**k²U: 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 k²U framework, that can be potentially applied to analyze a large set of real-time task models under any fixed-priority scheduling algorithm, on both uniprocessors and multiprocessors. The key to k²U is a k-point effective schedulability test, which can be viewed as a “blackbox” interface to apply the k²U framework. 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 k²U by applying it to different task models, which results in new and better 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 [30], the multi-frame task model [31], the self-suspending task model [25], 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.

Over the years, real-time systems researchers have devoted a significant amount of time and efforts to efficiently analyze different formal task models. Many successful stories have been told. For many of the above-mentioned task models, efficient scheduling and schedulability analysis techniques have been developed (see [12] for a recent survey). Unfortunately, for certain complex models such as the self-suspending task model, existing schedulability tests are rather pessimistic, particularly for the multiprocessor case (e.g., no utilization-based schedulability test exists for globally-scheduled multiprocessor self-suspending task systems). Moreover, 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 k²U, 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 τₖ under fixed-priority scheduling in uniprocessor systems, the time-demand analysis (TDA) developed in [22] can be adopted. That is, if

\[ \exists t \text{ with } 0 < t \leq D_k \text{ and } C_k + \sum_{i \in h_p(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t, \quad (1) \]

then task τₖ is schedulable under the fixed-priority scheduling algorithm, where hₚ(τₖ) is the set of tasks with higher priority than τₖ, Dₖ, Cₖ, and Tᵢ represent τₖ’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, Dₖ) for Eq. (1).

To obtain sufficient schedulability tests under fixed priority scheduling with reduced time complexity (e.g., polynomial-time), it may suffice to test only a subset of such points. This idea is implemented in the k²U 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 kᵈ-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 k²U 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 k²U is not only applicable to uniprocessor systems, and can also be applied to multiprocessor systems under global fixed-priority scheduling.

**Related Work.** An extensive amount of research has been conducted over the past forty years on verifying the schedulability of the classical sporadic task model in both uniprocessor and multiprocessor systems (see [12] for a survey of such results). Much progress has also been made in recent years on analyzing more complex task models that are more expressive.

There have been several results in the literature with respect to utilization-based, e.g., [6], [18]–[20], [26], [27], [32], and non-utilization-based, e.g., [10], [16], schedulability tests for the sporadic real-time task model and its generalizations in uniprocessor systems. The approaches in [10], [16] convert
the stair function $\left\lceil \frac{t}{T_i} \right\rceil$ in the time-demand analysis into a linear function if $t$ in Eq. (1) is large enough. The methods in [10, 16] are completely different from this paper, in which the linear function of task $\tau_i$ starts after $\frac{t}{T_{i1}} \geq k$, whereas our method is based on $k$ points, defined individually by $\tau_1, \tau_2, \ldots, \tau_k$.

Most of the existing utilization-based schedulability analysis 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 [9, 18, 27] are mainly for rate-monotonic (RM) scheduling, in which the results in [18] can be extended for arbitrary fixed-priority scheduling. Kuo et al. [19] further improve the total utilization bound by using the notion of divisibility. Lee et al. [6] use linear programming formulations for calculating total utilization bounds when a task can choose its own period. Moreover, Wu et al. [32] adopt the Network Calculus to analyze the total utilization bounds of several real-time task models.

The novelty of $k^2U$ comes from a different perspective from these approaches [18, 20, 27, 32]. 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 $\tau_k$. The critical value of task $\tau_k$ gives the difficulty of task $\tau_k$ to be schedulable under the scheduling policy. A nature form to test the schedulability of task $\tau_k$ 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 $k^2U$ analysis, in which the test by Bini et al. [6] for sporadic real-time tasks and our recent result in [26] for bursty-interference analysis are both special cases and simple implications from the $k^2U$ framework. With the hyperbolic forms, we are then able to provide many interesting observations with respect to the required quantitative 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 $k^2U$, 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, $k^2U$, that can be applied to analyze a number of complex real-time task models, on both unprocessors 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 using the $k^2U$ framework. We show the generality of $k^2U$ by applying it to several task models, which results in better or more general results 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 uniprocessor multi-frame task systems, proposed by Mok and Chen [31], our obtained utilization bound is superior to the results by Mok and Chen [31] analytically and Lu et al. [28] in our simulations.

3) For multiprocessor DAG task systems under global rate-monotonic (RM) scheduling, the capacity-augmentation factor, as defined in [24] and Sec. 7 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. [24].

4) For multiprocessor self-suspending task systems, we obtain the first utilization-based test for global RM.

Note that the emphasis of this paper is not to show that the resulting tests for different task models by applying the $k^2U$ framework are better than existing work. Rather, we want to show that the $k^2U$ 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 $k^2U$ 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, this is the first general schedulability analysis framework 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.

### 2 Sporadic Task and Scheduling Models

A sporadic task $\tau_i$ is released repeatedly, with each such invocation called a job. The $j^{th}$ job of $\tau_i$, 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 $\tau_i$ 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. Successive jobs of the same task are required to execute in sequence. Associated with each task $\tau_i$ are a period $T_i$, which specifies the minimum time between two consecutive job releases of $\tau_i$, 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 $\tau_i$ is defined as $U_i = C_i/T_i$.

A sporadic task system $\tau$ is said to be an implicit-deadline system if $D_i = T_i$ holds for each $\tau_i$. A sporadic task system $\tau$ is said to be a constrained-deadline system if $D_i \leq T_i$ holds for each $\tau_i$. Otherwise, such a sporadic task system $\tau$ 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* is to provide 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. 7 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 [27] and DM scheduling is an optimal fixed-priority scheduling policy for constrained-deadline systems [23].

To verify the schedulability of task $\tau_k$ under fixed-priority scheduling in uniprocessor systems, the time-demand analysis developed in [22] can be adopted, as discussed earlier. That is, if Eq. (1) holds, then task $\tau_k$ is schedulable under the fixed-priority scheduling algorithm.

For the simplicity of presentation, we will mainly introduce how the framework works based on the above definition of ordinary sporadic real-time task systems in Sec. 4 and Sec. 5. We will demonstrate more applications with respect to multi-frame tasks [31] in Sec. 6 and with respect to multiprocessor scheduling in Sec. 7.

3 Analysis Flow

The proposed $k^2U$ framework only works for providing the schedulability test of a specific task $\tau_k$, 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 $\tau_k$, that is being analyzed, under the above assumption. For notational brevity, we implicitly assume that there are $k-1$ tasks, says $\tau_1, \tau_2, \ldots, \tau_{k-1}$ with higher-priority than task $\tau_k$. Moreover, we only consider the cases when $k \geq 2$, since $k = 1$ is pretty trivial.

4 $k^2U$ Framework

This section presents the basic properties of the $k^2U$ framework for testing the schedulability of task $\tau_k$ 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 presentations. The concrete applications will be presented in Secs. 5-7.

The $k$-point effective schedulability test is a sufficient schedulability test by testing only $k$ time points, defined by the $k-1$ higher-priority tasks and task $\tau_k$. 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 by only testing $k$ points. However, if these $k$ points are effective 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 $\tau_k$ 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 by only testing $T_1, T_2, \ldots, T_k$. The case defined in the above example is utilized by Liu and Layland [27] 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 $f T_j \leq T_k$ for a given integer $f \geq 1$ for any higher-priority task $\tau_i$, for all $i= 1, 2, \ldots, k-1$. Let $t_i$ be $[\frac{T_j}{T_i} T_i$. 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 $\tau_k$ is schedulable under RM if there exists $j$ such that

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

where the inequality in Eq. (3) is due to the fact $C_k = T_k U_k \leq \frac{1}{f} t_k U_k$. That is, in the above example, it is sufficient by only testing $[\frac{t_k}{T_k} T_1, [\frac{t_k}{T_2} T_2, \ldots, [\frac{t_k}{T_{k-1}} T_{k-1}, T_k$.

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 by verifying the existence of $t_j \in \{t_1, t_2, \ldots, t_k\}$ with $t_1 \leq t_2 \leq \cdots \leq t_k$ 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 $\tau_i$.

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

Throughout the paper, we implicitly assume that $t_k \neq \infty$, as $t_k$ is usually related to the given relative deadline requirement. Moreover, we only consider non-trivial cases, in
which \( C_k > 0 \), \( \alpha_i \neq \infty \), \( \beta_i \neq \infty \), and \( 0 < U_i \leq 1 \) for \( i = 1, 2, \ldots, k - 1 \).

With these \( k \) points, we are able to define the corresponding coefficients \( \alpha_i \) and \( \beta_i \) in the \( k \)-point effective schedulability test of a scheduling algorithm. The elegance of the \( k^2 \)U framework is to use only the parameters \( \alpha_i \) and \( \beta_i \) defined in the \( k \)-point effective schedulability test to define the extreme case of \( C_k \) to be unschedulable under the schedulability test. Therefore, the \( k^2 \)U framework provides corresponding utilization-based tests automatically if the \( k \)-point effective schedulability test and the corresponding parameters \( \alpha_i \) and \( \beta_i \) can be defined, which will be further demonstrated in the following sections with several applications.

