arbitrarily often and for arbitrary durations, as long as the total suspension of that job does not exceed  $S_i$ . We say a job finishes at time  $f_{i,j}$  when it completes its execution. The lateness of job  $\tau_{i,j}$  is defined as  $Lat_{i,j} = f_{i,j} - d_{i,j}$ . Hence, whenever the lateness is non-positive, the job meets its deadline. The (overall) maximum lateness is the maximum lateness that can be observed in the schedule, i.e.,  $Lat = \max\{Lat_{i,j} \mid i = 1, \dots, n, j \in \mathbb{N}\}$ . We further denote by  $U_i = \frac{C_i}{T_i}$  the utilization of task  $\tau_i$ , and by  $U = \sum_{i=1}^n U_i$  the total utilization of task set  $\mathbb{T}$ . We say that the task set has implicit deadlines if  $D_i = T_i$  for all  $\tau_i \in \mathbb{T}$ , constrained deadlines if  $D_i \leq T_i$  for all  $\tau_i \in \mathbb{T}$ , and arbitrary deadlines if there are no constraints on the relation between deadline and minimum inter-arrival time.

To perform EDF-Like (EL) scheduling, each task  $\tau_i$  is equipped with an additional parameter  $\Pi_i$ , called its relative priority point. When a job  $\tau_{i,j}$  is released at time  $r_{i,j}$ , it is assigned an absolute priority point  $\pi_{i,j} = r_{i,j} + \Pi_i$ . The EL scheduling mechanism assigns the currently pending job with the lowest absolute priority points the highest priority.<sup>1</sup> We focus on the preemptive EL scheduling, where a newly released job  $\pi_{i',j'}$  with a lower absolute priority point than the currently executed job  $\pi_{i,j}$ , i.e.,  $\pi_{i',j'} < \pi_{i,j}$ , can preempt the currently executed job  $\tau_{i,j}$  and be executed instead. Figure 2 depicts our notation and a simple example where jobs are executed under EL scheduling.

## III. FINDING PRIORITY POINTS USING MBO

Model-based optimization (MBO) [6] is a method to optimize expensive objective functions. That is, given an objective function  $f: X \rightarrow \mathbb{R}$  with  $X \subseteq \mathbb{R}^d$ , MBO can be used for the optimization problem  $\arg \min_{x \in X} f(x)$ . For the optimization, MBO builds a surrogate to approximate the expensive black box function  $f$  based on an initial set  $\mathcal{D} = \{x_1, \dots, x_k\}$  of  $k$  points and the corresponding function values  $f(x_i)$  for  $x_i \in \mathcal{D}$ . MBO optimizes an acquisition function  $a: X \rightarrow \mathbb{R}$

<sup>1</sup>We assume that ties are broken arbitrarily but deterministically, e.g., if two jobs have the same absolute priority point then the job with the smaller task index obtains higher priority.

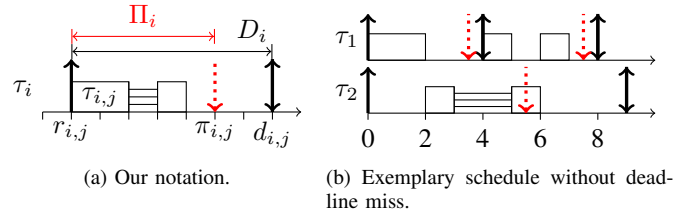


Figure 2. Notation for EL scheduling and an exemplary schedule.

based on the surrogate to determine the most promising point  $x^+ = \arg \min_{x \in X} a(x)$  and adds it to the surrogate, i.e.,  $\mathcal{D} := \mathcal{D} \cup \{x^+\}$ , until a predefined number  $i$  of iterations or an expected precision is reached. Afterwards,  $x^* \in X$  with the minimal value based on the final surrogate is returned as a result of the MBO optimization.

For the optimization problem of finding relative priority points to optimize the maximum lateness  $Lat$  (as defined in Section II), MBO is very suitable. That is, if we can calculate the maximum lateness  $Lat$  or an upper bound for that maximum lateness  $\tilde{Lat} \leq Lat$  from the list of relative priority points  $(\Pi_1, \dots, \Pi_n)$ , then we define this function as  $f: X \rightarrow \mathbb{R}$ , where  $X = \mathbb{R}^n$  is the set of relative priority points and  $f(\Pi_1, \dots, \Pi_n) = \tilde{Lat}$  is the upper bound on the maximum lateness given priority points  $(\Pi_1, \dots, \Pi_n)$ . When applying MBO, the result vector  $x^*$  yields priority points which optimize the guaranteed maximum lateness  $Lat$ .

