$\mathbf{k^2U}$: A General Framework from k-Point Effective Schedulability Analysis to Utilization-Based Tests

Jian-Jia Chen and Wen-Hung Huang
Department of Informatics
TU Dortmund University, Germany

Cong Liu
Department of Computer Science
The University of Texas at Dallas

***Abstract*—To deal with a large variety of workloads in different application domains in real-time embedded systems, a number of expressive task models have been developed. For each individual task model, researchers tend to develop different types of techniques for deriving schedulability tests with different computation complexity and performance. In this paper, we present a general schedulability analysis framework, namely the $\mathbf{k^2U}$ framework, that can be potentially applied to analyze a large set of real-time task models under any fixed-priority scheduling algorithm, on both uniprocessor and multiprocessor scheduling. The key to $\mathbf{k^2U}$ is a k-point effective schedulability test, which can be viewed as a "blackbox" interface. For any task model, if a corresponding k-point effective schedulability test can be constructed, then a sufficient utilization-based test can be automatically derived. We show the generality of $\mathbf{k^2U}$ by applying it to different task models, which results in new and improved tests compared to the state-of-the-art.**

1 Introduction

Given the emerging trend towards building complex cyber-physical systems that often integrate external and physical devices, many real-time and embedded systems are expected to handle a large variety of workloads. Different formal real-time task models have been developed to accurately represent these workloads with various characteristics. Examples include the sporadic task model [28], the multi-frame task model [29], the self-suspending task model [23], the directed-acyclic-graph (DAG) task model, etc. Many of such formal models have been shown to be expressive enough to accurately model real systems in practice. For example, the DAG task model has been used to represent many computation-parallel multimedia application systems and the self-suspending task model is suitable to model workloads that may interact with I/O devices. For each of these task models, researchers tend to develop different types of techniques that result in schedulability tests with different computation complexity and performance (e.g., different utilization bounds).

In this paper, we present $\mathbf{k^2U}$, a general schedulability analysis framework that is fundamentally based on a k-point effective schedulability test under fixed-priority scheduling. The key observation behind our proposed k-point test is the following. Traditional fixed-priority schedulability tests often have pseudo-polynomial-time (or even higher) complexity. For example, to verify the schedulability of a (constrained-deadline) task τ_k under fixed-priority scheduling in uniprocessor systems, the time-demand analysis (TDA) developed in [20] can be adopted. That is, if

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

then task τ_k is schedulable under the fixed-priority scheduling algorithm, where $hp(\tau_k)$ is the set of the tasks with higher priority than τ_k, D_k, C_k, and T_i represent τ_k's relative deadline, worst-case execution time, and period, respectively. TDA incurs pseudo-polynomial-time complexity to check the time points that lie in $(0, D_k]$ for Eq. (1).

To obtain sufficient schedulability tests under fixed priority scheduling with reduced time complexity (e.g., polynomial-time), our conceptual innovation is based on the observations by *testing only a subset of such points* to *derive the minimum C_k that cannot pass the schedulability tests*. This idea is implemented in the $\mathbf{k^2U}$ framework by providing a general k-point effective schedulability test, which only needs to test k points under *any* fixed-priority scheduling when checking schedulability of the task with the k^{th} highest priority in the system. This k-point effective schedulability test can be viewed as a "blackbox" interface that can result in sufficient utilization-based tests. We show the generality of $\mathbf{k^2U}$ by applying it to analyze several concrete example task models, including the constrained- and arbitrary-deadline sporadic task models, the multi-frame task model, the self-suspending task model, and the DAG task model. Note that $\mathbf{k^2U}$ is not only applicable to uniprocessor systems, but also applicable to multiprocessor systems.

Related Work. There have been several results in the literature with respect to utilization-based, e.g., [6], [16]–[18], [24], [25], [31], schedulability tests for the sporadic real-time task model and its generalizations in uniprocessor systems. Most of the existing utilization-based schedulability analyses focus on the total utilization bound. That is, if the total utilization of the task system is no more than the derived bound, the task system is schedulable by the scheduling policy. For example, the total utilization bounds derived in [8], [16], [25] are mainly for rate-monotonic (RM) scheduling, in which the results in [16] can be extended for arbitrary fixed-priority scheduling. Kuo et al. [17] further improve the total utilization bound by using the notion of divisibility. Lee et al. [18] use linear programming formulations for calculating total utilization bounds when the period of a task can be selected. Moreover, Wu et al. [31] adopt the Network Calculus to analyze the total utilization bounds of several task models.

The novelty of $\mathbf{k^2U}$ comes from a different perspective from these approaches [16]–[18], [25], [31]. We do not specifically seek for the total utilization bound. Instead, we look for the critical value in the specified sufficient schedulability test while verifying the schedulability of task τ_k. A natural schedulability condition to express the schedulability of task τ_k is a hyperbolic bound, (to be shown in Lemma 1), whereas the corresponding total utilization bound can be obtained (in

Lemmas 2 and 3).

The hyperbolic forms are the centric features in $\mathbf{k^2U}$ analysis, in which the test by Bini et al. [6] for sporadic real-time tasks and our recent result in [24] for bursty-interference analysis are both special cases and simple implications from the $\mathbf{k^2U}$ framework. With the hyperbolic forms, we are then able to provide many interesting observations with respect to the required quantitive features to be measured, like the total utilization bounds, speed-up factors, etc., not only for uniprocessor scheduling but also for multiprocessor scheduling. For more details, we will provide further explanations at the end of Sec. 4 after the framework is presented. For the studied task models to demonstrate the applicability of $\mathbf{k^2U}$, we will summarize some of the latest results on these task models in their corresponding sections.

Contributions. In this paper, we present a general schedulability analysis framework, $\mathbf{k^2U}$, that can be applied to analyze a number of complex real-time task models, on both uniprocessors and multiprocessors. For any task model, if a corresponding k-point effective schedulability test can be constructed, then a sufficient utilization-based test can be derived by the $\mathbf{k^2U}$ framework. We show the generality of $\mathbf{k^2U}$ by applying it to several task models, in which the results are better or more general compared to the state-of-the-art:

1) For uniprocessor constrained-deadline sporadic task systems, the speed-up factor of our obtained schedulability test is 1.76322. This value is the same as the lower bound and upper bound of deadline-monotonic (DM) scheduling shown by Davis et al. [14]. Our result is thus stronger (and requires a much simpler proof), as we show that the same factor holds for a polynomial-time schedulability test (not just the DM scheduler). For uniprocessor arbitrary-deadline sporadic task systems, our obtained utilization-based test works for any fixed-priority scheduling with arbitrary priority-ordering assignment.
2) For multiprocessor DAG task systems under global rate-monotonic (RM) scheduling, the capacity-augmentation factor, as defined in [22] and Sec. 6 in this paper, of our obtained test is 3.62143. This result is better than the best existing result, which is 3.73, given by Li et al. [22]. Our result is also applicable for conditional sporadic DAG task systems [3].
3) For multiprocessor self-suspending task systems, we obtain the *first* utilization-based test for global RM.
4) For uniprocessor multi-frame task systems, our obtained utilization bound is superior to the results by Mok and Chen [29] analytically and Lu et al. [26] in our simulations. Due to space limitations, the analysis is in Appendix C in the report [12], whereas the evaluation result is in Appendix D in [12].

Note that the emphasis of this paper is not to show that the resulting tests for different task models by applying the $\mathbf{k^2U}$ framework are better than existing work. Rather, we want to show that the $\mathbf{k^2U}$ framework is general, easy to use, and has relatively low time complexity, but is still able to generate good tests. By demonstrating the applicability of the $\mathbf{k^2U}$ framework to several task models, we believe that this framework has great potential in analyzing many other complex real-time task models, where the existing analysis approaches are insufficient or cumbersome. To the best of our knowledge, together with $\mathbf{k^2Q}$ to be explained later, these are the first general schedulability analysis frameworks that can be potentially applied to analyze a large set of real-time task models under any fixed-priority scheduling algorithm in both uniprocessor and multiprocessor systems.

Comparison to $\mathbf{k^2Q}$: The concept of testing k points only is also the key in another framework designed by us, called $\mathbf{k^2Q}$ [11]. Even though $\mathbf{k^2Q}$ and $\mathbf{k^2U}$ share the same idea by testing and evaluating only k points, they are based on completely different criteria for testing. In $\mathbf{k^2U}$, all the testings and formulations are based on *only the higher-priority task utilizations*. In $\mathbf{k^2Q}$, the testings are based *not only on the higher-priority task utilizations, but also on the higher-priority task execution times*. The above difference in the formulations results in completely different properties and mathematical closed-forms. The natural schedulability condition of $\mathbf{k^2U}$ is a *hyperbolic form* for testing the schedulability, whereas the natural schedulability condition of $\mathbf{k^2Q}$ is a *quadratic form* for testing the schedulability or the response time of a task.

If one framework were dominated by another or these two frameworks were just with minor difference in mathematical formulations, it wouldn't be necessary to separate and present them as two different frameworks. Both frameworks are in fact needed and have to be applied for different cases. Due to space limitation, we can only shortly explain their differences, advantages, and disadvantages in this paper. For completeness, another document has been prepared in [10] to present the similarity, the difference and the characteristics of these two frameworks in details.

Since the formulation of $\mathbf{k^2U}$ is more restrictive than $\mathbf{k^2Q}$, its applicability is limited by the possibility to formulate the tests purely by using higher-priority task utilizations without referring to their execution times. There are cases, in which formulating the higher-priority interference by using only task utilizations for $\mathbf{k^2U}$ is troublesome. For such cases, further introducing the upper bound of the execution time by using $\mathbf{k^2Q}$ is more precise. In general, if we can formulate the schedulability tests into the $\mathbf{k^2U}$ framework by purely using higher-priority task utilizations, it is also usually possible to formulate it into the $\mathbf{k^2Q}$ framework by further introducing the task execution times. In such cases, the same pseudo-polynomial-time (or exponential time) test is used, and the utilization bound or speed-up factor analysis derived from the $\mathbf{k^2U}$ framework is, in general, tighter and better.

2 Sporadic Task and Scheduling Models

A sporadic task τ_i is released repeatedly, with each such invocation called a job. The j^{th} job of τ_i, denoted $\tau_{i,j}$, is released at time $r_{i,j}$ and has an absolute deadline at time $d_{i,j}$. Each job of any task τ_i is assumed to have execution time C_i. Here in this paper, whenever we refer to the execution time of a job, we mean for the worst-case execution time of the job since all the analyses we use are safe by only considering the worst-case execution time. Successive jobs of the same task are required to execute in sequence. Associated with each task τ_i are a period T_i, which specifies the minimum time between two consecutive job releases of τ_i, and a deadline D_i, which specifies the relative deadline of each such job, i.e., $d_{i,j} = r_{i,j} + D_i$. The utilization of a task τ_i is defined as $U_i = C_i/T_i$.