For the rest of the section, we are going to present the properties resulting from the \( k \)-point effective schedulability test under given \( \alpha_i \) and \( \beta_i \). 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 \( \alpha_i \) and \( \beta_i \). To make the notations clear, these extreme testing points are denoted as \( t^*_k \) for the rest of this paper. The procedure will derive \( k - 1 \) extreme testing points, denoted as \( t^*_1, t^*_2, \ldots, t^*_k \), whereas \( t^*_k \) is defined as \( t_k \) for notational brevity. Lemmas \(^1\) to \(^3\) are useful to analyze the utilization bound, the hyperbolic bound, etc., for given scheduling strategies, when \( \alpha \) and \( \beta \) can be easily defined based on the scheduling policy, with \( 0 < \alpha_i \leq \alpha \neq \infty \), and \( 0 < \beta_i \leq \beta \neq \infty \) for any \( i = 1, 2, \ldots, k - 1 \), \( 0 < t_k \neq \infty \).

**Lemma 1.** For a given \( k \)-point effective schedulability test, defined in Definition \(^7\) of a scheduling algorithm, in which \( 0 < \alpha_i \leq \alpha \neq \infty \), and \( 0 < \beta_i \leq \beta \neq \infty \) for any \( i = 1, 2, \ldots, k - 1 \), \( 0 < t_k \neq \infty \), task \( \tau_k \) is schedulable by the scheduling algorithm if the following condition holds

\[
\frac{C_k}{t_k} \leq \frac{\alpha + 1}{\prod_{i=1}^{k-1}(\beta U_i + 1)} - \frac{\alpha}{\beta}.
\]

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

\[
C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i t_i U_i \geq C_k + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \sum_{i=1}^{j-1} \beta_i t_i U_i > t_j.
\]

To enforce the condition in Eq. \(^6\), we are goin to show that \( C_k \) must have some lower bound, denoted as \( C_k^* \). Therefore, if \( C_k \) is no more than this lower bound, then task \( \tau_k \) is schedulable by the scheduling policy. The unschedulability for satisfying Eq. \(^6\) implies that \( C_k > C_k^* \), where \( C_k^* \) is defined in the following optimization problem:

\[
\text{infimum } C_k^* \tag{7a}
\]

such that \( C_k^* + \sum_{i=1}^{k-1} \alpha_i t_i^* U_i + \sum_{i=1}^{j-1} \beta_i t_i^* U_i > t_j^* \geq 0, \forall j = 1, 2, \ldots, k \),

\[
\text{where } t_1^*, t_2^*, \ldots, t_{k-1}^* \text{ and } C_k^* \text{ are variables, } \alpha, \beta \text{ are constants, and } t_k^* \text{ is defined as } t_k (\text{a given positive constant}) \text{ for notational brevity.}
\]

By Eq. \(^7b\) when \( j = k \), we know that \( C_k^* > t_k^* - (\sum_{i=1}^{k-1} \alpha_i t_i^* U_i + \sum_{i=1}^{k-1} \beta_i t_i^* U_i) \). Therefore, we can replace the objective function and the constraints with the above inequality of \( C_k^* \). The objective function is to find the infimum value of \( t_k^* - (\sum_{i=1}^{k-1} \alpha_i t_i^* U_i + \sum_{i=1}^{k-1} \beta_i t_i^* U_i) \) such that

\[
t_k^* - (\sum_{i=1}^{k-1} \alpha_i t_i^* U_i + \sum_{i=1}^{k-1} \beta_i t_i^* U_i) + \sum_{i=1}^{k-1} \alpha_i t_i^* U_i + \sum_{i=1}^{k-1} \beta_i t_i^* U_i \]

\[
= t_k^* - \sum_{i=1}^{k-1} \beta_i t_i^* U_i > t_j^*, \quad \forall j = 1, 2, \ldots, k - 1. \tag{8}
\]

For the rest of the proof, we replace \( > \) with \( \geq \) in Eq. \(^8\), as the infimum and the minimum are the same when presenting the inequality with \( \geq \). As a result, since \( t_k^* \) is a positive constant, the above objective to find the infimum is equivalent to the following linear programming:

\[
\begin{align}
\text{maximize} & \quad (\alpha + \beta) \sum_{i=1}^{k-1} U_i t_i^* \\
\text{s.t.} & \quad t_k^* - \beta \sum_{i=1}^{k-1} t_i^* U_i \geq t_j^*, \quad \forall 1 \leq j \leq k - 1, \tag{9a} \\
& \quad t_j^* \geq 0 \quad \forall 1 \leq j \leq k - 1. \tag{9c}
\end{align}
\]

The linear programming in Eq. \(^9\) has \( k - 1 \) variables and \( 2(k - 1) \) constraints. Thus, according to the extreme point theorem for linear programming \(^29\), 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 \( t_j^* \leq t_k^* \) for \( j = 1, 2, \ldots, k - 1 \), due to \( t_j^* \geq 0 \), \( 0 < \beta \neq \infty \), \( 0 < \alpha \neq \infty \) and \( U_i > 0 \) for \( i = j, j + 1, \ldots, k - 1 \). As a result, the objective of the above linear programming is finite since a feasible solution has to satisfy \( t_j^* \leq t_k^* \).

According to the extreme point theorem, one of the extreme points is the optimal solution of Eq. \(^9\). There are \( k - 1 \) variables with \( 2(k - 2) \) constraints in Eq. \(^9\). An extreme point must have at least \( k - 1 \) active constraints in Eqs. \(^9b\) and \(^9c\), in which their \( \geq \) are set to equality \(^=\). One special extreme point solution by setting \( t_j^* > 0 \) is to put \( t_k^* - \beta \sum_{i=1}^{k-1} t_i^* U_i = t_j^* \) for every \( j = 1, 2, \ldots, k - 1 \), i.e.,

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

which implies that

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

Moreover,

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

\(^2\) A constraint is said active in linear programming if there is no slack in the inequality, i.e., the inequality is equality in the solution.
Lemma 2. For a given k-point effective schedulability test, defined in Eq. (4), of a scheduling algorithm, in which \(0 < \alpha_i \leq \alpha \neq \infty\) and \(0 \leq \beta_i \leq \beta \neq \infty\) for any \(i = 1, 2, \ldots, k-1\), \(0 < t_k \neq \infty\), task \(\tau_k\) is schedulable by the scheduling algorithm if
\[
\frac{C_k}{t_k} + \sum_{i=1}^{k-1} U_i \leq \frac{(k-1)((\alpha + \beta)^{\frac{k}{k}} - 1) + ((\alpha + \beta)^{\frac{k}{k}} - \alpha)(\beta)}{\beta}.
\]

Lemma 3. For a given k-point effective schedulability test, defined in Eq. (4), of a scheduling algorithm, in which \(0 < \alpha_i \leq \alpha \neq \infty\) and \(0 < \beta_i \leq \beta \neq \infty\) for any \(i = 1, 2, \ldots, k-1\), \(0 < t_k \neq \infty\), task \(\tau_k\) is schedulable by the scheduling algorithm if
\[
\beta \sum_{i=1}^{k-1} U_i \leq \ln\left(\frac{\alpha + 1}{\alpha \beta t_k}\right).
\]

Remarks and how to use the framework: After presenting the \(k^2U\) framework, here, we explain how to use the \(k^2U\) framework and summarize how we plan to demonstrate its applicability in several task models in the following sections. The \(k^2U\) framework relies on the users to index the tasks properly and define \(\alpha_i\) and \(\beta_i\) as constants \((\neq \infty)\) for \(i = 1, 2, \ldots, k-1\) based on Eq. (4). Therefore, the \(k^2U\) framework can only be applicable when \(\alpha_i\) and \(\beta_i\) are well-defined. These constants depend on the task models and the task parameters.

How to choose good parameters \(\alpha_i\) and \(\beta_i\): While \(\alpha_i\) and \(\beta_i\) affects the quality of the resulting schedulability bounds in Lemmas 1 to 3, however, deriving the good settings of \(\alpha_i\) and \(\beta_i\) is actually not the focus of this paper. The framework does not care how the parameters \(\alpha_i\) and \(\beta_i\) are obtained. The framework simply derives the bounds according to the given parameters \(\alpha_i\) and \(\beta_i\), regardless of the settings of \(\alpha_i\) and \(\beta_i\). The correctness of the settings of \(\alpha_i\) and \(\beta_i\) is not verified by the framework.

The ignorance of the settings of \(\alpha_i\) and \(\beta_i\) 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 \(\tau_k\) in a fixed-priority scheduling policy. With the availability of the \(k^2U\) framework, the hyperbolic bounds or utilization bounds can be automatically derived by adopting Lemmas 1 to 3 as long as the safe upper bounds \(\alpha\) and \(\beta\) to cover all the possible settings of \(\alpha_i\) and \(\beta_i\) for the schedulability test in Eq. (4) can be derived, regardless of the task model or the platforms.