To compute a bound  $\tilde{Lat}$  on the maximum lateness, given the relative priority points  $(\Pi_1, \dots, \Pi_n)$ , we rely on the response time analysis for dynamic self-suspending tasks under EL scheduling developed by Günzel et al. [4]<sup>2</sup>. That is, given relative priority points  $(\Pi_1, \dots, \Pi_n)$ , the response time analysis provides an upper bound  $\tilde{R}_i$  for each task  $\tau_i$ . Specifically, for each task  $\tau_i \in \mathbb{T}$ , we have  $\tilde{R}_i \geq f_{i,j} - r_{i,j}$  for all  $j \in \mathbb{N}$ . Consequently, the analysis provides a bound on the maximum lateness as  $Lat \leq \tilde{Lat} = \max_{\tau_i \in \mathbb{T}} \tilde{R}_i - D_i$ . The analysis relies on two configuration values  $\eta$  and  $depth$ , where  $\eta$  describes the granularity of the search for the response time bound and  $depth$  configures the number of improving runs. The higher these values are configured, the tighter the analysis, but also the more computationally expensive the computation of the response time bound. Hence, the capability of MBO to deal with expensive objective functions is a major benefit for our approach.

## IV. PRELIMINARY EVALUATION

For the preliminary evaluation, we generated multiple datasets, each containing 500 implicit-deadline task sets for each utilization  $U$  in the range  $U = 0.1, 0.2, \dots, 1.0$ , with different configurations of number of tasks  $n$  and with different ranges of maximum suspension and minimum inter-arrival time. (The specific configurations of the presented datasets are detailed below.) The process is as follows:

<sup>2</sup>Please note that we are restricting our attention to the constrained deadline case (i.e.,  $D_i \leq T_i$  for all tasks  $\tau_i$ ) for which the two approaches in [4] based on the fixed and the variable analysis window coincide.

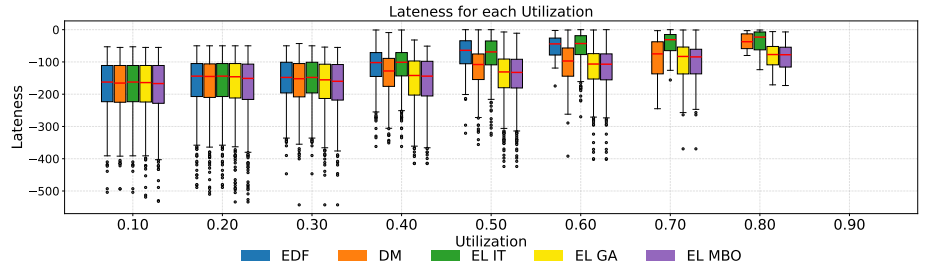
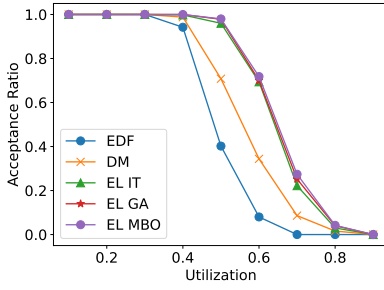


Figure 3. Same parameters as GA was trained on (Suspension uniform in  $[0, 0.5](T_i - C_i)$  and periods  $T_i$  log-uniform in  $[100, 10\,000]$ ).

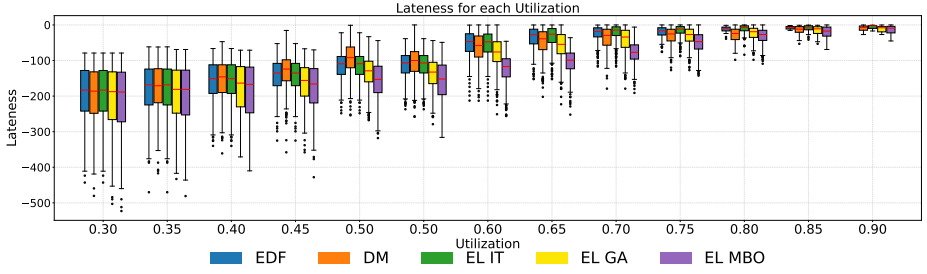
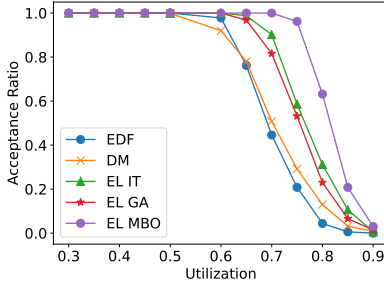


Figure 4. A positive example (Suspension uniform in  $[0.1, 0.1](T_i - C_i)$  and periods  $T_i$  log-uniform in  $[100, 1\,000]$ ).