A sporadic task system τ is said to be an implicit-deadline system if $D_i = T_i$ holds for each τ_i. A sporadic task system τ is said to be a constrained-deadline system if $D_i \leq T_i$ holds for each τ_i. Otherwise, such a sporadic task system τ is an arbitrary-deadline system.

A task is said *schedulable* by a scheduling policy if all of its jobs can finish before their absolute deadlines. A task system is said *schedulable* by a scheduling policy if all the tasks in the task system are schedulable. A *schedulability test* expresses sufficient conditions to ensure the feasibility of the resulting schedule by a scheduling policy.

Throughout the paper, we will focus on fixed-priority preemptive scheduling. That is, each task is associated with a priority level. For a uniprocessor system, i.e., except Sec. 6, the scheduler always dispatches the job with the highest priority in the ready queue to be executed. For a uniprocessor system, it has been shown that RM scheduling is an optimal fixed-priority scheduling policy for implicit-deadline systems [25] and DM scheduling is an optimal fixed-priority scheduling policy for constrained-deadline systems [21].

To verify the schedulability of task τ_k under fixed-priority scheduling in uniprocessor systems, the time-demand analysis developed in [20] can be adopted, as discussed earlier. That is, if Eq. (1) holds, then task τ_k is schedulable under the fixed-priority scheduling algorithm. For the simplicity of presentation, we will demonstrate how the framework works using ordinary sporadic real-time task systems in Sec. 4 and Sec. 5. We will demonstrate more applications with respect to multi-frame tasks [29] in Appendix C in the report [12] and with respect to multiprocessor scheduling in Sec. 6.

3 Analysis Flow

The proposed $\mathbf{k^2U}$ framework only tests the schedulability of a specific task τ_k, under the assumption that the higher-priority tasks are already verified to be schedulable by the given scheduling policy. Therefore, this framework has to be applied for each of the given tasks. A task system is schedulable by the given scheduling policy only when all the tasks in the system can be verified to meet their deadlines. The results can be extended to test the schedulability of a task system in linear time complexity or to allow on-line admission control in constant time complexity if the schedulability condition (or with some more pessimistic simplifications) is monotonic. Such extensions are provided in Appendix A for some cases.

Therefore, for the rest of this paper, we implicitly assume that all the higher-priority tasks are already verified to be schedulable by the scheduling policy. We will only present the schedulability test of a certain task τ_k, that is being analyzed, under the above assumption. For notational brevity, we implicitly assume that there are $k-1$ tasks, say $\tau_1, \tau_2, \ldots, \tau_{k-1}$ with higher-priority than task τ_k. Moreover, we only consider the cases when $k \geq 2$, since $k = 1$ is pretty trivial.

4 $\mathbf{k^2U}$ Framework

This section presents the basic properties of the $\mathbf{k^2U}$ framework for testing the schedulability of task τ_k in a given set of real-time tasks (depending on the specific models given in each application as shown later in this paper). We will first provide examples to explain and define the *k-point effective schedulability test*. Then, we will provide the fundamental properties of the corresponding utilization-based tests. Throughout this section, we will implicitly use sporadic task systems defined in Sec. 2 to simplify the presentation. The concrete applications will be presented in Secs. 5 - 6.

The k-point effective schedulability test is a sufficient schedulability test that verifies only k time points, defined by the $k-1$ higher-priority tasks and task τ_k. For example, instead of testing all the possible t in the range of 0 and D_k in Eq. (1), we can simply test only k points. It may seem to be very pessimistic to only test k points. However, if these k points are *effective*,[1] the resulting schedulability test may be already good. We now demonstrate two examples.

Example 1. Implicit-deadline task systems*: Suppose that the tasks are indexed by the periods, i.e., $T_1 \leq \cdots \leq T_k$. When $T_k \leq 2T_1$, task τ_k is schedulable by RM if there exists $j \in \{1, 2, \ldots, k\}$ where*

$$C_k + \sum_{i=1}^{k-1} C_i + \sum_{i=1}^{j-1} C_i = C_k + \sum_{i=1}^{k-1} T_i U_i + \sum_{i=1}^{j-1} T_i U_i \leq T_j. \quad (2)$$

■

That is, in the above example, it is sufficient to only test $T_1, T_2, \ldots, T_k$. The case defined in the above example is utilized by Liu and Layland [25] for deriving the least utilization upper bound 69.3% for RM scheduling. We can make the above example more generalized as follows:

Example 2. Implicit-deadline task systems with given ratios of periods*: Suppose that $fT_i \leq T_k$ for a given integer f with $f \geq 1$ for any higher-priority task τ_i, for all $i = 1, 2, \ldots, k-1$. Let t_i be $\left\lfloor \frac{T_k}{T_i} \right\rfloor T_i$. Suppose that the $k-1$ higher priority tasks are indexed such that $t_1 \leq t_2 \leq \cdots \leq t_{k-1} \leq t_k$, where t_k is defined as T_k. Task τ_k is schedulable under RM if there exists j with $1 \leq j \leq k$ such that*

$$C_k + \sum_{i=1}^{k-1} t_i U_i + \sum_{i=1}^{j-1} C_i \leq C_k + \sum_{i=1}^{k-1} t_i U_i + \sum_{i=1}^{j-1} \frac{1}{f} t_i U_i \leq t_j, \quad (3)$$

where the first inequality in Eq. (3) *is due to the fact $C_i = T_i U_i \leq \frac{1}{f} t_i U_i$. That is, in the above example, it is sufficient to only test $\left\lfloor \frac{T_k}{T_1} \right\rfloor T_1, \left\lfloor \frac{T_k}{T_2} \right\rfloor T_2, \ldots, \left\lfloor \frac{T_k}{T_{k-1}} \right\rfloor T_{k-1}, T_k$.* ■

With the above examples, for a given set $\{t_1, t_2, \ldots t_k\}$, we now define the k-point effective schedulability test as follows:

Definition 1. *A k-point effective schedulability test is a sufficient schedulability test of a fixed-priority scheduling policy, that verifies the existence of $t_j \in \{t_1, t_2, \ldots t_k\}$ with $0 < t_1 \leq t_2 \leq \cdots \leq t_k$ such that*

$$C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i t_i U_i \leq t_j, \quad (4)$$

where $C_k > 0$, $\alpha_i > 0$, $U_i > 0$, and $\beta_i > 0$ are dependent upon the setting of the task models and task τ_i.

[1] As to be clearly illustrated later, the k points can be considered effective if they can define certain extreme cases of task parameters. For example, the "difficult-to-schedule" concept first introduced by Liu and Layland [25] defines k effective points that are used in Example 1. In their case [25], the selected set of the k points was "very" effective because the tested task τ_k becomes unschedulable if C_k is increased by an arbitrarily small value ε.

For Example 1, the effective values in $\{t_1, t_2, \dots t_k\}$ are $T_1, T_2, \dots, T_k$, and $\alpha_i = \beta_i = 1$ for each task τ_i. For Example 2, the effective values in $\{t_1, t_2, \dots t_k\}$ are with $\alpha_i = 1$ and $\beta_i \leq \frac{1}{f}$ for each task τ_i.

Moreover, we only consider non-trivial cases, in which $C_k > 0$, $t_k > 0$, $0 < \alpha_i$, $0 < \beta_i$, and $0 < U_i \leq 1$ for $i = 1, 2, \dots, k-1$. The definition of the k-point last-release schedulability test $C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i t_i U_i \leq t_j$ in Definition 1 only slightly differs from the test $C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i C_i \leq t_j$ in the $\mathbf{k^2Q}$ framework [11]. However, since the tests are different, they are used for different situations.

With these k points, we are able to define the corresponding coefficients α_i and β_i in the k-point effective schedulability test of a scheduling algorithm. The elegance of the $\mathbf{k^2U}$ framework is to use only the parameters α_i and β_i to analyze whether task τ_k can pass the schedulability test. Therefore, the $\mathbf{k^2U}$ framework provides corresponding utilization-based tests *automatically* if the k-point effective schedulability test and the corresponding parameters α_i and β_i can be defined, which will be further demonstrated in the following sections with several applications.

We are going to present the properties resulting from the k-point effective schedulability test under given α_i and β_i. In the following lemmas, we are going to seek the extreme cases for these k testing points under the given setting of utilizations and the defined coefficients α_i and β_i. To make the notations clear, these extreme testing points are denoted as t_i^* for the rest of this paper. The procedure will derive $k-1$ extreme testing points, denoted as $t_1^*, t_2^*, \dots, t_{k-1}^*$, whereas t_k^* is defined as t_k plus a slack variable (to be defined in the proof of Lemma 1) for notational brevity. Lemmas 1 to 3 are useful to analyze the utilization bound, the hyperbolic bound, etc., for given scheduling strategies, when α and β can be easily defined based on the scheduling policy, with $0 < t_k$ and $0 < \alpha_i \leq \alpha$, and $0 < \beta_i \leq \beta$ for any $i = 1, 2, \dots, k-1$.

Lemma 1. *For a given k-point effective schedulability test of a scheduling algorithm, defined in Definition 1, in which $0 < t_k$ and $0 < \alpha_i \leq \alpha$, and $0 < \beta_i \leq \beta$ for any $i = 1, 2, \dots, k-1$, task τ_k is schedulable by the scheduling algorithm if the following condition holds*

$$\frac{C_k}{t_k} \leq \frac{\frac{\alpha}{\beta} + 1}{\prod_{j=1}^{k-1}(\beta U_j + 1)} - \frac{\alpha}{\beta}. \tag{5}$$

Proof: If $\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta} \leq 0$, the condition in Eq. (5) never holds since $C_k > 0$, and the statement is vacuously true. We focus on the case $\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta} > 0$.

We prove this lemma by showing that the condition in Eq. (5) leads to the satisfactions of the schedulability condition listed in Eq. (4) by using contrapositive. By taking the negation of the schedulability condition in Eq. (4), we know that if task τ_k is *not schedulable* by the scheduling policy, then for each $j = 1, 2, \dots, k$

$$C_k + \alpha \sum_{i=1}^{k-1} t_i U_i + \beta \sum_{i=1}^{j-1} t_i U_i \geq C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i t_i U_i > t_j, \tag{6}$$

due to $0 < \alpha_i \leq \alpha$, and $0 < \beta_i \leq \beta$ for any $i = 1, 2, \dots, k-1$ To enforce the condition in Eq. (6), we are going to show that C_k must have some lower bound. Therefore, if C_k is no more than this lower bound, then task τ_k is schedulable by the scheduling policy.