The other approaches in [2], [18], [20] 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 [9],
Lemma 4
Extreme points test
The $k$ framework

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

$$\sum_{\tau_i \in h p_i(\tau_k)} \left\lfloor \frac{t}{T_i} \right\rfloor C_i = \sum_{\tau_i \in h p_i(\tau_k)} C_i + \sum_{\tau_i \in h p_i(\tau_k)} \left\lfloor \frac{t}{T_i} \right\rfloor C_i.$$

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

$$C_k + \sum_{\tau_i \in h p_i(\tau_k)} C_i + \sum_{\tau_i \in h p_i(\tau_k)} \left\lfloor \frac{t}{T_i} \right\rfloor C_i \leq t.$$  (18)

We can then create a virtual sporadic task $\tau'_k$ with execution time $C'_k = C_k + \sum_{\tau_i \in h p_i(\tau_k)} C_i$, 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 $\tau'_k$ under the interference of the higher-priority tasks $h p_i(\tau_k)$ is the same as that of task $\tau_k$ under the interference of the higher-priority tasks $h p_i(\tau_k)$.

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

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

$$\left(\frac{C'_k}{D_k} + 1\right) \prod_{\tau_j \in h p_j(\tau_k)} (U_j + 1) \leq 2$$  (19)

or

$$\frac{C'_k}{D_k} + \sum_{\tau_j \in h p_j(\tau_k)} U_j \leq k(2^k - 1).$$  (20)

**Proof:** For notational brevity, suppose that there are $k - 1$ tasks in $h p_1(\tau_k)$. Now, we index the higher-priority tasks in $h p_1(\tau_k)$ to form the corresponding $\tau_1, \tau_2, \ldots, \tau_{k-1}$. The $k - 1$ higher-priority tasks in $h p_1(\tau_k)$ are ordered to ensure that the arrival times 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 $h p_1(\tau_k)$, we have $D_k$ $T_i \leq D_{i+1}$ $T_i + 1$ for $i = 1, 2, \ldots, k - 2$. Now, we set $t_i$ as $D_k$ $T_i$ 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 $t_k > 0$.

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

$$C_k + \sum_{\tau_i \in h p_2(\tau_k)} C_i + \sum_{\tau_i \in h p_1(\tau_k)} \left\lfloor \frac{t_j}{T_i} \right\rfloor C_i$$

$$= C_k + \sum_{i=1}^{k-1} \left\lfloor \frac{t_j}{T_i} \right\rfloor C_i \leq C_k + \sum_{i=1}^{k-1} \frac{t_j}{T_i} C_i + \sum_{i=1}^{j-1} C_i,$$

where the inequality comes from the indexing policy defined above, i.e., $\left\lfloor \frac{t_j}{T_i} \right\rfloor \leq \frac{t_j}{T_i} + 1$ if $j > i$ and $\left\lfloor \frac{t_j}{T_i} \right\rfloor \leq \frac{t_j}{T_i}$ if $j \leq i$.

Since $T_i < D_k$ for any task $\tau_i$ in $h p_1(\tau_k)$, we know that $t_j \geq T_i$. Instead of testing all the $t$ values in Eq. (18), we only apply the test for these $k$ different $t_i$ values, which is when $h p_2(\tau_k)$ is not empty, there are $k - 1 - |h p_2(\tau_k)|$ tasks in $h p_1(\tau_k)$.
Theorem 1 and Corollary 2

Theorem 3

Theorems 5 and 6 in [26].

c.f.

Therefore, we reach the conclusion.

When RM priority ordering is applied for an implicit-deadline task system, $C_k'$ is equal to $C_k$ and $C_k'/D_k$ is equal to $U_k$. For such a case, the condition in Eq. (19) is the same as the hyperbolic bound provided in [6], and the condition in Eq. (20) is the same as least utilization upper bound in [27].

It may seem pessimistic by using 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. (19) 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 by using the schedulability test in Eq. (19).

Theorem 2. The speed-up factor of the schedulability test in Eq. (19) 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 $\tau_i$ in $h_{p1}(\tau_k)$, where $f$ is a positive integer. Task $\tau_k$ in a constrained-deadline sporadic task system is schedulable by the fixed-priority scheduling algorithm if

$$\left(\frac{C_k}{f \cdot D_k} + 1\right) \prod_{\tau_j \in h_{p1}(\tau_k)} \left(\frac{U_j}{f} + 1\right) \leq \frac{f + 1}{f}$$

or

$$\frac{C_k'}{D_k} + \sum_{\tau_j \in h_{p1}(\tau_k)} U_j \leq f k \left(\left(\frac{f + 1}{f}\right)^{\frac{1}{f}} - 1\right).$$

Proof: This is based on the same proof as Theorem 1 by taking the fact that $\frac{u}{T_i} = \frac{\frac{C_k}{f}}{D_k} \geq f$. In the $k$-point effective schedulability test, we can set $\alpha_i$ to 1, $\beta_i = \frac{T_k}{T_i} \leq \frac{1}{f}$. Therefore, we have $\alpha_i \leq \alpha = 1$, $\beta_i \leq \beta = \frac{1}{f}$, and $\frac{1}{f}$ is $f$. By adopting Lemma 1, we know that task $\tau_k$ is schedulable under the scheduling policy if

$$\left(\frac{C_k'}{D_k} + f\right) \prod_{\tau_j \in h_{p1}(\tau_k)} \left(\frac{U_j}{f} + 1\right) \leq f + 1,$$

which is the same as the condition in Eq. 22 by dividing both sides by $f$. By using a similar argument and applying Lemma 2 we can reach the condition in Eq. 23.

Note that the right-hand side of Eq. 23 converges to $f \ln\left(\frac{f+1}{f}\right)$ when $k$ goes to $\infty$.

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 [21]. 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 $\tau_k$ 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 $\tau_k$ finishes before the next release of a job of task $\tau_k$.

It has been shown in [21] that the worst-case response time of task $\tau_k$ can be found in one of the jobs of task $\tau_k$ in the busy window.

Therefore, a simpler sufficient schedulability test for a task $\tau_k$ is to test whether the length of the busy window is within $D_k$. If so, all invocations of task $\tau_k$ 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

$$\exists t < D_k$$

and

$$t \leq \frac{C_k}{D_k} + \sum_{\tau_j \in h_{p1}(\tau_k)} \left(\frac{t}{T_i}\right) C_j \leq t.$$  

If the condition in Eq. 24 holds, it implies that the busy window (when considering task $\tau_k$) is no more than $D_k$, and, hence, task $\tau_k$ 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 \( \tau_k \) as shown above. For the rest of this section, we will focus on the case \( D_k > T_k \). We can rewrite Eq. (24) to use a more pessimistic case by releasing the workload \[ \frac{D_k}{T_k} \] \( C_k \) at time 0. That is, if

\[
\exists t \text{ with } 0 < t \leq D_k \text{ and } \frac{D_k}{T_k} \left[ \begin{array}{c} C_k + \sum_{\tau_i \in h_p(\tau_k)} \frac{t}{T_i} C_i \leq t, \\
\end{array} \right.
\]

(25)

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

Similarly, we can then create a virtual sporadic task \( \tau'_k \) with execution time \( C'_k = \frac{D_k}{T_k} C_k + \sum_{\tau_i \in h_p(\tau_k)} C_i \), relative deadline \( D'_k = D_k \), and period \( T'_k = D_k \). For notational brevity, suppose that there are \( k - 1 \) tasks in \( h_p(\tau_k) \). Now, we index the higher-priority tasks in \( h_p(\tau_k) \) to form the corresponding \( \tau_1, \tau_2, \ldots, \tau_{k-1} \). In the above definition of the busy window concept, \( \frac{D_k}{T_k} \) is the arrival time of the last job of task \( \tau_1 \) released no later than \( D_k \). The \( k - 1 \) higher-priority tasks in \( h_p(\tau_k) \) are ordered to ensure that the arrival times of the last jobs before \( D_k \) in a non-decreasing order. Moreover, \( t_k \) is the specified testing point \( D_k \). Instead of testing all the \( t \) values in Eq. (25), 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. (21) 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.

Therefore, we can then use the \( k^2 \) U framework, i.e., Lemmas 1 to 3, to test the schedulability of task \( \tau_k \). The following corollary comes from a similar argument as in Sec. 5.1

**Corollary 2.** Task \( \tau_k \) in a sporadic arbitrary-deadline task system is schedulable by the fixed-priority scheduling algorithm if Eq. (19) or (20) holds, in which there are \( k - 1 \) higher priority tasks in \( h_p(\tau_k) \) and \( C'_k \) is defined as \[ \frac{D_k}{T_k} C_k + \sum_{\tau_i \in h_p(\tau_k)} C_i \].

