Schedulability and Optimization Analysis for Non-Preemptive Static Priority Scheduling Based on Task Utilization and Blocking Factors

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

***Abstract*—For real time task sets, allowing preemption is often considered to be important to ensure the schedulability, as it allows high-priority tasks to be allocated to the processor nearly immediately. However, preemptive scheduling also introduces some additional overhead and may not be allowed for some hardware components, which motivates the needs of non-preemptive or limited-preemptive scheduling. We present a safe sufficient schedulability test for non-preemptive (NP) fixed priority scheduling that can verify the schedulability for Deadline Monotonic (DM-NP) and Rate Monotonic (RM-NP) scheduling in linear time, if task orders according to priority and period are given. This test leads to a better upper bound on the speedup factor for DM-NP and RM-NP in comparison to Earliest Deadline First (EDF-NP) than previously known, closing the gab between lower and upper bound. We improve our test, resulting in interesting properties of the blocking time that allow to determine schedulability by only considering the schedulability of the preemptive case if some conditions are met. Furthermore, we present a utilization bound for RM-NP, based on the ratio $\gamma > 0$ of the upper bound of the maximum blocking time to the execution time, significantly improving previous results.**

1 Introduction

To model the recurrent executions of real-time applications, the sporadic task model has been widely adopted in the real-time systems domain [19]. The sporadic task model characterizes a task τ_i by its relative deadline D_i, its worst-case execution time (WCET) C_i, and its minimum inter-arrival time T_i. A sporadic task represents an infinite sequence of task instances, referred to as *jobs*, where the minimum inter-arrival time is the minimum time interval between the release of any two consecutive jobs of a task. The utilization U_i of task τ_i is defined as $\frac{C_i}{T_i}$.

For real-time computation systems the correct behavior does not only depend on the value of the computation, the correct value must also be produced within a certain amount of time. Therefore, the satisfaction of the deadlines has to be ensured for systems with hard real-time constraints. There have been several scheduling policies and their corresponding schedulability tests in the literature for such guarantees.

Allowing preemptions enables the scheduler to allocate the processor to high priority tasks nearly immediately to ensure that these important tasks meet their deadlines, while they may experience long blocking times from the execution of lower priority tasks for non-preemptive scheduling. Preemptive Earliest-Deadline-First (EDF-P) scheduling has been shown to be an optimal scheduling policy for dynamic priority scheduling of implicit-deadline task sets (i.e. $D_i = T_i$ for each task τ_i) with a utilization bound of $\sum_{\tau_i} U_i \leq 1$ [17] that will meet the deadlines for all schedulable task sets [11]. Moreover, for preemptive static priority scheduling, Deadline-Monotonic (DM-P) scheduling is optimal for constrained deadline task sets (i.e. $D_i \leq T_i$ for each task τ_i) [16] while the Rate Monotonic (RM-P) scheduling is optimal for implicit deadlines [17].

However, the correct calculation of the WCET that is needed for a good schedulability test is not easy if preemption is allowed, as preemption introduces additional overhead to the system, e.g. for suspending the task, inserting it into the ready queue, flushing the processor pipeline, and dispatching the new incoming task. Alternatively, we can adopt non-preemptive scheduling, in which the WCETs can be calculated much easier as no preemption overhead has to be taken into account. Unfortunately, the utilization bound for non-preemptive scheduling drops to 0, both for static and dynamic priority scheduling. This has motivated deferred preemption, preemption threshold, and limited preemptive scheduling, e.g., in [6], [23], [22].