For the rest of the proof, we replace $>$ with $\geq$ in Eq. (6), as the infimum and the minimum are the same when presenting the inequality with $\geq$. Moreover, we also relax the problem by replacing the constraint $t_{j+1} \geq t_j$ with $t_j \geq 0$ for $j = 1, 2, \dots, k-1$. Therefore, the unschedulability for satisfying Eq. (6) implies that $C_k > C_k^*$, where C_k^* is defined in the following optimization problem:

$$\min \ C_k^* \tag{7a}$$

$$\text{s.t. } C_k^* + \sum_{i=1}^{k-1} \alpha t_i^* U_i + \sum_{i=1}^{j-1} \beta t_i^* U_i \geq t_j^* \quad \forall j = 1, \dots, k-1, \tag{7b}$$

$$t_j^* \geq 0 \qquad \forall j = 1, \dots, k-1, \tag{7c}$$

$$C_k^* + \sum_{i=1}^{k-1} (\alpha + \beta) t_i^* U_i \geq t_k, \tag{7d}$$

where $t_1^*, t_2^*, \dots, t_{k-1}^*$ and C_k^* are variables; and α, β are constants.

Let $s \geq 0$ be a slack variable such that $C_k^* + \sum_{i=1}^{k-1}(\alpha + \beta) t_i^* U_i = t_k + s$. Therefore, $C_k^* = t_k + s - \sum_{i=1}^{k-1}(\alpha+\beta) t_i^* U_i$. By replacing C_k^* in Eq. (7b), we have

$$\begin{aligned} & t_k + s - (\sum_{i=1}^{k-1} \alpha t_i^* U_i + \sum_{i=1}^{k-1} \beta t_i^* U_i) + \sum_{i=1}^{k-1} \alpha t_i^* U_i + \sum_{i=1}^{j-1} \beta t_i^* U_i \\ = & \ t_k + s - \sum_{i=j}^{k-1} \beta t_i^* U_i \geq t_j^*, \quad \forall j = 1, \dots, k-1. \end{aligned} \tag{8}$$

For notational brevity, let t_k^* be $t_k + s$. Therefore, the linear programming in Eq. (7) can be rewritten as follows:

$$\min \ t_k^* - \sum_{i=1}^{k-1} (\alpha + \beta) t_i^* U_i \tag{9a}$$

$$\text{s.t. } t_k^* - \beta \sum_{i=j}^{k-1} t_i^* U_i \geq t_j^*, \qquad \forall 1 \leq j \leq k-1 \tag{9b}$$

$$t_j^* \geq 0 \qquad \forall 1 \leq j \leq k-1 \tag{9c}$$

$$t_k^* \geq t_k \tag{9d}$$

The remaining proof is to solve the above linear programming to obtain the minimum C_k^* if $\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta} > 0$. Our proof strategy is to solve the linear programming analytically as a function of t_k^*. This can be imagined as if t_k^* is given. At the end, we will prove the optimality by considering all possible $t_k^* \geq t_k$. This involves three steps:

- Step 1: we analyze certain properties of optimal solutions based on the extreme point theorem for linear programming [27] under the assumption that t_k^* is given as a constant, i.e., s is known.
- Step 2: we present a specific solution in an *extreme point*, as a function of t_k^*.
- Step 3: we prove that the above extreme point solution gives the minimum C_k^* if $\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta} > 0$.

[Step 1:] In this step, we assume that s is given, i.e., t_k^* is specified as a constant. The original linear programming in Eq. (9) has k variables and $2(k-1)+1$ constraints. After specifying the value t_k^* as a given constant, the new linear programming without the constraint in Eq. (9d) has only $k-1$ variables and $2(k-1)$ constraints. Thus, according to the extreme point theorem for linear programming [27], the linear constraints form a polyhedron of feasible solutions. The extreme point theorem states that either there is no feasible solution or one of the extreme points in the polyhedron is an optimal solution when the objective of the linear programming is finite. To satisfy Eqs. (9b) and (9c), we know that $t_j^* \le t_k^*$ for $j = 1, 2, \ldots, k-1$, due to $t_i^* \ge 0$, $0 < \beta$, $0 < \alpha$ and $U_i > 0$ for $i = 1, 2, \ldots, k-1$. As a result, the objective of the above linear programming is $t_k^* - \sum_{i=1}^{k-1}(\alpha+\beta)t_i^*U_i \ge t_k^* - \sum_{i=1}^{k-1}(\alpha+\beta)t_k^*U_i$, which is finite (as a function of t_k^*, α, and β) under the assumption $0 < \beta$, $0 < \alpha$ and $0 < \sum_{i=1}^{k-1} U_i \le 1$.

According to the extreme point theorem, one of the extreme points is the optimal solution of Eq. (9) for a given t_k^*. There are $k-1$ variables with $2(k-1)$ constraints in Eq. (9) for a given t_k^*. An extreme point must have at least $k-1$ *active* constraints in Eqs. (9b) and (9c), in which their $\ge$ are set to equality $=$.

We now prove that an extreme point solution is feasible for Eq. (9) by setting t_j^* either to 0 or to $t_k^* - \beta\sum_{i=j}^{k-1} t_i^*U_i = t_j^* > 0$ for $j = 1, 2, \ldots, k-1$. Suppose for contradiction that there exists a task τ_h with $0 = t_h^* = t_k^* - \beta\sum_{i=h}^{k-1} t_i^*U_i$. Let ω be the index of the next task after τ_h with $t_\omega^* > 0$ in this extreme point solution, i.e., $t_h^* = t_{h+1}^* = \cdots = t_{\omega-1}^* = 0$. If $t_i^* = 0$ for $h \le i \le k-1$, then ω is set to k and t_ω^* is t_k^*. Therefore, the conditions $t_k^* - \beta\sum_{i=h}^{k-1} t_i^*U_i = t_h^* = 0$ and $t_{h+1}^* = \cdots = t_{\omega-1}^* = 0$ imply the contradiction that $0 = t_h^* = t_k^* - \beta\sum_{i=h}^{k-1} t_i^*U_i = t_k^* - \beta\sum_{i=\omega}^{k-1} t_i^*U_i \ge t_\omega^* > 0$ when $\omega \le k-1$ or $0 = t_h^* = t_k^* - \sum_{i=h}^{k-1} t_i^*U_i = t_k^* > 0$ when $\omega = k$.

By the above analysis and the pigeonhole principle, a feasible extreme point solution of Eq. (9) can be represented by two sets $\mathbf{T}_1$ and $\mathbf{T}_2$ of the $k-1$ higher-priority tasks, in which $t_j^* = 0$ if τ_j is in $\mathbf{T}_1$ and $t_k^* - \beta\sum_{i=j}^{k-1} t_i^*U_i = t_j^* > 0$ if task τ_j is in $\mathbf{T}_2$. With the above discussions, we have $\mathbf{T}_1 \cup \mathbf{T}_2 = \{\tau_1, \tau_2, \ldots, \tau_{k-1}\}$ and $\mathbf{T}_1 \cap \mathbf{T}_2 = \emptyset$.

[Step 2:] For a given t_k^*, one special extreme point solution is to put $t_k^* - \beta\sum_{i=j}^{k-1} t_i^*U_i = t_j^*$ for every $j = 1, 2, \ldots, k-1$, i.e., $\mathbf{T}_1$ as an empty set. Therefore,

$$\forall 1 \le i \le k-1, \quad t_{i+1}^* - t_i^* = \beta t_i^* U_i, \tag{10}$$

which implies that

$$\frac{t_{i+1}^*}{t_i^*} = \beta U_i + 1. \tag{11}$$

Moreover,

$$\frac{t_i^*}{t_k^*} = \prod_{j=i}^{k-1} \frac{t_j^*}{t_{j+1}^*} = \frac{1}{\prod_{j=i}^{k-1}(\beta U_j + 1)}. \tag{12}$$

The above extreme point solution is always feasible in the linear programming of Eq. (9) since

$$0 < \frac{t_i^*}{t_k^*}, \quad \forall i = 1, 2, \ldots, k-1. \tag{13}$$

By Eq. (10), we have

$$(\alpha+\beta)U_i t_i^* = (t_{i+1}^* - t_i^*)(\frac{\alpha}{\beta}+1). \tag{14}$$

Therefore, in this extreme point solution, the objective function of Eq. (9) is

$$\begin{aligned} &t_k^* - \sum_{i=1}^{k-1}(\alpha+\beta)U_i t_i^* = t_k^* - (t_k^* - t_1^*)(\frac{\alpha}{\beta}+1) \\ =&t_k^*\left(\frac{t_1^*}{t_k^*}(\frac{\alpha}{\beta}+1) - \frac{\alpha}{\beta}\right) \overset{1}{=} t_k^*\left(\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta}\right), \end{aligned} \tag{15}$$

where the last equality $\overset{1}{=}$ is due to Eq. (12) when i is 1.

[Step 3:] From Step 1, a feasible extreme point solution is with $\mathbf{T}_1 \cup \mathbf{T}_2 = \{\tau_1, \tau_2, \ldots, \tau_{k-1}\}$ and $\mathbf{T}_1 \cap \mathbf{T}_2 = \emptyset$. As a result, we can simply drop all the tasks in $\mathbf{T}_1$ and use the remaining tasks in $\mathbf{T}_2$ by adopting the same procedures from Eq. (10) to Eq. (14) in Step 2. The resulting objective function of this extreme point solution for the linear programming in Eq. (9) is $t_k^*(\frac{\frac{\alpha}{\beta}+1}{\prod_{\tau_j\in\mathbf{T}_2}(\beta U_j+1)} - \frac{\alpha}{\beta}) \ge t_k^*(\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta})$ due to the fact that $\prod_{\tau_j\in\mathbf{T}_2}(\beta U_j+1) \le \prod_{j=1}^{k-1}(\beta U_j+1)$.

Therefore, for a given $s \ge 0$, i.e., $t_k^* \ge t_k$, all the other extreme points of Eq. (9) were dominated by the one specified in Step 2. By the above analysis, if $\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta} > 0$, we know that $(t_k+s)(\frac{\frac{\alpha}{\beta}+1}{\prod_{j=1}^{k-1}(\beta U_j+1)} - \frac{\alpha}{\beta})$ is an increasing function of s, in which the minimum happens when s is 0. As a result, we reach the conclusion of the lemma. ∎

We now provide two extended lemmas based on Lemma 1, used for given α and β when $\alpha_i \le \alpha$ and $\beta_i \le \beta$ for $i = 1, 2, \ldots, k-1$. Their proofs are in Appendix B.