**Corollary 3.** Suppose that \( f : T_i \leq D_k \) for any higher task \( \tau_i \) in the task system and \( f : T_k \leq D_k \), where \( f \) is a positive integer. Task \( \tau_k \) in a sporadic task system is schedulable by using RM, i.e., \( T_i \leq T_i + 1 \), if

\[
\frac{U_k}{f} + 1 \sum_{j=1}^{k-1} \frac{U_j}{f} + 1 \leq \frac{f + 1}{f}
\]

(26)

or

\[
\sum_{j=1}^{k} U_j \leq f(k + 1) \left( \frac{f + 1}{f} - 1 \right)^{1/2}.
\]

(27)

**Proof:** Based on the above assumption \( f : T_i \leq D_k \) for any higher task \( \tau_i \) in the task system, \( f : T_k \leq D_k \), and the rate monotonic scheduling, we can safely set \( \alpha_i \leq 1 \), \( \beta_i \leq \frac{1}{f} \), and \( C'_k \) to \( f : C_k \). Note that \[ \frac{C_k}{T_k} \leq \frac{U_k}{T_k} \] 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 Multi-frame Tasks

This section adopts the schedulability test framework in Sec. 4 for multi-frame real-time tasks, proposed by Mok and Chen [31]. A multi-frame real-time task \( \tau_i \) with \( m_i \) frames is defined as a sporadic task with period \( T_i \) with an array \( C_i, 0, C_i, 1, \ldots, C_i, m_i - 1 \) of different execution times. The execution time of the \( j \)-th job of task \( \tau_i \) is defined as \( C_i, (j \mod m_i) \).

Mok and Chen [31] propose a utilization-based schedulability under rate monotonic (RM) scheduling by generalizing the Liu & Layland bound [27] for the multi-frame task. Kuo et al. [19] present a more precise schedulability test by merging the tasks with harmonic periods before inspecting the Mok & Chen bound. The researches in [20], [32] also demonstrate how to apply their methods to handle the multi-frame task model. More recently, Lu et al. [28] further consider the ratio between periods to improve the existing utilization-based test.

For a multi-frame task, we define the utilization \( U_i \) of task \( \tau_i \) based on its peak utilization, i.e., \[ U_i = \max_{j=0}^{m_i - 1} C_i, j \]. Without loss of generality, we assume that each task has at least two frames, i.e., \( m_i \geq 2 \). If a task has only one frame, we can artificially create a corresponding multi-frame task with 2 frames and with the same execution time. We will limit our attention in uniprocessor systems.

Let \( \phi_i(\ell) \) be the maximum of the sum of the execution time of any \( \ell \) consecutive frames of task \( \tau_i \). It is clear that \( \phi_i(1) = \max_{j=0}^{m_i - 1} C_i, j \) and \( \phi_i(2) = \max_{j=0}^{m_i - 1} (C_i, j + C_i, (j \mod m_i)) \). Therefore, we know that \( U_i \) is equal to \( \phi_i(1) \). For brevity, we define \( \phi_i(0) \) as 0. It is not difficult to see that \( \phi_i(\ell) \) is equal to \( \phi_i(\ell \mod m_i) + \left( \frac{\ell}{m_i} \right) \sum_{j=0}^{m_i - 1} C_i, j \) when \( \ell > m_i \), where \( \phi_i(0) \) is set to 0 for notational brevity. Therefore, we only need to build a table for the first \( m_i \) entries to construct \( \phi_i(\ell) \). Deriving \( \phi_i(\ell) \) can be done in \( O(m_i^2) \) for \( \ell = 1, 2, \ldots, m_i - 1 \).

Again, we consider testing the schedulability of task \( \tau_k \) under RM scheduling, in which there are \( k - 1 \) higher-priority multi-frame tasks \( \tau_1, \tau_2, \ldots, \tau_{k-1} \). We have the following schedulability condition for RM.

**Lemma 5.** Suppose that all the multi-frame tasks with higher priority than \( \tau_k \), i.e., \( \tau_1, \tau_2, \ldots, \tau_{k-1} \), are schedulable by RM. Multi-frame task \( \tau_k \) is schedulable under RM on a uniprocessor, if

\[
\exists t \text{ with } 0 < t \leq T_k \text{ and } \phi_k(1) + \sum_{i=1}^{k-1} \phi_i \left( \frac{t}{T_i} \right) \leq t.
\]

(28)

**Proof:** This comes from Theorem 5 and Lemma 6 by Mok and Chen in [31].

We now present a more pessimistic analysis than Eq. (28) to construct a \( k \)-point effective schedulability test. Let \( \delta_i(j) \) be \( \phi_i(j) - \phi_i(j - 1) \). That is, \( \delta_i(j) \) is the additional workload released from the \( j \)-th invocation of task \( \tau_i \) in the definition of \( \phi_i(\ell) \). Moreover, let \( \delta_i^{\min}(\ell) \) be \( \min_{j=1,2,\ldots,k} \delta_i(j) \), i.e., \( \delta_i^{\min}(\ell) \) is the minimum \( \delta_i(j) \) among the first \( \ell \) release of task \( \tau_i \).
We further define \( \phi_i'(\ell) \) as follows:

\[
\phi_i'(\ell) = \phi_i(\ell + 1) - \delta^\text{min}_i(\ell + 1)
\]

The definition of \( \phi_i'(\ell) \) comes from the operation by swapping the increased workload of the \( \ell + 1 \)-th release of task \( t_i \) with the workload \( \delta^\text{min}_i(\ell + 1) \).

Again, let \( t_i \) be \( \left\lfloor \frac{T_i}{T_k} \right\rfloor T_i \) for \( i = 1, 2, \ldots, k \), and reindex the tasks such that \( t_1 \leq t_2 \leq \cdots \leq t_k \). Instead of testing all the \( t \) values in Eq. (29) by referring to \( \phi_i() \), we only apply the test for these \( k \) different \( t_i \) values by referring to \( \phi_i'(\ell) \) as shown in the following lemma.

**Lemma 6.** Multi-frame task \( \tau_k \) is schedulable under RM on a uniprocessor, if there exists \( t_j \) such that

\[
\phi_k'(1) + \sum_{i=1}^{k-1} \phi_i'(t_i) + \delta^\text{min}_i(1) \leq t_j,
\]

where \( t_i = \left\lfloor \frac{T_i}{T_k} \right\rfloor T_i \) for \( i = 1, 2, \ldots, k \) and \( t_i \leq t_{i+1} \).

**Proof:** This property is due to the fact that \( \phi_i'(\ell) \geq \phi_i(\ell) \) for any \( 0 \leq \ell \leq \left\lfloor \frac{T_i}{T_k} \right\rfloor \) according to the definition of \( \phi_i'() \).

**Theorem 3.** Let \( f \) be \( \frac{\phi_i(1) - T_i}{\delta^\text{min}(T_i + 1)} \), where \( \ell_i \) is defined as \( \frac{t_i}{T_i} \) for notational brevity. Task \( \tau_k \) is schedulable under RM if

\[
\left( \frac{U_k}{f} + 1 \right) \prod_{j=1}^{k-1} \left( \frac{U_j}{f} + 1 \right) \leq \frac{f + 1}{f}
\]

or

\[
\sum_{j=1}^{k} U_j \leq f k \left( \left( \frac{f + 1}{f} \right)^{\frac{1}{f}} - 1 \right),
\]

where \( f \) is \( \min_{i=1}^{k-1} f_i \). Moreover, \( f_i \geq \frac{\phi_i(1)}{\phi_i(1) - \phi_i(T_i)} \).

**Proof:** By Lemma 6, task \( \tau_k \) is schedulable by RM if there exists \( t_j \) such that

\[
\phi_k(1) + \sum_{i=1}^{k-1} \alpha_i t_i U_i + \beta_i t_i U_i \leq t_j,
\]

where, for a higher-priority task \( \tau_i \), we know \( \alpha_i = \frac{\phi_i(T_i)}{\phi_i(1) - \phi_i(T_i)} \leq 1 \) and \( \beta_i = \frac{\delta^\text{min}(T_i + 1)}{\phi_i(1) - \phi_i(T_i)} = \frac{1}{T_i} \cdot \). Then, suppose that \( \beta_i \leq \frac{1}{f} \). The rest of the proof is the same as in the proof of Corollary 1 to reach the schedulability conditions in Eqs. (31) and (32).

Due to RM scheduling policy, we know that \( \ell_i \geq 1 \). Based on the definition of function \( \delta^\text{min}(\ell_i) \), we know that \( \delta^\text{min}(\ell) \) is a non-increasing function with respect to \( \ell \). Therefore, since \( \ell_i \geq 1 \), we know that \( \delta^\text{min}(T_i + 1) \leq \delta^\text{min}(2) = \phi_i(2) - \phi_i(1) \).

With \( \ell_i \geq 1 \) and \( \delta^\text{min}(T_i + 1) \leq \phi_i(2) - \phi_i(1) \), we know that \( f_i \geq \frac{\phi_i(1)}{\phi_i(2)} \). The hyperbolic bound test in Eq. (31) is the first one for multi-frame tasks. The result of the utilization bound in Eq. (32) is the same as the result by Mok and Chen [31] when \( f \) is set to \( \min_{i=1}^{k-1} \frac{\phi_i(1)}{\phi_i(2) - \phi_i(1)} \).