Non-preemptive scheduling may also be enforced by the hardware. For example, messages in control area network (CAN) buses are not preemptable [1]. When considering non-preemptive scheduling, it may be necessary that the processor idles, even if there are jobs in the ready queue, to ensure the schedulability, e.g., when the computation time for some task is larger than another tasks deadline. If a task set is sporadic, no online algorithm can decide, whether the processor should idle or not [14]. It was shown that Non-Preemptive EDF (EDF-NP) is optimal among work-conserving non-preemptive schedulers [12], i.e., the processor is not allowed to go idle if at least one job is ready to be executed. However, when quan-

Task Set Constraints	Preemptive Lower Bound	Preemptive Upper Bound	Non-Preemptive Lower Bound	Non-Preemptive Upper Bound
Implicit Deadline	$1/ln(2) \approx 1.44269$ [17]		$1/\Omega \approx 1.76322$ [9]	2 [9] $1/\Omega$ [this paper]
Constrained Deadline	$1/\Omega \approx 1.76322$ [10]		$1/\Omega \approx 1.76322$ [9]	2 [9] $1/\Omega$ [this paper]
Arbitrary Deadline	$1/\Omega \approx 1.76322$ [8]	2 [8]	$1/\Omega \approx 1.76322$ [9]	2 [9]

TABLE I: Fixed Priority Scheduling Speedup Factors

tifying non-preemptive scheduling with utilization bounds, it can be easily shown that the utilization bound is 0 [20].

Although RM-P, DM-P, Non-Preemptive RM (RM-NP) and Non-Preemptive DM (DM-NP) are not optimal scheduling policies, it has been shown by Davis et al. [8][9][10] that these algorithms perform relatively well, compared to preemptive and non-preemptive EDF. The quantification is based on the speedup factor ρ, compared to EDF. That is, for any task set that is schedulable by preemptive (non-preemptive, respectively) EDF, it is also schedulable by the fixed-priority preemptive (non-preemptive, respectively) scheduling if speeding up the processor by factor ρ (i.e., the WCET of task τ_i becomes $\frac{C_i}{\rho}$). For constrained deadline task sets, an exact speedup factor of $\frac{1}{\Omega} \approx 1.76322$ was shown [10], where $\Omega \approx 0.56714$ is a constant defined by the transcendental equation $\Omega = ln\left(\frac{1}{\Omega}\right)$. For preemptive arbitrary deadline task sets a lower bound of $\frac{1}{\Omega} \approx 1.76322$ and an upper bound of 2 was shown for the speedup factor [8]. In the non-preemptive case they use EDF-NP for comparison, as it is optimal among work conserving scheduling algorithms (RM-NP and DM-NP vs. EDF-NP). They show the speedup factor is lower bounded by $\frac{1}{\Omega} \approx 1.76322$ and upper bounded by 2 for implicit, constrained and arbitrary deadline task sets [9]. Table I contains the previously known upper and lower bounds on this speedup factors for preemptive and non-preemptive fixed priority scheduling and the contributions of this paper. Analogously, analyzing the required speedup factor of non-preemptive scheduling, by comparing to the optimal preemptive scheduling, has also been proposed by Thekkilakattil et al. [21], in which the speedup factor cannot be bounded by a constant. Moreover, prior to this paper, the only existing utilization bound for non-preemptive fixed priority scheduling was provided by Andersson and Tovar [1]. They show that the utilization bound for RM-NP is $\frac{1}{1+\gamma}$ when $\gamma \geq 2$, where the *blocking factor* γ is defined as the ratio of (the upper bound of) the maximum blocking time divided by the execution time.

Our Contributions: This paper focuses on the schedulability and speedup factor analysis of non-preemptive fixed-priority scheduling for task sets with constrained deadlines:

- We provide a schedulability test in Section 4 for non-preemptive fixed-priority scheduling, that is in a hyperbolic form of the utilizations of the higher-priority tasks, by considering the execution time and the blocking time of the task that is being analyzed. This hyperbolic form is proven in Section 5 to reach the speedup factor $1/\Omega \approx 1.76322$ for DM-NP and RM-NP, by comparing to EDF-NP, as also shown in Table I. Therefore, this closes the gap between the upper bound and the lower bound for RM-NP and DM-NP.
- We also improve the schedulability test by considering two separated schedulability conditions more carefully in Section 6. This results in a better schedulability test that involves two polynomial-time analyses. These schedulability tests reveal some interesting properties on the blocking time. If the blocking time is less than a certain threshold, the schedulability of the task can be determined purely by considering the higher-priority tasks. This means that the blocking time has no drawback in the schedulability analysis.
- Moreover, our utilization bound (based on γ) for RM-NP in Theorems 8 and 9 covers any arbitrary setting of $\gamma > 0$, where γ is defined as the ratio of the maximum blocking time divided by the execution time. We show that the utilization bound of RM-NP can still be up to 69.3% (the same as RM-P) if $\gamma \leq \frac{1-ln(2)}{ln(2)} \approx 0.44269$. ~~if none of the lower priority tasks has larger execution time, i.e., $\gamma \leq 1$. As long as the lower-priority tasks do not have much longer execution time than the higher-priority tasks, e.g., $\gamma \leq 1.59$, the utilization bound can still be more than 50%.~~

2 System Model and Notations

This section presents the task model, scheduling model, and the notations used in this paper. We consider sporadic [19] and periodic [17] task sets with constrained or implicit deadlines on a single core processor.

2.1 Task and Scheduling Models

We assume a given task set $\tau = \{\tau_1, \ldots, \tau_n\}$ of n tasks. Each task releases an infinite number of jobs, where the j-th job of τ_i is denoted $\tau_{i,j}$. Each τ_i is specified by a 3-tuple of parameters (C_i, T_i, D_i). C_i is the worst case execution time (WCET) and D_i is the relative deadline of task τ_i. The period T_i is the minimum interarrival time between any two consecutive job releases of τ_i. Each job $\tau_{i,j}$ has a release time $r_{i,j}$, a finishing time $f_{i,j}$ and an absolute deadline $d_{i,j} = r_{i,j} + D_i$. The response time $R_{i,j} = f_{i,j} - r_{i,j}$ is the interval between arrival and finishing time for job $\tau_{i,j}$ and we denote the Worst Case Response Time of τ_i with R_i. The

processor utilization of τ_i is defined as $U_i = \frac{C_i}{T_i}$ and the total or system utilization as $U_{sum} = \sum_{i=1}^{n} U_i$

For each task τ_k we define $hp(\tau_k)$ as the set of tasks with higher priority and $lp(\tau_k)$ as the set of tasks with lower priority. We assume a total order on task priorities given by the fixed-priority scheduling policy, in which ties are broken arbitrarily. If not stated differently, the tasks in τ or any subsets of τ (especially in $hp(\tau_k)$ and $lp(\tau_k)$) are always ordered and indexed according to these priorities.

If $D_i \leq T_i$ holds for each $\tau_i \in \tau$, the task set is called with *constrained deadline*. If $D_i = T_i$ holds for each $\tau_i \in \tau$ then τ is an *implicit-deadline* set. Otherwise τ has *arbitrary deadlines*. For the rest of this paper, we only consider sporadic task systems with constrained or implicit deadlines.

We assume a uniprocessor and that the scheduler always dispatches the job in the ready queue that has the highest priority. This priority is given by the scheduling policy. We focus ourselves on fixed-priority scheduling by specifying a fixed priority level for a task. That means all the jobs of a task have the same fixed-priority level. For preemptive scheduling, it has been shown that RM and DM are the optimal fixed-priority scheduling policies for task sets with implicit deadlines or constrained deadlines, respectively. We further assume that tasks can not suspend themselves and tasks have no precedence constraints. Each task can be executed the moment it arrives and the arrivals of tasks are independent, and thus two or more tasks may be released at the same time.

As motivated in the introduction of this paper, we will consider non-preemptive fixed-priority scheduling in this paper. Specifically, a job cannot be preempted once its execution has been started. For the paper, when we use acronyms a -P will denote the preemptive case (e.g. RM-P for preemptive Rate Monotonic scheduling) while -NP will denote the non-preemptive case (e.g. RM-NP). The extension for limited preemptive scheduling will be discussed in Section 7.