Lemma 2. *For a given k-point effective schedulability test of a scheduling algorithm, defined in Definition 1, in which $0 < t_k$ and $0 < \alpha_i \le \alpha$ and $0 < \beta_i \le \beta$ for any $i = 1, 2, \ldots, k-1$, task τ_k is schedulable by the scheduling algorithm if*

$$\frac{C_k}{t_k} + \sum_{i=1}^{k-1} U_i \le \begin{cases} 1, & (\alpha+\beta)^{\frac{1}{k}} < 1 \\ (k-1)\frac{\left((1+\frac{\beta}{\alpha})^{\frac{1}{k-1}} - 1\right)}{\beta}, & (\alpha+\beta)^{\frac{1}{k}} < \alpha \\ \frac{(k-1)((\alpha+\beta)^{\frac{1}{k}}-1)+((\alpha+\beta)^{\frac{1}{k}}-\alpha)}{\beta} & \textit{otherwise}. \end{cases} \tag{16}$$

Lemma 3. *For a given k-point effective schedulability test of a scheduling algorithm, defined in Definition 1, in which $0 < t_k$ and $0 < \alpha_i \le \alpha$ and $0 < \beta_i \le \beta$ for any $i = 1, 2, \ldots, k-1$, task τ_k is schedulable by the scheduling algorithm if*

$$\beta\sum_{i=1}^{k-1} U_i \le \ln(\frac{\frac{\alpha}{\beta}+1}{\frac{C_k}{t_k}+\frac{\alpha}{\beta}}). \tag{17}$$

We also construct the following Lemma 4, c.f. Appendix B for the proof, as the most powerful method for a concrete task system. Throughout the paper, we will not build theorems

and corollaries based on Lemma 4, the performance evaluation can be found in the report [12].

Lemma 4. *For a given k-point effective schedulability test of a fixed-priority scheduling algorithm, defined in Definition 1, task τ_k is schedulable by the scheduling algorithm, in which $0 < t_k$ and $0 < \alpha_i$ and $0 < \beta_i$ for any $i = 1, 2, \ldots, k-1$, if the following condition holds*

$$0 < \frac{C_k}{t_k} \leq 1 - \sum_{i=1}^{k-1} \frac{U_i(\alpha_i + \beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j + 1)}. \quad (18)$$

Remarks and how to use the framework: After presenting the $\mathbf{k^2U}$ framework, here, we explain how to use the $\mathbf{k^2U}$ framework and summarize how we plan to demonstrate its applicability in several task models in the following sections. The $\mathbf{k^2U}$ framework relies on the users to index the tasks properly and define α_i and β_i as constants for $i = 1, 2, \ldots, k-1$ based on Eq. (4). The set $\{t_1, t_2, \ldots, t_{k-1}\}$ in Definition 1 is used only for the users to define those constants, where t_k is usually defined to be the interval length of the original schedulability test, e.g., D_k in Eq. (1). Therefore, the $\mathbf{k^2U}$ framework can only be applicable when α_i and β_i are well-defined. These constants depend on the task models and the task parameters.

The choice of good parameters α_i and β_i affects the quality of the resulting schedulability bounds in Lemmas 1 to 3. However, deriving the *good* settings of α_i and β_i is actually not the focus of this paper. The framework does not care how the parameters α_i and β_i are obtained. The framework simply derives the bounds according to the given parameters α_i and β_i, regardless of the settings of α_i and β_i. The correctness of the settings of α_i and β_i is not verified by the framework.

The ignorance of the settings of α_i and β_i actually leads to the elegance and the generality of the framework, which works as long as Eq. (4) can be successfully constructed for the sufficient schedulability test of task τ_k in a fixed-priority scheduling policy. With the availability of the $\mathbf{k^2U}$ framework, the hyperbolic bounds or utilization bounds can be automatically derived by adopting Lemmas 1 to 3 as long as the safe upper bounds α and β to cover all the possible settings of α_i and β_i for the schedulability test in Eq. (4) can be derived, regardless of the task model or the platforms.

The other approaches in [8], [16], [18] also have similar observations by testing only several time points in the TDA schedulability analysis based on Eq. (1) in their problem formulations. Specifically, the problem formulations in [8], [16] are based on non-linear programming by selecting several good points under certain constraints. Moreover, the linear-programming problem formulation in [18] considers U_i as variables and t_i as constants and solves the corresponding linear programming analytically. However, as these approaches in [8], [16], [18] seek for the total utilization bounds, they have limited applications and are less flexible. For example, they are typically not applicable directly when considering sporadic real-time tasks with arbitrary deadlines or multiprocessor systems. Here, we are more flexible in the $\mathbf{k^2U}$ framework. For task τ_i, after α_i and β_i or their safe upper bounds α and β are derived, we completely drop out the dependency to the periods and models inside $\mathbf{k^2U}$.

We will demonstrate the applicability and generality of

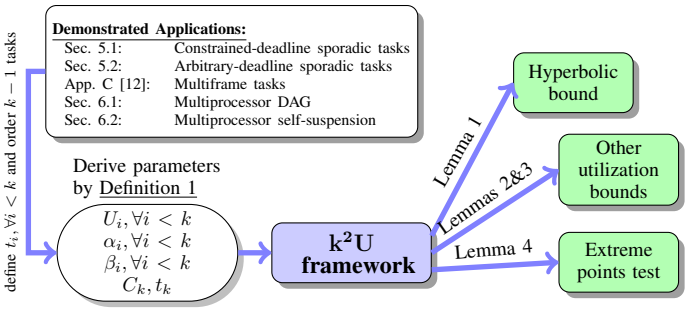

Fig. 1: The $\mathbf{k^2U}$ framework.

$\mathbf{k^2U}$ by using the most-adopted sporadic real-time task model in Sec. 5, multi-frame tasks in Appendix C in the report [12] and multiprocessor scheduling in Sec. 6, as illustrated in Figure 1. In all these cases, we can find reasonable settings of α and β to provide better results or new results for schedulability tests, with respect to the utilization bounds, speed-up factors, or capacity augmentation factors, compared to the literature. More specifically, (after certain reorganizations), we will greedily set t_i as $\left\lfloor \frac{t_k}{T_i} \right\rfloor T_i$ in all of the studied cases.[2] Table I summarizes the α_i and β_i parameters derived in this paper, as well as an earlier result by Liu and Chen in [24] for self-suspending task models and deferrable servers.

5 Applications for Fixed-Priority Scheduling

This section provides demonstrations on how to use the $\mathbf{k^2U}$ framework to derive efficient schedulability analysis for sporadic task systems in uniprocessor systems. We will consider constrained-deadline systems first in Sec. 5.1 and explain how to extend to arbitrary-deadline systems in Sec. 5.2. For the rest of this section, we will implicitly assume $C_k > 0$.

5.1 Constrained-Deadline Systems

For a specified fixed-priority scheduling algorithm, let $hp(\tau_k)$ be the set of tasks with higher priority than τ_k. We now classify the task set $hp(\tau_k)$ into two subsets:

- $hp_1(\tau_k)$ consists of the higher-priority tasks with periods smaller than D_k.
- $hp_2(\tau_k)$ consists of the higher-priority tasks with periods larger than or equal to D_k.

For any $0 < t \leq D_k$, we know that a safe upper bound on the interference due to higher-priority tasks is given by

$$\sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i = \sum_{\tau_i \in hp_2(\tau_k)} C_i + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i.$$

As a result, the schedulability test in Eq. (1) is equivalent to the verification of the existence of $0 < t \leq D_k$ such that

$$C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t. \quad (19)$$

[2] Setting t_i as $\left\lfloor \frac{t_k}{T_i} \right\rfloor T_i$ is actually the same in [18] for the sporadic real-time task model with implicit deadlines and the multi-frame task model when T_i is given and U_i is considered as a variable.

Model	α_i	β_i	c.f.
Uniprocessor Sporadic Tasks	$\alpha_i = 1$	$0 < \beta_i \leq 1$	Theorem 1 and Corollary 2
Multiprocessor Global RM for Different Models	$0 < \alpha_i \leq \frac{2}{M}$	$0 < \beta_i \leq \frac{1}{M}$	Theorems 4 - 5
Uniprocessor Multi-frame Tasks	$0 < \alpha_i \leq 1$	$0 < \beta_i \leq \frac{\phi_i(2)-\phi_i(1)}{\phi_i(1)}$	Theorem 7 in [12]
Uniprocessor Self-Suspending Tasks	$0 < \alpha_i \leq 2$	$0 < \beta_i \leq 1$	Theorems 5 and 6 in [24], implicitly

TABLE I: The applicability of $\mathbf{k^2U}$ for different task models: their α_i and β_i parameters, where M is the number of processors in global RM scheduling and $\phi_i(\ell)$ is the maximum of the sum of the execution time of any ℓ consecutive frames of task τ_i in the multi-frame model.

We can then create a virtual sporadic task τ'_k with execution time $C'_k = C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i$, relative deadline $D'_k = D_k$, and period $T'_k = D_k$. It is clear that the schedulability test to verify the schedulability of task τ'_k under the interference of the higher-priority tasks $hp_1(\tau_k)$ is the same as that of task τ_k under the interference of the higher-priority tasks $hp(\tau_k)$.

Therefore, with the above analysis, we can use the $\mathbf{k^2U}$ framework in Sec. 4 as in the following theorem.

Theorem 1. *Task τ_k in a sporadic task system with constrained deadlines is schedulable by the fixed-priority scheduling algorithm if*

$$(\frac{C'_k}{D_k} + 1) \prod_{\tau_j \in hp_1(\tau_k)} (U_j + 1) \leq 2 \tag{20}$$

or

$$\frac{C'_k}{D_k} + \sum_{\tau_j \in hp_1(\tau_k)} U_j \leq k(2^{\frac{1}{k}} - 1). \tag{21}$$

Proof: For notational brevity, suppose that there are $k-1$ tasks in $hp_1(\tau_k)$.[3] Now, we index the higher-priority tasks in $hp_1(\tau_k)$ to form the corresponding $\tau_1, \tau_2, \ldots, \tau_{k-1}$. The $k-1$ higher-priority tasks in $hp_1(\tau_k)$ are ordered to ensure that the arrival times, i.e., $\left\lfloor \frac{D_k}{T_i} \right\rfloor T_i$, of the last jobs no later than D_k are in a non-decreasing order. That is, with the above indexing of the higher-priority tasks in $hp_1(\tau_k)$, we have $\left\lfloor \frac{D_k}{T_i} \right\rfloor T_i \leq \left\lfloor \frac{D_k}{T_{i+1}} \right\rfloor T_{i+1}$ for $i = 1, 2, \ldots, k-2$. Now, we set t_i as $\left\lfloor \frac{D_k}{T_i} \right\rfloor T_i$ for $i = 1, 2, \ldots, k-1$, and t_k as D_k. Due to the fact that $T_i \leq D_k$ for $i = 1, 2, \ldots, k-1$, we know that $t_i > 0$.