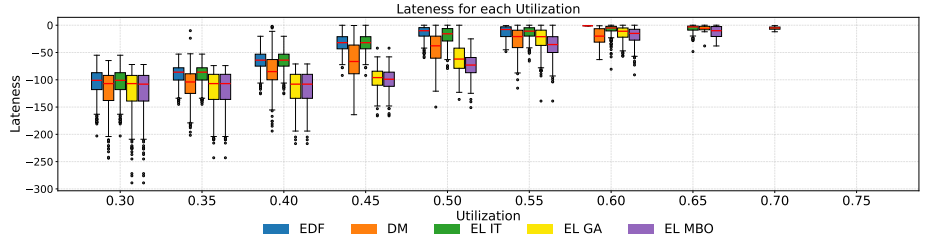
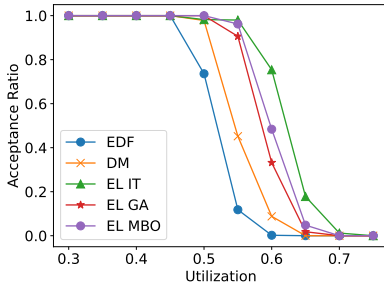


Figure 5. A negative example (Suspension uniform in  $[0.2, 0.2](T_i - C_i)$  and periods  $T_i$  log-uniform in  $[100, 1\,000]$ ).

- 1) First, for each task  $\tau_i$  a period  $T_i$  is drawn at random, as detailed in the dataset description.
- 2) Second, utilization values  $U_i$  with total sum  $\sum_{i=1}^n U_i = U$  are obtained using UUnifast [2].
- 3) Third, the suspension  $\leq (T_i - C_i)$  is drawn at random, as detailed in the dataset description.

Furthermore, the schedulability test [4] is configured with a step size of  $\eta = 0.01$  and the number of improving runs for the search algorithm  $depth = 3$ . For each dataset, we evaluate the following relative priority points:

- **EDF**: The relative priority points to achieve the typical EDF behavior, i.e.,  $\Pi_i = D_i$ .
- **DM**: The relative priority points to achieve the typical DM behavior (following [4, Section 3]), i.e.,  $\Pi_i = \sum_{j=1}^i D_j$  with the task set ordered by relative deadline.
- **EL IT**: The iterative approach presented in [5] with number of iterations set to 200.
- **EL GA**: The priority points of the highest performing chromosome obtained from the genetic algorithm pre-

sented in [5], i.e.,

$$\Pi_i = C_i^{0.42} \cdot S_i^{-0.10} \cdot T_i^{0.76} - 0.52 \cdot C_i - 3.79 \cdot S_i + 2.47 \cdot T_i \quad (1)$$

- **EL MBO**: The priority points achieved by MBO, as discussed in Section III, with the following configuration: The MBO uses the Gaussian Process as a surrogate model. For initialization of the model it is fitted with random sets of relative priority points  $(\Pi_1, \dots, \Pi_n)$  and the highest performing chromosome from Equation (1). The search-space of relative priority points for each task set is bounded by  $[0, \sum_i^n T_i]$ . For each task set, the MBO does 25 independent runs with 5 Bayesian optimization iterations each, using the Log Expected Improvement as acquisition function. The experiments are implemented in Python 3.11.2 using the BoTorch library version 0.14.0 [1].

As preliminary evaluation results, we present the results from three datasets:

- **Dataset 1**: In the first dataset, the same configuration as

in [5] is chosen, i.e., 10 tasks are generated for each task set, the suspension is drawn uniformly at random from the interval  $[0, 0.5](T_i - C_i)$ , and the periods  $T_i$  are drawn log-uniformly at random in the interval  $[100, 10\,000]$ . The results for this dataset are depicted in Figure 3.

- **Dataset 2:** In the second dataset, 5 tasks are generated for each task set, the suspension is drawn uniformly at random from the interval  $[0.1, 0.1](T_i - C_i)$ , and the periods  $T_i$  are drawn log-uniformly at random in the interval  $[100, 1\,000]$ . The results for this dataset are depicted in Figure 4.
- **Dataset 3:** In the third dataset, 15 tasks are generated for each task set, the suspension is drawn uniformly at random from the interval  $[0.2, 0.2](T_i - C_i)$ , and the periods  $T_i$  are drawn log-uniformly at random in the interval  $[100, 1\,000]$ . The results for this dataset are depicted in Figure 5.

For each dataset, we report the acceptance ratio, i.e., the ratio of task sets for which a schedulable configuration has been found (i.e.,  $\tilde{Lat} \leq 0$ ). Furthermore, we show the guaranteed lateness in the form of a boxplot.

We observe that EL MBO performs reasonably well in all three scenarios. For Dataset 1, the performance of EL MBO is similar to EL IT and EL GA. Further, there are configurations where EL MBO clearly outperforms the previous approach (see Dataset 2), but there are also configurations where the current version of EL MBO cannot reach the acceptance ratio achieved by EL IT (see Dataset 3).