A task τ_i is called schedulable by a given scheduling policy if it always meets its deadline, i.e. $R_i \leq D_i$. If this holds true for all $\tau_i \in \tau$ we call τ schedulable by the scheduling policy. A schedulability test is sufficient (with respect to the scheduling policy and the system model) if all task sets that are schedulable according to the test are schedulable. A schedulability test is called necessary, if a task set is unschedulable when the test is not passed. A test is called exact if the test is sufficient and necessary.

2.2 Speedup Factor

We define the speedup factors in relation to an optimal workload-conserving non-preemptive algorithm, i.e., EDF-NP [13], or an optimal preemptive scheduling algorithm, i.e., EDF-P [17]. Let $f^{opt}(\tau)$ be the processor speed an optimal algorithm needs to schedule τ, and $f^A(\tau)$ the speed necessary for scheduling algorithm A. We can now define a maximum speedup factor for any task set τ and a scheduling algorithm A as done in [10]

$$f^A = \max_{\forall \tau} \left\{ \frac{f^A(\tau)}{f^{opt}(\tau)} \right\} \tag{1}$$

As EDF-P is an optimal scheduling algorithm for preemptive scheduling, the speedup factor for other preemptive scheduling algorithms is determined in relation to EDF-P (e.g., fixed priority preemptive vs. EDF-P). For implicit deadline task sets that are schedulable by EDF-P, we get $f^{RM-P} = \frac{1}{\frac{UB(RM-P)}{UB(EDF-P)}} = \frac{1}{\frac{\ln(2)}{1}} = \frac{1}{\ln(2)} \approx 1.4427$ using the utilization bounds for EDF-P and RM-P [17]. The other cases for preemptive fixed-priority scheduling are also listed in Table I. For non-preemptive scheduling, we define the speedup factor by referring to the optimal workload-conserving non-preemptive scheduling, i.e. EDF-NP.

George et al. [13] presented an exact test for arbitrary deadline task sets to be schedulable under EDP-NP. They use the demand bound function [2][3]

$$h(t) = \sum_{i=1}^{n} max \left\{ 0, \left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right\} C_i \tag{2}$$

and the blocking time $B(t)$ to determine schedulability. We call $max \left\{ 0, \left\lfloor \frac{t-D_i}{T_i} \right\rfloor + 1 \right\} C_i = dbf_i(t)$ the demand bound function of task τ_i at time t. The maximum blocking time $B(t)$ for EDF-NP at time interval length t is defined as $\max_{\forall i, D_i > t}(C_i - \Delta)$ in [13]. As τ_i has to run when τ_k arrives to block it, τ_k can only be blocked by τ_i for $C_i - \Delta$, with $\Delta > 0$ but infinitesimally small. We assume that Δ is so small that it does not have any impact on the calculations, e.g., one processor cycle, and thus we can remove it from further equations for notation brevity.

The exact schedulability test of George et al. [13] tests if the following two conditions both hold to determine schedulability for the complete task set:

$$U_{sum} \leq 1 \tag{3a}$$

$$\frac{h(t) + B(t)}{t} \leq 1 \ \forall t \in S \tag{3b}$$

with $S = \{\cup_{i=1}^{n} \{kT_i + D_i\}, k \in \mathbb{N}\} \cap (0, L]$ where L is the longest Level-k Busy Interval. S is the union of the deadlines of all tasks in the interval $(0, L]$.

Therefore, if a non-preemptive fixed-priority scheduling algorithm A has a speedup factor f^A with respect to EDF-NP, we know that the unschedulability of algorithm A implies that either $U_{sum} > \frac{1}{f^A}$ or $\exists t \in S$ with $\frac{h(t) + B(t)}{t} > \frac{1}{f^A}$.