Therefore, for a given t_j with $j = 1, 2, \ldots, k$, the demand requested up to time t_j is at most

$$\begin{aligned} & C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t_j}{T_i} \right\rceil C_i \\ =& C'_k + \sum_{i=1}^{k-1} \left\lceil \frac{t_j}{T_i} \right\rceil C_i \leq C'_k + \sum_{i=1}^{k-1} \frac{t_i}{T_i} C_i + \sum_{i=1}^{j-1} C_i, \end{aligned}$$

where the inequality comes from the indexing policy defined above, i.e., $\left\lceil \frac{t_j}{T_i} \right\rceil \leq \frac{t_i}{T_i} + 1$ if $j > i$ and $\left\lceil \frac{t_j}{T_i} \right\rceil \leq \frac{t_i}{T_i}$ if $j \leq i$. Since $T_i < D_k$ for any task τ_i in $hp_1(\tau_k)$, we know that $t_i \geq T_i$. Instead of testing all the t values in Eq. (19), we only apply the test for these k different t_i values, which is equivalent to the test of the existence of t_j such that

$$C'_k + \sum_{i=1}^{k-1} t_i U_i + \sum_{i=1}^{j-1} \frac{T_i}{t_i} t_i U_i \leq t_j. \tag{22}$$

Therefore, we reach the schedulability in the form of Eq. (4) under the setting of α_i to 1 and β_i to $\frac{T_i}{t_i} \leq 1$ (due to $t_i \geq T_i$), for $i = 1, 2, \ldots, k-1$. By taking $\beta_i \leq 1$ and $\alpha_i = 1$ for $i = 1, 2, \ldots, k-1$ in Lemmas 1 and 2, we reach the conclusion. ■

When RM priority ordering is applied for an implicit-deadline task system, C'_k is equal to C_k and $\frac{C'_k}{D_k}$ is equal to U_k. For such a case, the condition in Eq. (20) is the same as the hyperbolic bound provided in [6], and the condition in Eq. (21) is the same as least utilization upper bound in [25].

The schedulability test in Theorem 1 may seem pessimistic at first glance. We evaluate the quality of the schedulability test in Theorem 1 by quantifying the speed-up factor with respect to the optimal schedule (i.e., EDF scheduling in such a case). We show that the speed-up factor of the schedulability test in Eq. (20) is 1.76322, which is the same as the lower bound and upper bound of DM as shown in [14]. The speed-up factor of DM, regardless of the schedulability tests, obtained by Davis et al. [14] is the same as our result. Our result is thus stronger, as we show that the factor already holds when using the schedulability test in Eq. (20) in the following theorem.

Theorem 2. *The speed-up factor of the schedulability test in Eq.* (20) *under DM scheduling for constrained-deadline tasks is* 1.76322 *with respect to EDF.*

The proof of Theorem 2 in the Appendix B, which is much simpler than the proof in [14], can also be considered as an alternative proof of the speed-up factor of DM.

Corollary 1. *Suppose that $f \cdot T_i \leq D_k$ for any higher priority task τ_i in $hp_1(\tau_k)$, where f is a positive integer. Task τ_k in a constrained-deadline sporadic task system is schedulable by the fixed-priority scheduling algorithm if*

$$(\frac{C'_k}{f \cdot D_k} + 1) \prod_{\tau_j \in hp_1(\tau_k)} (\frac{U_j}{f} + 1) \leq \frac{f+1}{f} \tag{23}$$

or

$$\frac{C'_k}{D_k} + \sum_{\tau_j \in hp_1(\tau_k)} U_j \leq fk \left(\left(\frac{f+1}{f} \right)^{\frac{1}{k}} - 1 \right). \tag{24}$$

[3] When $hp_2(\tau_k)$ is not empty, there are $k-1-|hp_2(\tau_k)|$ tasks in $hp_1(\tau_k)$. To be notationally precise, we can denote the number of tasks in $hp_1(\tau_k)$ by a new symbol k^*. Since $k^* \leq k$, we know $k^*(2^{\frac{1}{k^*}} - 1) \geq k(2^{\frac{1}{k}} - 1)$ as a safe bound in Eq. (21). However, this may make the notations too complicated. We have decided to keep it as $k-1$ for the sake of notational brevity.

Proof: This is based on the same proof as Theorem 1, by taking the fact that $\frac{t_i}{T_i} = \frac{\left\lfloor \frac{D_k}{T_i} \right\rfloor T_i}{T_i} \geq f$. In the k-point effective schedulability test, we can set α_i to 1, $\beta_i = \frac{T_i}{t_i} \leq \frac{1}{f}$. Therefore, we have $\alpha_i \leq \alpha = 1$, $\beta_i \leq \beta = \frac{1}{f}$, and $\frac{\alpha}{\beta}$ is f. By adopting Lemma 1, we know that task τ_k is schedulable under the scheduling policy if $(\frac{C'_k}{D_k}+f)\prod_{\tau_j \in hp_1(\tau_k)}(\frac{U_j}{f}+1) \leq f+1$, which is the same as the condition in Eq. (23) by dividing both sides by f. By using a similar argument and applying Lemma 2, we can reach the condition in Eq. (24). ■

5.2 Arbitrary-Deadline Systems

We now further explore how to use the proposed framework to perform the schedulability analysis for arbitrary-deadline task sets. The exact schedulability analysis for arbitrary-deadline task sets under fixed-priority scheduling has been developed in [19]. The schedulability analysis is to use a *busy-window* concept to evaluate the worst-case response time. That is, we release all the higher-priority tasks together with task τ_k at time 0 and all the subsequent jobs are released as early as possible by respecting to the minimum inter-arrival time. The busy window finishes when a job of task τ_k finishes before the next release of a job of task τ_k. It has been shown in [19] that the worst-case response time of task τ_k can be found in one of the jobs of task τ_k in the busy window.

Therefore, a simpler sufficient schedulability test for a task τ_k is to test whether the length of the busy window is within D_k. If so, all invocations of task τ_k released in the busy window can finish before their relative deadline. Such an observation has also been adopted in [13]. Therefore, a sufficient test is to verify whether[4]

$$\exists t \text{ with } 0 < t \leq D_k \text{ and } \left\lceil \frac{t}{T_k} \right\rceil C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t. \tag{25}$$

If the condition in Eq. (25) holds, it implies that the busy window (when considering task τ_k) is no more than D_k, and, hence, task τ_k has worst-case response time no more than D_k.

If $D_k \leq T_k$, the analysis in Sec. 5.1 can be directly applied. If $D_k > T_k$, we need to consider the length of the busy-window for task τ_k as shown above. For the rest of this section, we will focus on the case $D_k > T_k$. We can rewrite Eq. (25) to use a more pessimistic condition by releasing the workload $\left\lceil \frac{D_k}{T_k} \right\rceil C_k$ at time 0. That is, if

$$\exists t \text{ with } 0 < t \leq D_k \text{ and } \left\lceil \frac{D_k}{T_k} \right\rceil C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t, \tag{26}$$

then, the length of the busy window for task τ_k is no more than D_k. Again, similar to the strategy we use in Sec. 5.1, we classify the tasks in $\tau_i \in hp(\tau_k)$ into two sets $hp_1(\tau_k)$ and $hp_2(\tau_k)$ with the same definition.

Similarly, we can then create a virtual sporadic task τ'_k with execution time $C'_k = \left\lceil \frac{D_k}{T_k} \right\rceil C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i$, relative deadline $D'_k = D_k$, and period $T'_k = D_k$. For notational brevity, suppose that there are $k-1$ tasks in $hp_1(\tau_k)$. Now, we index the higher-priority tasks in $hp_1(\tau_k)$ to form the corresponding $\tau_1, \tau_2, \ldots, \tau_{k-1}$. In the above definition of the busy window concept, $\left\lfloor \frac{D_k}{T_i} \right\rfloor T_i$ is the arrival time of the last job of task τ_i released no later than D_k. The $k-1$ higher-priority tasks in $hp_1(\tau_k)$ are ordered to ensure that the arrival times of the last jobs before D_k are in a non-decreasing order. Moreover, t_k is the specified testing point D_k. Instead of testing all the t values in Eq. (26), we only apply the test for these k different t_i values, which is equivalent to the test of the existence of t_j such that Eq. (22) holds, where $\alpha_i \leq 1$ and $\beta_i = \frac{T_i}{t_i} \leq 1$ for $i = 1, 2, \ldots, k-1$, similar to the proof of Theorem 1. The following corollary comes from a similar argument as in Sec. 5.1.

Corollary 2. *Task τ_k in a sporadic arbitrary-deadline task system is schedulable by the fixed-priority scheduling algorithm if Eq.* (20) *or* (21) *holds, in which there are $k-1$ higher priority tasks in $hp_1(\tau_k)$ and C'_k is defined as $\left\lceil \frac{D_k}{T_k} \right\rceil C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i$.*

Corollary 3. *Suppose that $f \cdot T_i \leq D_k$ for any higher task τ_i in the task system and $f \cdot T_k \leq D_k$, where f is a positive integer. Task τ_k in a sporadic task system is schedulable by using RM, i.e., $T_i \leq T_{i+1}$, if*

$$(\frac{U_k}{f}+1)\prod_{j=1}^{k-1}(\frac{U_j}{f}+1) \leq \frac{f+1}{f} \tag{27}$$

or

$$\sum_{j=1}^{k} U_j \leq fk\left(\left(\frac{f+1}{f}\right)^{\frac{1}{k}} - 1\right). \tag{28}$$

Proof: Based on the above assumption $f \cdot T_i \leq D_k$ for any higher task τ_i in the task system, $f \cdot T_k \leq D_k$, and the rate monotonic scheduling, we can safely set α_i to 1, β_i to $\frac{1}{f}$, and C'_k to $f \cdot C_k$. Note that $\frac{C'_k}{f \cdot t_k} \leq \frac{U_k}{f}$ in this case. Therefore, by adopting Lemma 1 and Lemma 2, we reach the conclusion, as in the proof of Corollary 1. ■

6 Application for Multiprocessor Scheduling

It may seem, at first glance, that the $\mathbf{k^2U}$ framework only works for uniprocessor systems. We demonstrate in this section how to use the framework in multiprocessor global RM scheduling when considering implicit-deadline, DAG, and self-suspending task systems. The methodology can also be extended to handle constrained-deadline systems.