**7 Application for Multiprocessor Scheduling**

It may seem, at first glance, that the \( k^2U \) 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, in which each of them has 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 \( \sum_{i=1}^{M} \frac{C_i}{T_k} \leq \frac{1}{b} \) and the maximum utilization \( \max_{i=1}^{M} \frac{C_i}{T_k} \leq \frac{1}{b} \), it is possible to provide the schedulability guarantee of global RM by setting \( b = 3 - \frac{1}{M} \). Such a factor \( b \) has been recently named as a capacity augmentation factor [24].

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

\[
W_i(t) = \left( \left\lfloor \frac{t}{T_i} \right\rfloor - 1 \right) C_i + 2 C_i.
\]

That is, we allow the first release of task \( \tau_i \) to be inflated by a factor \( 2 \), whereas the other jobs of task \( \tau_i \) have the same execution time \( C_i \). Again, we consider testing the schedulability of task \( \tau_k \) under global RM, in which there are \( k \) and \( k \)-higher-priority tasks \( \tau_1, \tau_2, \ldots, \tau_{k-1} \). We have the following schedulability condition for global RM.

**Lemma 7.** Task \( \tau_k \) is schedulable under global RM on \( M \) identical processors, if

\[
\forall t \text{ with } 0 < t \leq T_k \text{ and } C_k + \sum_{i=1}^{k-1} \frac{W_i(t)}{M} \leq t,
\]

where \( W_i(t) \) is defined in Eq. (34).

**Proof:** This has been shown in Sec. 3.2 (The Basic Multiprocessor Case) in [17].

**Theorem 4.** Task \( \tau_k \) in a sporadic implicit-deadline task system is schedulable by global RM on \( M \) processors if

\[
\frac{C_k}{T_k} + \frac{1}{M} \prod_{j=1}^{k-1} \left( \frac{U_j}{M} + 1 \right) \leq 3,
\]

or

\[
\frac{1}{M} \sum_{j=1}^{k-1} \frac{U_j}{M} \leq \ln \left( \frac{3}{M} \right) - 1.
\]

**Proof:** Let \( t_i \) be \( \left\lfloor \frac{T_i}{T_k} \right\rfloor T_i \) for \( i = 1, 2, \ldots, k \), and reindex the tasks such that \( t_1 \leq t_2 \leq \cdots \leq t_k \). By testing only
these \( k \) points in the schedulability test in (35) 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 \( k^2U \) framework. By Lemma 3 and Lemma 6 we have concluded the proof. \( \square \)

Note that Theorem 4 is not superior to the known analysis for sporadic task systems [1], [2], [4], as the schedulability condition in Lemma 7 is too pessimistic. This is only used as the basis to analyze a sporadic task system with implicit deadlines, as the schedulability of task \( \tau_k \) is also needed due to the unpredictable nature of I/O operations. We have the following lemma, in which the proof is in Appendix B.

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

**Proof:** Suppose that \( \sum \Psi_i \leq \frac{1}{b} \) and \( \Psi_i \leq \max_i \frac{\Psi_i}{T_k} \leq \frac{1}{b} \). Therefore, by Eq. (40), we can guarantee the schedulability of task \( \tau_k \) if \( \frac{1}{b} \leq \ln \frac{3}{2T_k} \). This is equivalent to solving \( x = \ln \frac{3}{2x^2} \), which holds when \( x \approx 3.62143 \) by solving the equation numerically. Therefore, we reach the conclusion of the capacity augmentation factor 3.62143. \( \square \)

### 7.1 Global RM for DAG Task Systems

For multiprocessor scheduling, the DAG task model has been recently studied [8]. The utilization-based analysis can be found in [24] and [8]. Each task \( \tau_i \in T \) 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 \( \tau_i \) 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 \( \tau_i \):

- **total execution time (or work)** \( C_i \) of task \( \tau_i \); This is the summation of the execution times of all the subtasks of task \( \tau_i \) among all the DAGs of task \( \tau_i \).
- **critical-path length** \( \Psi_i \) of task \( \tau_i \); 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 \( \tau_i \).

The analysis is based on the two given parameters \( C_i \) and \( \Psi_i \). 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 \( \Psi_i \). With the above definition, we have the following lemma, in which the proof is in Appendix B.

**Lemma 8.** Task \( \tau_k \) in a sporadic DAG system with implicit deadlines is schedulable under global RM on \( M \) identical processors, if

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

where \( W_i(t) \) is defined in Eq. (34).

**Theorem 5.** Task \( \tau_k \) in a sporadic DAG system with implicit deadlines is schedulable by global RM on \( M \) processors if

\[
\left( \frac{\Psi_k}{T_k} + 2 \right) \prod_{j=1}^{k} \left( \frac{U_j}{M} + 1 \right) \leq 3
\]

or

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

**Proof:** Based on Lemma 8, which is very similar to Lemma 7, we can perform a similar transformation as in Theorem 4 in which \( \alpha_i \leq \frac{2}{M} \) and \( \beta_i \leq \frac{1}{M} \). By adopting Lemma 1, we have that if

\[
\left( \frac{\Psi_k + C_k - \Psi_k}{T_k} + 2 \right) \prod_{j=1}^{k} \left( \frac{U_j}{M} + 1 \right) \leq 3,
\]

then task \( \tau_k \) is schedulable. According to the fact that \( C_k - \Psi_k \leq C_k \), we know that \( \left( \frac{\Psi_k + C_k - \Psi_k}{T_k} + 2 \right) \leq \left( \frac{\Psi_k}{T_k} + 2 \right) \cdot \left( \frac{U_j}{M} + 1 \right) \). Therefore, if the condition in Eq. (39) holds, the condition in Eq. (41) also holds, which implies the schedulability. With the result in Eq. (39), we can use the same procedure as in Lemma 3 to obtain Eq. (40). \( \square \)

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

**Proof:** Suppose that \( \sum \Psi_i \leq \frac{1}{b} \) and \( \frac{\Psi_i}{T_k} \leq \max_i \frac{\Psi_i}{T_k} \leq \frac{1}{b} \). Therefore, by Eq. (40), we can guarantee the schedulability of task \( \tau_k \) if \( \frac{1}{b} \leq \ln \frac{3}{2x^2} \). This is equivalent to solving \( x = \ln \frac{3}{2x^2} \), which holds when \( x \approx 3.62143 \) by solving the equation numerically. Therefore, we reach the conclusion of the capacity augmentation factor 3.62143. \( \square \)

### 7.2 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 [26]. In [26], a general interference-based analysis framework was developed that can be applied to derive sufficient utilization-based tests for self-suspending task systems on uniprocessors.

Similar to sporadic tasks, a self-suspending task releases jobs sporadically. Jobs alternate between computation and suspension phases. We assume that each self-suspending task \( \tau_i \) 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 \( \tau_i \in T \); 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.

**Lemma 9.** Task \( \tau_k \) in a self-suspending system with implicit deadlines is schedulable under global RM on \( M \) identical processors, if

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

where \( W_i(t) \) is defined in Eq. (34).

**Theorem 6.** Task \( \tau_k \) in a sporadic self-suspending system with implicit deadlines is schedulable by global RM on \( M \) processors if

\[
\exists t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + \sum_{i=1}^{k} \frac{W_i(t)}{M} \leq t,
\]
processors if
\[
\left( \frac{C_k + S_k}{T_k} + 2 \prod_{j=1}^{k-1} \frac{U_j}{M} + 1 \right) \leq 3. \tag{43}
\]

Proof: By Lemma 9, we can perform a similar transformation as in Theorem 4 with \( \alpha_i \leq \frac{2}{3} \) and \( \beta_i \leq \frac{1}{3} \).

\[\Box\]

8 Conclusion

With the presented applications, we believe that the general schedulability analysis \( k^2U \) 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 \( k^2U \) framework are better than existing work, some analysis results by applying the \( k^2U \) framework have been shown superior to the state of the art.

Appendix C provides some case studies with evaluation results of some selected utilization-based schedulability tests. Appendix D further provides some additional properties that come directly from the \( k^2U \) framework. These properties were not directly used in any of the demonstrated examples in this paper, but they may be helpful in other cases.

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. 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.
[4] 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.
[5] E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. Real-Time Systems, 30(1-2):129–154, 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] E. Bini, T. H. C. Nguyen, P. Richard, and S. K. Baruah. A response-time bound in fixed-priority scheduling with arbitrary deadlines. IEEE Transactions on Computers, 58(2):279, 2009.
[8] V. Bonifaci, A. Marchetti-Spaccamela, S. Stiller, and A. Wiese. Feasibility analysis in the sporadic dag task model. In ECRTS, pages 225–233, 2013.
[9] A. Burchard, J. Liebeherr, Y. Oh, and S. H. Son. New strategies for assigning real-time tasks to multiprocessor systems. pages 1429–1442, 1995.
[10] S. Chakrabarty, S. Künzli, and L. Thiele. Approximate schedulability analysis. In IEEE Real-Time Systems Symposium, pages 159–168, 2002.
[11] J.-J. Chen and K. Agrawal. Capacity augmentation bounds for parallel dag tasks under G-EDF and G-RM. Technical Report 845, Faculty for Informatik at TU Dortmund, 2014.
[12] R. Davis and A. Burns. A survey of hard real-time scheduling for multiprocessor systems. Journal of ACM Computing Surveys, 43(4):35, 2011.
[13] R. Davis, T. Rothvöll, 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.