3 TDA Schedulability Test and Simplification

To analyse the schedulability of task sets under non-preemptive scheduling, the blocking time has to be taken into account. A task τ_k can only be blocked by tasks with lower priority, as the scheduler executes tasks with higher priority first anyway, and thus not adding additional time to the response time of τ_k. Since the scheduler always chooses the highest priority job in the ready queue for execution, a job of τ_k can only be blocked if at its arrival time a job $\tau_i \in lp(\tau_k)$ is executed and can be blocked at most once. Thus, for non-preemptive scheduling we can define the *strict upper bound* B_k of the maximum blocking time of a task τ_k as

$$B_k = \max_{\tau_i \in lp(\tau_k)}\{C_i\} > \max_{\tau_i \in lp(\tau_k)}\{C_i - \Delta\}. \quad (4)$$

The blocking factor γ of task τ_k is defined as B_k/C_k.

For preemptive fixed-priority scheduling of constrained-deadline task sets, the schedulability test can be done by using Time Demand Analysis (TDA) [15] as follows:
τ_k is schedulable if the following equation holds [15]:

$$\exists t \text{ with } 0 < t \leq D_k \text{ and } C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t \quad (5)$$

If this holds true for all $\tau_k \in \tau$ the task set is schedulable under the preemptive fixed-priority scheduling. For the non-preemptive case the maximum blocking time has to be added here, resulting in the following sufficient schedulability test for task τ_k [6], [9]:

$$\exists t \text{ with } 0 < t \leq D_k \text{ and } B_k + C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t \quad (6)$$

As B_k and C_k are both fixed, we can rewrite Eq. (6) by defining $\widehat{C_k} = C_k + B_k$, which results in

$$\exists t \text{ with } 0 < t \leq D_k \text{ and } \widehat{C_k} + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t \quad (7)$$

We further divide $hp(\tau_k)$ into two disjunct subsets with the following properties:

- $hp_1(\tau_k)$ consists of the $\tau_i \in hp(\tau_k)$ with $T_i < D_k$
- $hp_2(\tau_k)$ consists of the $\tau_i \in hp(\tau_k)$ with $T_i \geq D_k$

We *abuse* k by resetting it to $k = |hp_1(\tau_k)| + 1$ for brevity, where $|hp_1(\tau_k)|$ is the cardinality of $hp_1(\tau_k)$.[1] We know, that $\tau_i \in hp_2(\tau_k)$ interferes with τ_k at most once in the schedulability test in Eq. (7). Thus, we can bound the interference due to tasks $\tau_i \in hp_2(\tau_k)$ by summing up there execution times and adding them up with the tasks execution time and the blocking time: $\widehat{C'_k} = \widehat{C_k} + \sum_{\tau_i \in hp_2(\tau_k)} C_i = B_k + C_k + \sum_{\tau_i \in hp_2(\tau_k)} C_i$. Thus τ_k is schedulable if the following equation holds:

$$\exists t \text{ with } 0 < t \leq D_k \text{ and } \widehat{C'_k} + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t \quad (8)$$

TDA results in pseudo-polynomial runtime for the sufficient schedulability test by testing all time points with job arrivals from higher-priority tasks. We will use a more pessimistic test by testing only k time points $\{t_1, \ldots, t_{k-1}, t_k\}$, namely the last arrival points of higher priority tasks and the absolute deadline of τ_k:

$$t_i = \left\lfloor \frac{D_k}{T_i} \right\rfloor T_i \; \forall \; \tau_i \in hp_1(\tau_k) \text{ and } t_k = D_k \quad (9)$$

Thus, we get the following sufficient test for the schedulability of τ_k under the assumption that the schedulability of $\{\tau_1, \ldots, \tau_{k-1}\}$ has been ensured already:
$\exists t_j \in \{t_1, \ldots, t_k\}$ and

$$\widehat{C'_k} + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t_j}{T_i} \right\rceil C_i = \widehat{C'_k} + \sum_{i=1}^{k-1} \left\lceil \frac{t_j}{T_i} \right\rceil C_i \leq t_j \quad (10)$$

As we are only interested in the workload a higher priority task generates, and not in the concrete order they are executed in, we can reorder the tasks in $hp_1(\tau_k)$ according to their last release times, i.e., $t_1 \leq t_2 \leq \ldots \leq t_{k-1} \leq t_k$.