In multiprocessor global scheduling, we consider that the system has M identical processors, each with the same computation power. Moreover, there is a global queue and a global scheduler to dispatch the jobs. We consider only global RM scheduling, in which the priority of the tasks is defined based on RM. At any time, the M-highest-priority jobs in the ready queue are dispatched and executed on these M processors.

Global RM in general does not have good utilization bounds. However, if we constrain the total utilization $\sum_{\tau_i} \frac{C_i}{MT_i} \leq \frac{1}{b}$ and the maximum utilization $\max_{\tau_i} \frac{C_i}{T_i} \leq \frac{1}{b}$, it is possible to provide the schedulability guarantee of global

[4]This analysis is pretty pessimistic. But, as our objective in this paper is to show the applicability and generality of $\mathbf{k^2U}$, how to use $\mathbf{k^2U}$ to get tighter analysis for arbitrary-deadline task systems is not our focus in this paper. Some evaluations can be found in [10].

RM by setting b to $3 - \frac{1}{M}$ [1], [2], [5]. Such a factor b has been recently named as a *capacity augmentation factor* [22].

We will use the following time-demand function $W_i(t)$ for the simple sufficient schedulability analysis:

$$W_i(t) = (\left\lceil \frac{t}{T_i} \right\rceil - 1)C_i + 2C_i, \tag{29}$$

which can be imagined as if the *carry-in* job is fully carried into the analysis interval [15]. That is, we allow the first release of task τ_i to be inflated by a factor 2, whereas the other jobs of task τ_i have the same execution time C_i. Again, we consider testing the schedulability of task τ_k under global RM, in which there are $k-1$ higher-priority tasks $\tau_1, \tau_2, \ldots, \tau_{k-1}$. We have the following schedulability condition for global RM.

Lemma 5. *Task τ_k is schedulable under global RM on M identical processors, if*

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + \sum_{i=1}^{k-1} \frac{W_i(t)}{M} \leq t, \tag{30}$$

where $W_i(t)$ is defined in Eq. (29)*.*

Proof: This has been shown in Sec. 3.2 (The Basic Multiprocessor Case) in [15]. ∎

Theorem 3. *Task τ_k in a sporadic implicit-deadline task system is schedulable by global RM on M processors if*

$$(\frac{C_k}{T_k} + 2) \prod_{j=1}^{k-1} (\frac{U_j}{M} + 1) \leq 3, \tag{31}$$

or

$$\sum_{j=1}^{k-1} \frac{U_j}{M} \leq \ln \frac{3}{\frac{C_k}{T_k} + 2}. \tag{32}$$

Proof: Let t_i be $\left\lfloor \frac{T_k}{T_i} \right\rfloor T_i$ for $i = 1, 2, \ldots, k$, and reindex the tasks such that $t_1 \leq t_2 \leq \ldots \leq t_k$. By testing only these k points in the schedulability test in (30) results in a k-point effective schedulability test with $\alpha_i \leq \frac{2}{M}$ and $\beta_i \leq \frac{1}{M}$. Therefore, we can adopt the $\mathbf{k^2U}$ framework. By Lemma 1 and Lemma 3, we have concluded the proof. ∎

Note that Theorem 3 is not superior to the known analysis for sporadic task systems [1], [2], [5], as the schedulability condition in Lemma 5 is too pessimistic. This is only used as the basis to analyze a sporadic task set with DAG tasks in Sec. 6.1 and self-suspending tasks in Sec. 6.2 when adopting global RM, in which we will demonstrate similar structures as used in Theorem 3. We also demonstrate a tighter test in Appendix F in the report [12] for improving the schedulability test of global RM for sporadic tasks.

6.1 Global RM for DAG Task Systems

For multiprocessor scheduling, the DAG task model has been recently studied [7]. The utilization-based analysis can be found in [22] and [7]. Each task $\tau_i \in \mathbf{T}$ in a DAG task system is a parallel task. Each task is characterized by its execution pattern, defined by a set of directed acyclic graphs (DAGs). The execution time of a job of task τ_i is one of the DAGs. Each node (subtask) in a DAG represents a sequence of instructions (a thread) and each edge represents a dependency between nodes. A node (subtask) is *ready* to be executed when all its predecessors have been executed. We will only consider two parameters related to the execution pattern of task τ_i:

- *total execution time (or work)* C_i of task τ_i: This is the summation of the execution times of all the subtasks of task τ_i among all the DAGs of task τ_i.
- *critical-path length* Ψ_i of task τ_i: This is the length of the critical path among the given DAGs, in which each node is characterized by the execution time of the corresponding subtask of task τ_i.

The analysis is based on the two given parameters C_i and Ψ_i. Therefore, we can also allow flexible DAG structures. That is, jobs of a task may have different DAG structures, under the total execution time constraint C_i and the critical path length constraint Ψ_i. Therefore, the model can also be applied for conditional sporadic DAG task systems [3]. With the above definition, we have the following lemma, in which the proof is Appendix B in the report [12].

Lemma 6. *Task τ_k in a sporadic DAG system with implicit deadlines is schedulable under global RM on M identical processors, if*

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } \Psi_k + \frac{C_k - \Psi_k}{M} + \sum_{i=1}^{k-1} \frac{W_i(t)}{M} \leq t, \tag{33}$$

where $W_i(t)$ is defined in Eq. (29)*.*

Theorem 4. *Task τ_k in a sporadic DAG system with implicit deadlines is schedulable by global RM on M processors if*

$$(\frac{\Psi_k}{T_k} + 2) \prod_{j=1}^{k} (\frac{U_j}{M} + 1) \leq 3 \tag{34}$$

or

$$\sum_{j=1}^{k} \frac{U_j}{M} \leq \ln \frac{3}{\frac{\Psi_k}{T_k} + 2}. \tag{35}$$

Proof: Based on Lemma 6, which is very similar to Lemma 5, we can perform a similar transformation as in Theorem 3, in which $\alpha_i \leq \frac{2}{M}$ and $\beta_i \leq \frac{1}{M}$. By adopting Lemma 1, we know that if

$$(\frac{\Psi_k + \frac{C_k - \Psi_k}{M}}{T_k} + 2) \prod_{j=1}^{k-1} (\frac{U_j}{M} + 1) \leq 3, \tag{36}$$

then task τ_k is schedulable. Due to the fact that $C_k - \Psi_k \leq C_k$, we know that $(\frac{\Psi_k + \frac{C_k - \Psi_k}{M}}{T_k} + 2) \leq (\frac{\Psi_k + \frac{C_k}{M}}{T_k} + 2) \leq (\frac{\Psi_k}{T_k} + 2) \cdot (\frac{U_k}{M} + 1)$. Therefore, if the condition in Eq. (34) holds, the condition in Eq. (36) also holds, which implies the schedulability. With the result in Eq. (34), we can use the same procedure as in Lemma 3 to obtain Eq. (35). ∎

Corollary 4. *The capacity augmentation factor of global RM for a sporadic DAG system with implicit deadlines is* 3.62143*.*

Proof: Suppose that $\sum_{\tau_i} \frac{C_i}{MT_i} \leq \frac{1}{b}$ and $\frac{\Psi_k}{T_k} \leq \max_{\tau_i} \frac{\Psi_i}{T_i} \leq \frac{1}{b}$. Therefore, by Eq. (35), we can guarantee the schedulability of task τ_k if $\frac{1}{b} \leq \ln \frac{3}{2+\frac{1}{b}}$. This is equivalent to solving $x = \ln \frac{3}{2+x}$, which holds when $x \approx 3.62143$ by solving the equation numerically. Therefore, we reach the

conclusion of the capacity augmentation factor 3.62143. ■

6.2 Global RM for Self-Suspending Tasks

The self-suspending task model extends the sporadic task model by allowing tasks to suspend themselves. An overview of work on scheduling self-suspending task systems can be found in [24].

Similar to sporadic tasks, a self-suspending task releases jobs sporadically. Jobs alternate between computation and suspension phases. We assume that each job of τ_i executes for at most C_i time units (across all of its execution phases) and suspends for at most S_i time units (across all of its suspension phases). We assume that $C_i + S_i \leq T_i$ for any task $\tau_i \in \tau$; for otherwise deadlines would be missed. The self-suspending model is general: we place no restrictions on the number of phases per-job and how these phases interleave (a job can even begin or end with a suspension phase). Different jobs belong to the same task can also have different phase-interleaving patterns. For many applications, such a general self-suspending model is needed due to the unpredictable nature of I/O operations. We have the following lemma, in which the proof is in Appendix B in the report [12].

Lemma 7. *Task τ_k in a self-suspending system with implicit deadlines is schedulable under global RM on M identical processors, if*

$$\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + \sum_{i=1}^{k-1} \frac{W_i(t)}{M} \leq t, \quad (37)$$

where $W_i(t)$ is defined in Eq. (29)*.*

Theorem 5. *Task τ_k in a sporadic self-suspending system with implicit deadlines is schedulable by global RM on M processors if*

$$(\frac{C_k + S_k}{T_k} + 2) \prod_{j=1}^{k-1} (\frac{U_j}{M} + 1) \leq 3. \quad (38)$$

Proof: By Lemma 7, we can perform a similar transformation as in Theorem 3 with $\alpha_i \leq \frac{2}{M}$ and $\beta_i \leq \frac{1}{M}$. ■

7 Conclusion

With the presented applications, we believe that the general schedulability analysis $\mathbf{k^2U}$ framework for fixed-priority scheduling has high potential to be adopted for analyzing other task models in real-time systems. We constrain ourselves by demonstrating the applications for simple scheduling policies, like global RM in multiprocessor scheduling. The framework can be used, once the k-point effective scheduling test can be constructed. Although the emphasis of this paper is not to show that the resulting tests for different task models by applying the $\mathbf{k^2U}$ framework are better than existing work, some analysis results by applying the $\mathbf{k^2U}$ framework have been shown superior to the state of the art. For completeness, another document has been prepared in [10] to present the similarity, the difference and the characteristics of $\mathbf{k^2U}$ and $\mathbf{k^2Q}$. With our frameworks, some difficult schedulability test and response time analysis problems may be solved by building a good (or exact) exponential-time test and applying these frameworks.