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 tests in Theorems 8 and 9 can be modified to the following theorems:

Theorem 7. An implicit-deadline multiframe system \( \tau \) is schedulable under RM if
\[
\prod_{\tau_i \in \mathcal{C}} \left( \frac{U_i}{f} + 1 \right) \leq \left( 1 + \frac{1}{f} \right)
\]
for each frame \( f \), where \( \mathcal{C} \) is the set of all critical frames.

Theorem 8. An implicit-deadline DAG system \( \tau \) is schedulable...
by global RM on M processors if
\[(\Delta_{\text{max}} + 2) \prod_{i=1}^{M} \frac{U_i}{U} + 1 \leq 3, \tag{45}\]
where \(\Delta_{\text{max}} = \max_{t_i \in T'} \frac{\omega_i}{T_i'}.

Appendix B: Proofs

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

Let \(\lambda\) be the Lagrange Multiplier and \(F\) be \(\frac{C_k}{U} + (k-1)U_1 - \lambda \left( \frac{C_k}{U} + \frac{k}{2}\right)(\beta U_1 + 1)^{-k-1} - \left(\frac{k}{2} + 1\right)\). The minimum \(\frac{C_k}{U} + (k-1)U_1\) happens when \(\frac{\partial F}{\partial U_1} = \frac{\beta U_1}{1 + \beta U_1} \) and \(\frac{\partial F}{\partial \lambda} = 1 - \lambda(\beta U_1 + 1)^{-k-1} = 0\). When \(k \geq 2\), by reorganizing the above two equations, we have \(1 - \lambda(\beta U_1 + 1)^{-k} = 0\). By the Lagrange Multiplier method, the minimum happens when \(\frac{C_k}{U} = \frac{k}{2}\) and \(1 - \lambda(\beta U_1 + 1)^{-k} = 0\). By solving the above equation, the non-linear programming is minimized when \(U_1 = \frac{\alpha + \beta}{\beta} + 1\) and \(\frac{C_k}{U} = \frac{(\alpha + \beta)^{\frac{k}{2} - 1}}{\beta}\). By the above analysis, we reach the conclusion in Eq. (45).

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 \(\prod_{i=1}^{k-1} (\beta U_i + 1) > \frac{\alpha + \beta}{\beta} + \pi\) 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 as follows is the same as in the proof of Lemma 1

\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{k-1} (\alpha_i + \beta_i)U_i t^\ast_i \tag{46a} \\
\text{s.t.} & \quad t^\ast_i - \sum_{i<j} (\beta_i t^\ast_i U_i) \geq t^\ast_j \quad \forall 1 \leq j \leq k-1, \tag{46b} \\
& \quad t^\ast_j \geq 0 \quad \forall 1 \leq j \leq k-1. \tag{46c}
\end{align*}

Similarly, a feasible extreme point solution can be represented by two sets \(T_1\) and \(T_2\) of the \(k-1\) higher-priority tasks, in which \(t^\ast_j = 0\) if \(\tau_j\) is in \(T_1\) and \(t^\ast_j > 0\) if task \(\tau_j\) is in \(T_2\).

One specific extreme point solution is to have \(T_1 = \emptyset\). For such a case, we can use the same steps from Eq. (10) to Eq. (12), where
\[t^\ast_{i} t^\ast_{i} = \prod_{j=1}^{k-1} t^\ast_{i+1} = \prod_{j=i}^{k-1} \left(\beta_j U_j + 1\right). \tag{47}\]

The resulting objective function of this extreme point solution for Eq. (10) is \(t^\ast_{k} \left(\sum_{i=1}^{k-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} \beta_j U_j} + 1\right)\).

We will show, in the rest of the proof, that the above extreme point solution is either optimal for the objective function of Eq. (46) or \(\sum_{i=1}^{k-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} (\beta_j U_j + 1)} < 1\). For a feasible extreme point solution with \(T_1 \neq \emptyset\), we will convert it to the above extreme point solution with \(T_1 = \emptyset\) by steps, in which each step moves one task from \(T_1\) to \(T_2\) by increasing the objective function. For the rest of the proof, we start from a feasible extreme point solution, specified by \(S = \{T_1, T_2\}\). Suppose that \(\tau_i\) is the first task in this extreme point solution \(S\) with \(t^\ast_i\) set to 0, i.e., \(t^\ast_{k} - \sum_{i=1}^{k-j} \beta_j t^\ast_i U_i = t^\ast_j\) for \(j = 1, 2, \ldots, k-1\).

Assume that \(\omega > \ell\) is the index of the next task with \(t^\ast_{\omega} > 0\) in the extreme point solution \(S\), i.e., \(t^\ast_{\omega} = t^\ast_{\omega+1} = \cdots = t^\ast_{\omega+\ell} = 0\). If all the remaining tasks are with \(t^\ast_i = 0\) for \(\ell \leq i \leq k-1\), then \(\omega\) is set to \(k\) and \(t^\ast_{k}\) is \(t^\ast_{k}\). If \(k\) is 1, we can easily set \(t^\ast_{1}\) to \(\frac{1}{1 + \frac{k}{\beta}_{i=1} U_i}\), which is > 0 and the objective function of the linear programming becomes larger. We focus on the cases where \(\ell > 1\).

We can conclude the following conditions by using the same steps from Eq. (10) to Eq. (12):
\[t^\ast_{i+1} - t^\ast_{i} = \beta_i U_i \quad \text{for} \quad i = 1, 2, \ldots, \ell-2 \quad \text{and} \quad t^\ast_{\ell} - t^\ast_{\ell-1} = \beta_{\ell-1} U_{\ell-1}. \]
Therefore, \(\sum_{i=1}^{\ell-1} (\alpha_i + \beta_i)U_i t^\ast_{i} = \sum_{i=1}^{\ell-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} \beta_j U_j} + 1\). There are two cases:

Case 1: If \(\sum_{i=1}^{\ell-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} \beta_j U_j} \geq 1\), then we can conclude
\[\sum_{i=1}^{k-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} \beta_j U_j} + 1 > 1,\]
\[\sum_{i=1}^{k-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} \beta_j U_j} > 1,\]
where \(> 1\) comes from the assumption \(\sum_{i=1}^{k-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} \beta_j U_j} \geq 1\) and \(\alpha\ell > 0\).

Case 2: If \(\sum_{i=1}^{\ell-1} \frac{U_i (\alpha_i + \beta_i)}{\prod_{j=1}^{i} \beta_j U_j} < 1\), then we can greedily set \(t^\ast_{i} > 0\) (i.e., move task \(\tau_i\) from \(T_1\) to \(T_2\)). Such a change of \(\tau_i\) from \(T_1\) to \(T_2\) has no impact on task \(\tau_j\) with \(i > \ell\), but has impact on all the tasks \(\tau_j\) with \(i \leq \ell\). That is, after changing, by using the same steps from Eq. (10) to Eq. (12), we have
\[t^\ast_{i+1} - t^\ast_{i} = \beta_i U_i \quad \text{for} \quad i = 1, 2, \ldots, \ell-1 \quad \text{and} \quad t^\ast_{\ell} - t^\ast_{\ell} = \beta_{\ell} U_{\ell}.\]
The change of the objective function in Eq. (46) is
\[
\frac{t^* \left( U_t(a_t + b_t) + \sum_{i=1}^{t-1} \frac{U_t(a_i + b_i)}{1 + \beta_i U_t} - \sum_{i=1}^{t-1} \frac{U_t(a_i + b_i)}{1 + \beta_i U_t} + 1 \right)}{1 + \beta_i U_t} = \frac{t^* U_t(a_t + b_t)}{1 + \beta_i U_t} - \sum_{i=1}^{t-1} \frac{U_t(a_i + b_i)}{1 + \beta_i U_t} + 1.
\]

where \( \frac{2}{1 + \beta_i U_t} \) comes from the condition \( \sum_{i=1}^{t-1} \frac{U_t(a_i + b_i)}{1 + \beta_i U_t} < 1 \).

Therefore, with case 1, we can conclude that \( t_k \) is not schedulable by the scheduling policy for any \( C_k > 0 \) since the interference from the higher-priority tasks is sufficiently large to disallow any execution of task \( t_k \). With case 2, we can repeatedly move one task from \( T_1 \) to \( T_2 \) by changing the extreme point solution \( S \) to another extreme point solution \( S' \) to improve the objective function in Eq. (46). That is, the objective function in Eq. (46) is maximized when \( T_1 \) is an empty set. Therefore, we reach the conclusion. \( \square \)

**Proof of Theorem 2** Clearly, if \( \prod_{j=1}^{k-1} (U_j + 1) \geq 2 \), we can already conclude that \( \sum_{j=1}^{k-1} U_j \geq \ln 2 \), and the speed-up factor is \( 1/\ln 2 < 1.76322 \) for such a case. We focus on the other case with \( \prod_{j=1}^{k-1} (U_j + 1) < 2 \), in which \( C_k^* \geq 0 \). The condition \( \prod_{j=1}^{k-1} (U_j + 1) < 2 \) also implies that \( \sum_{j=1}^{k-1} U_j < 1 \).