We note that EL GA was trained on the task set configuration of Dataset 1, and therefore it is understandable that the acceptance ratio drops when EL GA is applied to Datasets 2 and 3. Retraining EL GA on other Datasets is beyond the scope of this preliminary evaluation.

Furthermore, although the acceptance ratio for EL MBO on Dataset 3 is below that for EL IT, we can see that EL MBO clearly improves the maximum lateness. The reason is that EL IT is built to push the maximum lateness as close as possible to 0 to give other tasks sufficient slack to finish until their deadline as well (cf. [5]).

Despite providing more precise results in some cases, the runtime of MBO is significantly higher compared to previous approaches. Specifically, for Dataset 1, MBO took 21323 seconds while the iterative approach finished in 344 seconds.<sup>3</sup> For Dataset 2, MBO finished after 8243 seconds and EL IT in 81 seconds, and for Dataset 3, MBO took 39492 seconds while the iterative approach only required 1150 seconds.

## V. OPEN PROBLEMS

While we have demonstrated the potential of MBO to determine relative priority points for optimizing the maximum lateness and hence the schedulability, there are still multiple open questions to be addressed.

First, it remains unclear how to effectively **configure the hyperparameters of MBO** to achieve the best results. While

<sup>3</sup>Runtime of the other approaches are not reported as they are negligible in comparison.

our evaluation mainly relies on a standard configuration, there are multiple parameters that can potentially be tuned further, such as the surrogate model, the acquisition function and its further optimization, Monte Carlo samplers or constraints on the input space. Further exploration has to be conducted to find more suitable configurations, and the full potential of MBO has yet to be realized.

Second, since we observe (in Figure 5) that the iterative approach EL IT has an advantage over EL MBO in this case, this raises the immediate question whether MBO can benefit from the insights used in the iterative approach. That is, the iterative approach is built upon [5, Observation 1], which states that *increasing the relative priority point  $\Pi_i$  of a task by a value  $X > 0$  cannot increase its worst-case response time by more than  $X$* . Whether this observation of the **iterative approach can be exploited to guide MBO** further remains as an open problem.

Third, while this exploration of relative priority points builds upon the analysis provided by Günzel et al. [4], recently, **different analyses** have been provided by Wang et al. [8]. Specifically, the authors propose an alternative response time analysis and a utilization bound. While both of them do not dominate the analysis that is provided in [4], integrating them into the lateness function  $\tilde{Lat}$  could impact the performance of MBO and the other approaches. The extent of this impact requires further investigation.

## VI. CONCLUSION

This work investigates the optimization of the scheduling algorithm to minimize the maximum lateness and hence to optimize task sets for schedulability. To that end, we employ EDF-Like (EL) scheduling and explore the configuration of priority points to optimize the task set. Although, previous work suggests two approaches to choose priority points solely to improve the schedulability ratio, these approaches either rely on training or are unsuitable for minimizing the maximum lateness. In this work, we propose the usage of Model-Based Optimization (MBO) to determine relative priority points that minimize the maximum lateness. While the preliminary evaluation demonstrates that our approach can achieve promising results, we identify multiple open problems that have to be addressed before its full potential can be realized.

## ACKNOWLEDGMENTS

This result is part of a project (PropRT) that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No. 865170).

## REFERENCES

- [1] M. Balandat, B. Karrer, D. R. Jiang, S. Daulton, B. Letham, A. G. Wilson, and E. Bakshy. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*, 2020.
- [2] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real Time Syst.*, 30(1-2):129–154, 2005.

- [3] J. Chen, T. Hahn, R. Hoeksma, N. Megow, and G. von der Brüggen. Scheduling self-suspending tasks: New and old results. In *ECRTS*, volume 133 of *LIPICs*, pages 16:1–16:23. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
- [4] M. Günzel, G. von der Brüggen, K. Chen, and J. Chen. EDF-like scheduling for self-suspending real-time tasks. In *RTSS*, pages 172–184. IEEE, 2022.
- [5] M. Günzel, K.-H. Chen, J.-J. Chen, and C.-C. Lin. Priority point exploration in edf-like scheduling for self-suspending tasks. In *Workshop on OPTimization for Embedded and ReAl-time systems (OPERA) co-located with the 44th IEEE Real-Time Systems Symposium (RTSS)*, 2023.
- [6] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.
- [7] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, 1973.
- [8] Y. Wang, B. Lv, Q. Zhou, J. Li, and T. Tan. Schedulability analysis for self-suspending tasks under edf-like scheduling. *IEEE Trans. Computers*, 74(7):2364–2375, 2025.