Appendix D in the report [12] provides some case studies with evaluation results of some selected utilization-based schedulability tests. Appendix E in the report [12] further provides some additional properties that come directly from the $\mathbf{k^2U}$ framework. More applications can be found in partitioned scheduling [9], non-preemptive scheduling [30], etc.

Acknowledgement: This paper has been supported by DFG, as part of the Collaborative Research Center SFB876 (http://sfb876.tu-dortmund.de/), and the priority program "Dependable Embedded Systems" (SPP 1500 - http://spp1500.itec.kit.edu). We would also like to thank Dr. Vincenzo Bonifaci for his valuable input to improve the presentation of the paper.

References

[1] B. Andersson, S. K. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium (RTSS)*, pages 193–202, 2001.

[2] T. P. Baker. Multiprocessor EDF and deadline monotonic schedulability analysis. In *IEEE Real-Time Systems Symposium*, pages 120–129, 2003.

[3] S. Baruah, V. Bonifaci, and A. Marchetti-Spaccamela. The global EDF scheduling of systems of conditional sporadic DAG tasks. In *ECRTS*, pages 222–231, 2015.

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

[5] M. Bertogna, M. Cirinei, and G. Lipari. New schedulability tests for real-time task sets scheduled by deadline monotonic on multiprocessors. In *Principles of Distributed Systems, 9th International Conference, OPODIS*, pages 306–321, 2005.

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

[7] V. Bonifaci, A. Marchetti-Spaccamela, S. Stiller, and A. Wiese. Feasibility analysis in the sporadic dag task model. In *ECRTS*, pages 225–233, 2013.

[8] A. Burchard, J. Liebeherr, Y. Oh, and S. H. Son. New strategies for assigning real-time tasks to multiprocessor systems. pages 1429–1442, 1995.

[9] J.-J. Chen. Partitioned multiprocessor fixed-priority scheduling of sporadic real-time tasks. *Computing Research Repository (CoRR)*, abs/1505.04693, 2015.

[10] J.-J. Chen, W.-H. Huang, and C. Liu. Evaluate and compare two utilization-based schedulability-test frameworks for real-time systems. *Computing Research Repository (CoRR)*, abs/1505.02155, 2015.

[11] 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.

[12] J.-J. Chen, W.-H. Huang, and C. Liu. k^2U: A general framework from k-point effective schedulability analysis to utilization-based tests. *Computing Research Repository (CoRR)*, abs/1501.07084, 2015.

[13] R. 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 and Network Systems (RTNS)*, pages 23–31, 2009.

[14] R. I. Davis, T. Rothvoß, S. K. Baruah, and A. Burns. Exact quantification of the sub-optimality of uniprocessor fixed priority pre-emptive scheduling. *Real-Time Systems*, 43(3):211–258, 2009.

[15] N. Guan, M. Stigge, W. Yi, and G. Yu. New response time bounds for fixed priority multiprocessor scheduling. In *IEEE Real-Time Systems Symposium*, pages 387–397, 2009.

[16] C.-C. Han and H.-Y. Tyan. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. In *Real-Time Systems Symposium (RTSS)*, pages 36–45, 1997.

[17] T.-W. Kuo, L.-P. Chang, Y.-H. Liu, and K.-J. Lin. Efficient online schedulability tests for real-time systems. *Software Engineering, IEEE Transactions on*, 29(8):734–751, 2003.

[18] C.-G. Lee, L. Sha, and A. Peddi. Enhanced utilization bounds for QoS management. *IEEE Trans. Computers*, 53(2):187–200, 2004.

[19] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *RTSS*, pages 201–209, 1990.

[20] J. P. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. In *IEEE Real-Time Systems Symposium*, pages 166–171, 1989.

[21] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Perform. Eval.*, 2(4):237–250, 1982.

[22] J. Li, J. Chen, K. Agrawal, C. Lu, C. Gill, and A. Saifullah. Analysis of federated and global scheduling for parallel real-time tasks. In *Euromicro Conference on Real-Time Systems*, 2014.

[23] C. Liu and J. Anderson. Task scheduling with self-suspensions in soft real-time multiprocessor systems. In *Proceedings of the 30th Real-Time Systems Symposium*, pages 425–436, 2009.

[24] C. Liu and J.-J. Chen. Bursty-interference analysis techniques for analyzing complex real-time task models. In *IEEE Real-Time Systems Symposium*, 2014.

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

[26] W.-C. Lu, K.-J. Lin, H.-W. Wei, and W.-K. Shih. New schedulability conditions for real-time multiframe tasks. In *ECRTS*, pages 39–50. IEEE, 2007.
[27] D. G. Luenberger and Y. Ye. *Linear and nonlinear programming*, volume 116. Springer, 2008.
[28] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. 1983.
[29] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Trans. Software Eng.*, pages 635–645, 1997.
[30] G. von der Bruggen, J.-J. Chen, and W. Huang. Schedulability and optimization analysis for non-preemptive static priority scheduling based on task utilization and blocking factors. In *ECRTS*, pages 90–101, 2015.
[31] J. Wu, J. Liu, and W. Zhao. On schedulability bounds of static priority schedulers. In *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 529–540, 2005.

Appendix A: Monotonic Schedulability Test

The tests presented in the theorems or corollaries do not guarantee to have the monotonicity with respect to the k-th highest-priority task. However, by sacrificing the quality of the schedulability tests, we can still obtain monotonicity, with which the schedulability test of a task set can be done with linear-time complexity. These tests can be used for on-line admission control. For example, the test in Theorem 4 can be modified to the following theorem:

Theorem 6. *An implicit-deadline DAG system τ is schedulable by global RM on M processors if*

$$(\Delta_{\max}+2)\prod_{\tau_i}(\frac{U_i}{M}+1)\leq 3, \tag{39}$$

where $\Delta_{\max}$ is $max_{\tau_i\in\tau}\frac{\Psi_i}{T_i}$.

Appendix B: Proofs

Proof of Lemma 2. This lemma is proved by sketch with Lagrange Multiplier to find the infimum $\frac{C_k}{t_k}+\sum_{i=1}^{k-1}U_i$ such that Eq. (5) does not hold, which is a non-linear programming problem. Due to the fact that $(1+\beta U_1)(1+\beta U_2)\leq(1+\beta\frac{U_1+U_2}{2})^2$ when $\beta\geq 0, U_1\geq 0, U_2\geq 0$, the infimum $\frac{C_k}{t_k}+\sum_{i=1}^{k-1}U_i$ happens when $U_1=U_2=\cdots=U_{k-1}$. So, there are only two variables $\frac{C_k}{t_k}$ and U_1 to minimize $\frac{C_k}{t_k}+(k-1)U_1$ such that $(\frac{C_k}{t_k}+\frac{\alpha}{\beta})(\beta U_1+1)^{k-1}\geq\frac{\alpha}{\beta}+1$.

Let λ be the Lagrange Multiplier and F be $\frac{C_k}{t_k}+(k-1)U_1-\lambda\left((\frac{C_k}{t_k}+\frac{\alpha}{\beta})(\beta U_1+1)^{k-1}-(\frac{\alpha}{\beta}+1)\right)$. The minimum $\frac{C_k}{t_k}+(k-1)U_1$ happens when $\frac{\partial F}{\partial U_1}=(k-1)-\lambda\beta(k-1)(\frac{C_k}{t_k}+\frac{\alpha}{\beta})(\beta U_1+1)^{k-2}=0$ and $\frac{\partial F}{\partial\frac{C_k}{t_k}}=1-\lambda(\beta U_1+1)^{k-1}=0$. When $k\geq 2$, by reorganizing the above two equations, we have $1-\frac{\beta(\frac{\alpha}{\beta}+\frac{C_k}{t_k})}{\beta U_1+1}=0$. By the Lagrange Multiplier method, the minimum happens when $(\frac{C_k}{t_k}+\frac{\alpha}{\beta})(\beta U_1+1)^{k-1}=\frac{\alpha}{\beta}+1$ and $1-\frac{\beta(\frac{\alpha}{\beta}+\frac{C_k}{t_k})}{\beta U_1+1}=0$. By solving the above equation, the non-linear programming is minimized when U_1 is $\frac{(\alpha+\beta)^{\frac{1}{k}}-1}{\beta}$ and $\frac{C_k}{t_k}$ is $\frac{(\alpha+\beta)^{\frac{1}{k}}-\alpha}{\beta}$.

We also need to consider the boundary cases when $(\alpha+\beta)^{\frac{1}{k}}-1<0$ or $(\alpha+\beta)^{\frac{1}{k}}-\alpha<0$ with Karush Kuhn Tucker (KKT) conditions. The Lagrange Multiplier method may result in a solution with a negative U_1 when $(\alpha+\beta)^{\frac{1}{k}}-1<0$. If this happens, we know that the extreme case happens when U_1 is 0 by using KKT condition. Moreover, if $(\alpha+\beta)^{\frac{1}{k}}-\alpha<0$, then we know that $\frac{C_k}{t_k}$ should be set to 0 in the extreme case by using KKT condition. By the above analysis, we reach the conclusion in Eq. (16). □

Proof of Lemma 3. This comes directly from Eq. (5) in Lemma 1 with a simpler Lagrange Multiplier procedure as in the proof of Lemma 2, in which the infimum total utilization under $\prod_{j=1}^{k-1}(\beta U_j+1)>\frac{\frac{\alpha}{\beta}+1}{\frac{C_k}{t_k}+\frac{\alpha}{\beta}}$ happens when all the $k-1$ tasks have the same utilization. □

Proof of Lemma 4. The first part of the proof by constructing the corresponding linear programming to minimize C_k^* as follows is the same as in the proof of Lemma 1 by setting t_k^* as t_k+s with $s\geq 0$:

$$\min\ t_k^*-\sum_{i=1}^{k-1}(\alpha_i+\beta_i)U_i t_i^* \tag{40a}$$

$$\text{s.t. } t_k^*-\sum_{i=j}^{k-1}\beta_i t_i^* U_i\geq t_j^* \qquad \forall 1\leq j\leq k-1, \tag{40b}$$

$$t_j^*\geq 0 \qquad \forall 1\leq j\leq k-1. \tag{40c}$$

$$t_k^*\geq t_k. \tag{40d}$$

Similarly, a feasible extreme point solution can be represented by two sets $\mathbf{T}_1$ and $\mathbf{T}_2$ of the $k-1$ higher-priority tasks, in which $t_j^*=0$ if τ_j is in $\mathbf{T}_1$ and $t_j^*>0$ if task τ_j is in $\mathbf{T}_2$.