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. [3] provide a demand-bound function (dbf) test to verify such a case. That is, the demand bound function \( dbf_i(t) \) of task \( \tau_i \) 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 one processor if and only if
\[
\forall t \geq 0 \sum_i \dbf_i(t) \leq t.
\]

Therefore, if there exists \( t \) such that \( \sum_i \frac{\dbf_i(t)}{s} > t \) 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 2 with \( a_k = 1 \) 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. (47) is also an optimal solution for
\[
\text{inf} \sum_{i=1}^{k-1} t_i U_i + \sum_{i=1}^{k-1} t_i U_i > 0, \forall j = 1, 2, \ldots, k.
\]

That is, the corresponding objective function of Eq. (49) is \( \prod_{j=1}^{k-1} (U_j + 1) = 2 \). This implies that
\[
C_k^* = \sum_{i=1}^{k-1} \left( \frac{t_i}{D_k} \right) = \frac{2}{1 + \frac{C_k^*}{D_k}} \geq \frac{2}{1 + x}.
\]

Therefore, we know that if task \( t_k \) is not schedulable by DM, then,
\[
C_k^* + \sum_{i=1}^{k-1} dbf_i(D_k) > C_k^* + \sum_{i=1}^{k-1} t_i U_i
\]

\[
\prod_{j=1}^{k-1} (U_j + 1) = \frac{1 + \frac{C_k^*}{D_k}}{2} \geq \frac{1 + x}{2}.
\]

where \( \prod_{j=1}^{k-1} (U_j + 1) \) comes from the relation \( \frac{t_i}{D_k} \) in Eq. (12).

Moreover, with Lemma 3 we have
\[
\sum_{i=1}^{k-1} U_i > \frac{2}{1 + x}.
\]

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 \( \inf_{0 < x < 1} \max \left\{ \frac{1 + x}{2}, \ln(\frac{2}{1 + x}) \right\} \) is the intersection of \( \frac{1 + x}{2} \) and \( \ln(\frac{2}{1 + x}) \), which is 1/1.76322. Therefore,
\[
\max \left\{ \frac{C_k^* + \sum_{i=1}^{k-1} dbf_i(D_k)}{D_k}, \sum_{i=1}^{k-1} U_i \right\} \geq \frac{1 + x}{2} \geq \frac{1}{1.76322}.
\]

As a result, the speed-up factor of the schedulability test in Eq. (49) for DM scheduling for constrained-deadline systems is 1.76322. \( \square \)

**Proof of Lemma 8** This is based on the simple observations in the previous results, e.g., [2], [17], [24]. We prove by contrapositive. Suppose that a job of task \( t_k \) misses its deadline. Let the arrival time of this job be \( t \) and the absolute deadline be \( a + D_k \). Let \( X \) be the total amount of time in \( (a, a + D_k) \), in which at least one processor is not executing any job. Due to the assumption that \( t_k \) misses its deadline, the DAG structure of task \( t_k \) and the global RM scheduling policy, we know that \( X \leq \Psi_k \). The workload resulting from the higher-priority tasks in \( (a, a + t) \) is at most \( W_i(t) \), by greedily considering that the job of \( t_k \) released before \( a \) is completely not executed before \( a \). This part is pessimistic enough to be independent upon the
DAG structure. Therefore, we know that the unschedulability of task \( \tau_k \) implies that

\[
\forall t \text{ with } 0 < t \leq T_k \text{ and } \Psi_k + \frac{C_k - \Psi_k}{M} + \sum_{i=1}^{k-1} \frac{W_i(t)}{M} > t,
\]

which concludes the proof. \( \square \)

**Proof of Lemma 9.** This lemma can be proved in a similar manner as shown in our previous work [26]. We prove by contrapositive. Suppose that a job of task \( \tau_k, \tau_{k,j} \), misses its deadline. Let the arrival time of this job be \( a \) and the absolute deadline be \( a + D_k \).

We first construct a task set \( \tau' \) from \( \tau \), where the only difference between the two task sets is on \( \tau_k \). In \( \tau' \), we convert all suspensions of jobs released by \( \tau_k \) into computation. That is, we treat \( \tau_k \) as an ordinary sporadic task by factoring its suspension length into the worst-case execution time parameter. Thus, \( \tau_k \) executes just like an ordinary sporadic task (without suspensions) in the corresponding schedule, with an execution time of \( C_k + S_k \). Note that \( \tau_k \)'s computation (both the original computation and the computation converted from suspensions) will be preempted by higher-priority tasks. If \( \tau_{k,j} \) in the original task set \( \tau \) is not schedulable, then in the interval \( (a, a + D_k) \), the system can idle or execute tasks with lower priority than \( \tau_k \) by at most \( S_k \) amount of time; otherwise, job \( \tau_{k,j} \) has to suspend more than \( S_k \) amount of time in this interval. In the setting of \( \tau' \), we can consider the same pattern for the other jobs, but only convert the suspensions of task \( \tau_k \) in \( \tau \) to computation time. The additional \( S_k \) amount of computation time of \( \tau_{k,j} \) in \( \tau' \) can only be granted when the processor is idle or executes tasks with lower priority than \( \tau_k \), which is in total at most \( S_k \), as explained above. Therefore, \( \tau_k \) in \( \tau' \) is also not schedulable under global RM.

Within \( (a, a + t_o) \in (a, a + D_k) \), the work done by any high-priority task \( \tau_i (i < k) \) in the worst case can be divided into three parts: (i) body jobs: jobs of \( \tau_i \) with both release time and absolute deadline in \( (a, a + t_o) \), (ii) carry-in job: a job of \( \tau_i \) with release time earlier than \( a \) and absolute deadline in \( (a, a + t_o) \), and (iii) carry-out job: a job of \( \tau_i \) with release time in \( (a, a + t_o) \) and absolute deadline after \( a + t_o \). Since the carry-in and the carry-out job can each contribute at most \( C_i \) workload in \( [a, a + t_o] \), a safe upper bound of the interference due to task \( \tau_i \) in \( (a, a + t_o) \) is obtained by assuming that the carry-in and carry-out jobs of \( \tau_i \) both contribute \( C_i \) each in \( (a, a + t_o) \). Thus, the workload resulting from any higher-priority task \( \tau_i \) in \( (a, a + t_o) \) is at most \( W_i(t_o) \) (defined in Eq. (24)). Therefore, in order for \( \tau_{k,j} \) in \( \tau' \) to miss its deadline at \( a + D_k \), we know that

\[
\forall t \text{ with } 0 < t \leq T_k \text{ and } C_k + S_k + \sum_{i=1}^{k-1} \frac{W_i(t)}{M} > t,
\]

must hold, which concludes the proof. \( \square \)

**Appendix C: Experiments**

This section presents evaluation results by measuring the success ratio of the proposed tests with respect to a given goal of task set utilization. Due to space limitation, we only present the evaluations of utilization-based tests derived from our k^2U framework and the existing tests for multiframe systems, DAG systems, and implicit-deadline systems. For each specified set of utilization configuration, we generated 100 task sets. The success ratio of a configuration is the number of task sets that are schedulable under RM (or global RM in DAG systems) divided by the number of task sets for this configuration, i.e., 100.

We first generated a set of sporadic tasks, and then the corresponding tests were converted from this set according to different task models, e.g., multiframe and DAG tasks. The UUniFast method [5] was adopted to generate a set of utilization values with the given goal. We here used the approach suggested by Davis and Burns [15] to generate the task periods according to an exponential distribution. The order of magnitude \( p \) to control the period values between largest and smallest periods is parameterized in evaluations. (E.g., \( 1 - 10ms \) for \( p = 1, 1 - 100ms \) for \( p = 2, \text{ etc.} \)). The worst-case execution time was set accordingly, i.e., \( C_i = T_i U_i \) for multiframe systems and \( C_i = T_i U_i \) for DAG and uniprocessor implicit-deadline systems. Note that all the task systems are with implicit deadlines in our tests.

**Evaluations for Multiframe**

The multiframe tasks were then converted from the sporadic tasks as follows: The frame was generated in a similar manner to the method in [28]. The size of frame types \( m_i \) was randomly drawn from the interval \([2, 20]\). For each frame we randomly chose a scaling factor \( r_{i,j} \) in the range \((2, 5)\) to assign its execution time based on that of the first frame, i.e., \( C_{i,j} = C_{i,j_0}/r_{i,j} \). The cardinality of the task set was 10.

In this experiment, the proposed tests (the first three) and the existing tests are listed as follows:

- **Extreme Points Multiframe test (EPMF):** by using Lemma 4 in Theorem 3.
- **Hyperbolic Bound Multiframe (HPMF):** Eq. (31).
- **Total utilization Bound Multiframe (TBMF):** Eq. (32).
- **Mok:** Theorem 7 by Mok and Chen in [31].
- **Lu:** Theorem 3 by Lu et al. in [28].