We know that $\left\lceil \frac{t_i}{T_i} \right\rceil$ is an integer for the last release time t_i (therefore we can remove the ceiling function) and that

$$\frac{t_i}{T_i} + 1 \geq \left\lceil \frac{t_j}{T_i} \right\rceil \; \forall \tau_i, \tau_j \in \{\tau_1, \ldots, \tau_k\} \quad (11)$$

as t_i is the last release time for a job of τ_i before D_k and $t_j < D_k \; \forall \tau_j \in \{\tau_1, \ldots, \tau_k\}$.

If we are looking at $t_j \in \{t_1, \ldots, t_k\}$, the last release of $\tau_i \in hp(\tau_k)$ only happened before t_j if $i < j$ after we reordered them. If $t_j > t_i$ we know that $\left\lceil \frac{t_j}{T_i} \right\rceil = \left\lceil \frac{t_i}{T_i} \right\rceil + 1 = \frac{t_i}{T_i} + 1$ where the inequality holds true due to the fact that t_i is the last release of τ_i before D_k. If $t_j \leq t_i$, we get $\left\lceil \frac{t_j}{T_i} \right\rceil \leq \left\lceil \frac{t_i}{T_i} \right\rceil = \frac{t_i}{T_i}$.

[1] Technically we would have to introduce another variable here, say k^*. As all tasks in $\tau_i \in hp_2(\tau_k)$ are summed up in $\widehat{C'_k}$ and we only have to consider the $\tau_i \in hp_1(\tau_k)$, such a new notation has no additional value for the analysis but makes the paper more difficult to read.

Thus, for each time t_j we can split the summation in Eq. (10) into two parts, where the first summation represents all jobs of higher priority tasks but the last one, and the second summation represents the last job for the task where this last job is already released at t_j:

$$\widehat{C'_k} \;+\; \sum_{i=1}^{k-1} \left\lceil \frac{t_j}{T_i} \right\rceil C_i \;\leq\; \widehat{C'_k} \;+\; \sum_{i=1}^{k-1} \frac{t_i}{T_i} C_i \;+\; \sum_{i=1}^{j-1} C_i \quad (12)$$

We unify these considerations in a safe sufficient schedulability test for a task τ_k stated in the the following lemma:

Lemma 1. *If the schedulability of all higher priority tasks is ensured already, task τ_k is schedulable by a non-preemptive static priority scheduling policy if $\exists t_j \in \{t_1, \ldots, t_k\}$ such that*

$$\widehat{C'_k} \;+\; \sum_{i=1}^{k-1} \frac{t_i}{T_i} C_i \;+\; \sum_{i=1}^{j-1} C_i \;\leq\; t_j \quad (13)$$

Proof: This follows directly from the argumentation in this section. ∎

4 Polynomial-Time Schedulability Tests

To get a schedulability test in a hyperbolic form we need a schedulability test based on the utilization of the higher priority tasks and the execution and blocking time of the task currently tested. The left summation of Eq. (13) in Lemma 1 can easily be converted to be utilization-based as $U_i = \frac{C_i}{T_i}$. To get the right summation utilization-based, we have to do some simple transformations:

$$\sum_{i=1}^{j-1} C_i \;=\; \sum_{i=1}^{j-1} \frac{T_i}{T_i} C_i \;=\; \sum_{i=1}^{j-1} T_i U_i \;\leq\; \sum_{i=1}^{j-1} t_i U_i \quad (14)$$

where the inequality holds true due to the fact that $t_i = f_i * T_i$ for some $f_i \in \mathbb{Z}^+$. This results in the following more pessimistic utilization-based schedulability test

$$\exists t_j \in \{t_1, \ldots, t_k\} \text{ and } \widehat{C'_k} + \sum_{i=1}^{k-1} t_i U_i + \sum_{i=1}^{j-1} t_i U_i \leq t_j \quad (15)$$

For non-preemptive scheduling, it is important to verify the schedulability for each task individually, as the utilization of the task set is not a monotonically increasing function, because the blocking time has to be considered individually for each task and is a monotonically decreasing function.