One specific extreme point solution is to have $\mathbf{T}_1=\emptyset$. For such a case, we use the same steps from Eq. (10) to Eq. (12):

$$\frac{t_i^*}{t_k^*}=\prod_{j=i}^{k-1}\frac{t_j^*}{t_{j+1}^*}=\frac{1}{\prod_{j=i}^{k-1}(\beta_j U_j+1)}. \tag{41}$$

The resulting objective function of this extreme point solution for Eq. (40) is $t_k^*(1-\sum_{i=1}^{k-1}\frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j+1)})$. The above steps are identical to Step 1 and Step 2 in the proof of Lemma 1. We will show, similarly to Step 3 in the proof of Lemma 1, for the rest of the proof, that the above extreme point solution is either optimal for the objective function of Eq. (40) or $1-\sum_{i=1}^{k-1}\frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j+1)}\leq 0$.

For a feasible extreme point solution with $\mathbf{T}_1\neq\emptyset$, we will convert it to the above extreme point solution with $\mathbf{T}_1=\emptyset$ by steps, in which each step moves one task from $\mathbf{T}_1$ to $\mathbf{T}_2$ by decreasing the objective function in the linear programming in Eq. (40). For the rest of the proof, we start from a feasible extreme point solution, specified by $S=<\mathbf{T}_1,\mathbf{T}_2>$. Suppose that τ_ℓ is the *first* task in this extreme point solution S with t_ℓ^* set to 0, i.e., $t_k^*-\sum_{i=j}^{k-1}\beta_i t_i^* U_i=t_j^*$ for $j=1,2,\ldots,\ell-1$.

Assume that $\omega>\ell$ is the index of the next task with $t_\omega^*>0$ in the extreme point solution S, i.e., $t_\ell^*=t_{\ell+1}^*=\cdots=t_{\omega-1}^*=0$. If all the remaining tasks are with $t_i^*=0$ for $\ell\leq i\leq k-1$, then ω is set to k and t_ω^* is t_k^*. If ℓ is 1, we can easily set t_1^* to $\frac{t_\omega^*}{1+\beta_1 U_1}$, which is >0 and the objective function of the linear programming becomes smaller. We focus on the cases where $\ell>1$. We can use the same steps from Eq. (10) to Eq. (12): $t_{i+1}^*-t_i^*=\beta_i U_i$ for $i=1,2,\ldots,\ell-2$ and $t_\omega^*-t_{\ell-1}^*=\beta_{\ell-1}U_{\ell-1}$. Therefore, $\sum_{i=1}^{\ell-1}(\alpha_i+\beta_i)U_i t_i^*=t_\omega^*(\sum_{i=1}^{\ell-1}\frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)})$. There are two cases:

Case 1: If $\sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)} \geq 1$, then we can conclude

$$\begin{aligned}
&\sum_{i=1}^{k-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j+1)} \\
=& \sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{(\prod_{j=i}^{\ell-1}(\beta_j U_j+1))(\prod_{j=\ell}^{k-1}(\beta_j U_j+1))} + (\sum_{i=\ell}^{k-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j+1)}) \\
\overset{1}{\geq}& \frac{1}{\prod_{j=\ell}^{k-1}(\beta_j U_j+1)} + (\sum_{i=\ell}^{k-1} \frac{U_i\beta_i}{\prod_{j=i}^{k-1}(\beta_j U_j+1)}) \\
=& \frac{1}{\prod_{j=\ell}^{k-1}(\beta_j U_j+1)} + (1 - \frac{1}{\prod_{j=\ell}^{k-1}(\beta_j U_j+1)}) \\
=& 1,
\end{aligned}$$

where $\overset{1}{\geq}$ comes from the assumption $\sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)} \geq 1$ and $\alpha_\ell > 0$.

Case 2: If $\sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)} < 1$, then we can greedily set $t_\ell^* > 0$ (i.e., move task τ_ℓ from $\mathbf{T}_1$ to $\mathbf{T}_2$). Such a change of τ_ℓ from $\mathbf{T}_1$ to $\mathbf{T}_2$ has no impact on task τ_i with $i > \ell$, but has impact on all the tasks τ_i with $i \leq \ell$. That is, after changing, by using the same steps from Eq. (10) to Eq. (12), we have $t_{i+1}^* - t_i^* = \beta_i U_i$ for $i = 1, 2, \ldots, \ell-1$ and $t_\omega^* - t_\ell^* = \beta_\ell U_\ell$. The change of the objective function in Eq. (40) after moving task τ_ℓ from $\mathbf{T}_1$ to $\mathbf{T}_2$ is

$$\begin{aligned}
& t_\omega^* \left(- \frac{U_\ell(\alpha_\ell+\beta_\ell) + \sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)}}{1+\beta_\ell U_\ell} + \sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)} \right) \\
=& -t_\omega^* \left(\frac{U_\ell(\alpha_\ell+\beta_\ell) - \beta_\ell U_\ell \sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)}}{1+\beta_\ell U_\ell} \right) \\
\overset{2}{<}& -t_\omega^* \left(\frac{U_\ell(\alpha_\ell+\beta_\ell) - \beta_\ell U_\ell}{1+\beta_\ell U_\ell} \right) < 0,
\end{aligned}$$

where $\overset{2}{<}$ comes from the condition $\sum_{i=1}^{\ell-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{\ell-1}(\beta_j U_j+1)} < 1$.

With the above two cases, either (1) we can repeatedly move one task from $\mathbf{T}_1$ to $\mathbf{T}_2$ by changing the extreme point solution S to another extreme point solution S' to improve the objective function in Eq. (40) or (2) $1 - \sum_{i=1}^{k-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j+1)} \leq 0$. Similar to the argument in the proof of Lemma 1, the minimum C_k^* happens when s is 0 if $1 - \sum_{i=1}^{k-1} \frac{U_i(\alpha_i+\beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j+1)} > 0$. □

Proof of Theorem 2. Clearly, if $\prod_{j=1}^{k-1}(U_j+1) \geq 2$, we can already conclude that $\sum_{j=1}^{k-1} U_j \geq \ln 2$ by following the same analysis in [6], [25], and the speed-up factor is $1/\ln 2 < 1.76322$ for such a case. We focus on the other case with $\prod_{j=1}^{k-1}(U_j+1) < 2$.

To understand whether the task set is schedulable under any scheduling policy, we only have to test the feasibility of preemptive EDF schedule, as preemptive EDF is an optimal scheduling policy to meet the deadlines in uniprocessor systems. Baruah et al. [4] provide a demand-bound function (dbf) test to verify such a case. That is, the demand bound function $dbf_i(t)$ of task τ_i with interval length t is

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

A system of independent, preemptable, sporadic tasks can be feasibly scheduled (under EDF) on a processor if and only if

$$\forall t \geq 0 \quad \sum_i dbf_i(t) \leq t.$$

Therefore, if there exists t such that $\frac{\sum_i dbf_i(t)}{s} > t$ or $\sum_i U_i > s$, then the task set is not schedulable by EDF on a uniprocessor platform with speed s.

Recall that we can construct the corresponding k-point effective schedulability test defined in Definition 1 with $\alpha_i = 1$ and $\beta_i \leq 1$ as shown in the proof of Theorem 1. Now, we take a look of the proof in Lemma 1 again. The same proof can also be applied to show that the extreme point solution that leads to the solution in Eq. (14) is also an optimal solution for the following linear programming when $\prod_{j=1}^{k-1}(U_j+1) < 2$:

$$\begin{aligned}
\text{infimum} \quad & C_k^* + \sum_{i=1}^{k-1} t_i^* U_i \\
\text{s.t.} \quad & C_k^* + \sum_{i=1}^{k-1} t_i^* U_i + \sum_{i=1}^{j-1} t_i^* U_i > t_j^* \geq 0, \ \forall j = 1, 2, \ldots, k.
\end{aligned}$$

That is, the corresponding objective function of Eq. (9) is $t_k + s - \beta \sum_{i=1}^{k-1} U_i t_i^*$, by setting $\alpha = 1$ and $\beta = 1$. Let C_k^* be the optimal C_k^* of the above linear programming when $\prod_{j=1}^{k-1}(U_j+1) < 2$. By the above argument, we know that $(1 + \frac{C_k^*}{D_k}) \prod_{j=1}^{k-1}(U_j+1) = 2$. Therefore, $C_k^* > 0$. Moreover,

$$\prod_{j=1}^{k-1}(U_j+1) = \frac{2}{1+\frac{C_k^*}{D_k}}. \tag{43}$$

For the rest of the proof, let C_k^*/D_k be x. If task τ_k is not schedulable by DM (or does not pass Eq. (20)), then,

$$\begin{aligned}
& \frac{C_k' + \sum_{i=1}^{k-1} dbf_i(D_k)}{D_k} \geq \frac{C_k' + \sum_{i=1}^{k-1} t_i U_i}{D_k} \\
>& \frac{C_k^* + \sum_{i=1}^{k-1} t_i^* U_i}{t_k} = \frac{t_1^*}{t_k} \\
\overset{1}{=}& \frac{1}{\prod_{j=1}^{k-1}(U_j+1)} = \frac{1+\frac{C_k^*}{D_k}}{2} = \frac{1+x}{2}
\end{aligned}$$

where $\overset{1}{=}$ comes from the relation $\frac{t_1^*}{t_k}$ in Eq. (12) when s is 0. Moreover, with Lemma 3, we have

$$\sum_{i=1}^{k-1} U_i > \ln(\frac{2}{1+x}). \tag{44}$$

Due to the fact that $\frac{1+x}{2}$ is an increasing function of x and $\ln(\frac{2}{1+x})$ is a decreasing function of x, we know that $\inf_{0 \leq x < 1} \max\left\{\frac{1+x}{2}, \ln(\frac{2}{1+x})\right\}$ is the intersection of $\frac{1+x}{2}$ and $\ln(\frac{2}{1+x})$, which is $1/1.76322$. Therefore,

$$\begin{aligned}
& \max\left\{\frac{C_k' + \sum_{i=1}^{k-1} dbf_i(D_k)}{D_k}, \sum_{i=1}^{k-1} U_i\right\} \\
>& \max\left\{\frac{1+x}{2}, \ln(\frac{2}{1+x})\right\} \geq \frac{1}{1.76322}.
\end{aligned}$$

As a result, the speed-up factor of the test in Eq. (20) for DM scheduling for constrained-deadline systems is 1.76322. □