![Fig. 2: Success ratio comparison in multiframe systems](image-url)
The cardinality of the task set was 50.

Our proposed tests are better than CAB, especially when the order of magnitude is greater. Our proposed tests are superior than the others for all different settings of $p$.

Figure 2 presents the result for the performance in terms of the success ratio. For all tests, the success ratio are slightly better when the order of magnitude is greater. Our proposed tests are superior than the others for all different settings of $p$.

Note that the experiment conducted in [28] applies the technique of task merging proposed in [19] as a preprocess and then tests the utilization bound. Apparently, the former can be also used in our proposed tests. However, we do not adopt this preprocess in our evaluations but focus on the effectiveness of utilization bounds themselves instead. The conclusion remains the same after adopting the preprocess on both sides.

**Evaluations for DAG Task Systems**

Similarly, the DAG tasks were converted from the sporadic tasks as follows: The critical-path length $\Psi_i$ of task $\tau_i$ was set by multiplying its WCET by uniform random values in the range $[0.75, 1]$. The following tests for global RM are evaluated:

- **Extreme Points test (EPDAG):** by using the following testing $\frac{\Psi_i + \Psi_k - \Psi}{t_k} \leq 1 - \sum_{i=1}^{k-1} \frac{U_i(\alpha_i + \beta_i)}{\prod_{j=i}^{k-1}(\beta_j U_j + 1)}$ derived from Lemma 4 where $\alpha_i$ is defined as $\frac{2}{M}$ and $\beta_i$ as $\frac{1}{M} / \frac{t_k}{T_k}$.  
- **Hyperbolic Bound DAG (HPDAG):** in Theorem 5.  
- **Chen and Agrawal Bound (CAB):** in Corollary 4 in [11].

CAB has the best known capacity augmentation bound for DAG systems under global RM.

The cardinality of the task set was 50.

Figure 3 depicts results with different numbers of processors, i.e., $M = 2, 4, 8$. For these algorithms, the success ratios are better when the number of processors is less. Apparently, our proposed tests are better than CAB, especially when the number of processors is large.

Fig. 3: Success ratio comparison in DAG systems where $U_{\Sigma}$ is the total utilization of a DAG task set.

**Evaluations for Implicit-Deadline Systems**

These tests in this experiment are as follows:

- **Extreme Points test (EP):** We use Lemma 4 in the analysis in Sec. 5.1.
- **Hyperbolic Bound (HP):** in Corollary 1.
- **Bini:** in Corollary 2 in [7].

The cardinality of the task set was 10.

Figure 4 depicts the results for 3 different orders of magnitude, i.e., $p = 1, 2, 3$. For there tests, the success ratio is better if the order of magnitude is greater. The EP dominates all the other tests for all different orders of magnitude. On the other hand, the results from Bini et al. in [7] and the proposed hyperbolic bound in Corollary 1 are comparable. The performance by the proposed hyperbolic bound is better than by Bini et al. in [7] for a smaller $p$ whereas Bini outperforms the proposed hyperbolic bound for a larger $p$.

**Appendix D: Additional Properties in $k^2U$**

We provide some additional properties that come directly from the $k^2U$ framework. These properties were not directly used in any of the demonstrated examples. The following lemma is useful when the index of the $k - 1$ higher priority tasks is not provided and cannot be determined while applying the schedulability tests. The results in Section 4 are highly rely on the given order of the $k - 1$ tasks. Therefore, without the given ordering, to be safe, we have to test all the permutations of the ordering of the $k - 1$ tasks. Fortunately, the following lemma, as an extension of Lemma 4, shows that testing only one particular ordering is enough to provide a safe schedulability test.

**Lemma 10.** Suppose that the given $k$-point effective schedulability test, defined in Eq. 4, of a fixed-priority scheduling algorithm does not have a predefined order to index the $k - 1$ higher-priority tasks. Task $\tau_k$ is schedulable by the scheduling algorithm if the following condition holds

$$0 < \frac{C_k}{t_k} \leq 1 - \sum_{i=1}^{k-1} \frac{U_i(\alpha + \beta)}{\prod_{j=i}^{k-1}(\beta_j U_j + 1)},$$

where $\alpha = \frac{2}{M}$ and $\beta = \frac{1}{M} / \frac{t_k}{T_k}$.
by indexing the $k-1$ higher-priority tasks in a non-decreasing order of $\frac{\alpha_i}{\beta_i}$, in which $0 < \alpha_i \neq \infty$ and $0 < \beta_i \neq \infty$ for any $i = 1, 2, \ldots, k - 1$, $0 < t_k \neq \infty$.

**Proof:** This lemma is proved by showing that the schedulability condition in Lemma 11, i.e., $1 - \sum_{i=1}^{k-1} \frac{U_i(\alpha_i + \beta_i)}{\prod_{j=i}^{k-1} (\beta_j U_j + 1)}$, is minimized, when the $k-1$ higher-priority tasks are indexed in a non-decreasing order of $\frac{\alpha_i}{\beta_i}$. Suppose that there are two adjacent tasks $\tau_\ell$ and $\tau_{\ell+1}$ with $\frac{\alpha_\ell}{\beta_\ell} > \frac{\alpha_{\ell+1}}{\beta_{\ell+1}}$. Let’s now examine the difference of $\sum_{i=1}^{k-1} \frac{U_i(\alpha_i + \beta_i)}{\prod_{j=i}^{k-1} (\beta_j U_j + 1)}$ by swapping the index of task $\tau_\ell$ and task $\tau_{\ell+1}$. It can be easily observed that the other tasks $\tau_i$ with $i \neq \ell$ and $i \neq \ell + 1$ do not change their corresponding values $\frac{U_i(\alpha_i + \beta_i)}{\prod_{j=i}^{k-1} (\beta_j U_j + 1)}$ in both orderings (before and after swapping $\tau_\ell$ and $\tau_{\ell+1}$). Suppose that $\prod_{j=i}^{k-1} (\beta_j U_j + 1)$ is $Q$, in which $Q > 0$. The difference in the term $\sum_{i=1}^{k-1} \frac{U_i(\alpha_i + \beta_i)}{\prod_{j=i}^{k-1} (\beta_j U_j + 1)}$ before and after swapping tasks $\tau_\ell$ and $\tau_{\ell+1}$ is

\[
((\alpha_\ell + \beta_\ell)U_\ell Q + (\alpha_{\ell+1} + \beta_{\ell+1})U_{\ell+1}Q(1 + \beta_\ell U_\ell))
- ((\alpha_{\ell+1} + \beta_{\ell+1})U_{\ell+1}Q + (\alpha_\ell + \beta_\ell)U_\ell Q(1 + \beta_{\ell+1}U_{\ell+1}))
= U_\ell U_{\ell+1}Q(\beta_\ell\alpha_{\ell+1} - \beta_{\ell+1}\alpha_\ell)
= \beta_\ell\beta_{\ell+1}U_\ell U_{\ell+1}Q\left(\frac{\alpha_{\ell+1}}{\beta_{\ell+1}} - \frac{\alpha_\ell}{\beta_\ell}\right) < 0.
\]

Therefore, we reach the conclusion by repetitively swapping the tasks to achieve a non-decreasing order of $\frac{\alpha_i}{\beta_i}$ for maximizing $\sum_{i=1}^{k-1} \frac{U_i(\alpha_i + \beta_i)}{\prod_{j=i}^{k-1} (\beta_j U_j + 1)}$. ■

The following lemma provides a tighter result than Lemma 11, by considering the two special cases $(\alpha+\beta)^{\frac{1}{k}} - 1 < 0$ and $(\alpha+\beta)^{\frac{1}{k}} - \alpha < 0$.

**Lemma 11.** For a given $k$-point effective schedulability test, defined in Eq. (53), of a scheduling algorithm, in which $0 < \alpha_i \leq \alpha \neq \infty$ and $0 < \beta_i \leq \beta \neq \infty$ for any $i = 1, 2, \ldots, k - 1$, $0 < t_k \neq \infty$, task $\tau_k$ is schedulable by the scheduling algorithm if

\[
\frac{C_k}{t_k} \sum_{i=1}^{k-1} U_i \leq \begin{cases} 
(\alpha+\beta)^{\frac{1}{k}} - \alpha & \text{if } (\alpha+\beta)^{\frac{1}{k}} - 1 < 0 \\
(\alpha+\beta)^{\frac{1}{k}} - \alpha & \text{if } (\alpha+\beta)^{\frac{1}{k}} - \alpha < 0 \\
\frac{(k-1)((\alpha+\beta)^{\frac{1}{k}} - 1)}{\beta} & \text{otherwise}.
\end{cases}
\]

**Proof:** The proof is essentially the same as that in the proof of Lemma 11. The right-hand side of Eq. (53) can be further improved by considering the cases $(\alpha+\beta)^{\frac{1}{k}} - 1 < 0$ 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 $\frac{U_1}{t_k}$ should be set to 0 in the extreme case by using KKT condition. This would require us to reformulate Eq. (53) by considering the two additional cases. ■