We will use Eq. (15) to show the following theorem, that allows a schedulability test in a hyperbolic form for fixed priority non-preemptive scheduling. The following theorem can also be easily proved by using the k^2u framework presented in [7] with $\alpha = \alpha_i = 1$ and $\beta = \beta_i = 1$ in the setting in [7].

Theorem 1. *A task τ_k in a non-preemptive sporadic task system with constrained deadlines can be feasibly scheduled by a fixed-priority scheduling algorithm, if the schedulability for all higher priority tasks has already been ensured and the following condition holds:*

$$\left(\frac{\widehat{C'_k}}{D_k} \;+\; 1 \right) \prod_{\tau_j \in hp_1(\tau_k)} (U_j \;+\; 1) \;\leq\; 2 \quad (16)$$

Proof: We will prove the theorem by showing that if the condition in Eq. (16) is satisfied, the condition in Eq. (15) will be satisfied as well by using contrapositive, thus showing that if Eq. (15) is not satisfied, Eq. (16) will not be satisfied as well. The proof uses the same strategy as the proof of Lemma 1 in [7]. For completeness, we will list the corresponding linear programming and the optimal extreme point solution. The proof for the optimality of the extreme point solution is in the Appendix.

If a task τ_k is not schedulable, by Eq. (8) we know that

$$\forall t \text{ with } 0 < t \leq D_k : \; \widehat{C'_k} + \sum_{\tau_i \in hp_1(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i > t$$

This must hold true $\forall\; t \;\in\; (0,\; D_k]$. Therefore, it must hold true for the t of interest, particularly for the times of the last releases of higher priority tasks. All transformations we made to get Eq. (15) from Eq. (8) only increased the left side of the equations, thus an unschedulable task τ_k will fail Eq. (15) as well. With this we know, that if τ_k is not schedulable

$$\forall\; j \;\in \{1,\; \ldots,\; k-1,\; k\} : \widehat{C'_k} \;+\; \sum_{i=1}^{k-1} t_i U_i \;+\; \sum_{i=1}^{j-1} t_i U_i \;>\; t_j \quad (17)$$

Note, that a task set might still be schedulable if Eq. (17) holds, as Eq. (15) is only a sufficient scheduling condition. But we know, that if a task is not schedulable Eq. (17) will hold and prove, that Eq. (16) will not hold if Eq. (17) holds. Thus Eq. (16) will not hold for any unschedulable task τ_k. This results in the following optimization problem represented by a linear programming:

$$\inf\; C_k^* \quad (18a)$$

$$\text{s.t } C_k^* + \sum_{i=1}^{k-1} t_i^* U_i + \sum_{i=1}^{j-1} t_i^* U_i > t_j^*, \qquad \forall\; 1 \leq j \leq k \quad (18b)$$

$$t_j^* \;\geq\; 0, \qquad \forall\; 1 \leq j \leq k \quad (18c)$$

where $t_1^*,\; \ldots,\; t_{k-1}^*$ and C_k^* are variables and t_k^* is defined as t_k for notational brevity. We replace $>$ with $\geq$ in (18b) as infimum and minimum are the same if $\geq$ is used. Thus Eq. (17) will hold $\forall\; C_k \;>\; C_k^*$

We get $C_k^* \;\geq\; t_k^* \;-\; \left(\sum_{i=1}^{k-1} t_i^* U_i \;+\; \sum_{i=1}^{k-1} t_i^* U_i \right)$ from Eq. (18b) if $j \;=\; k$. We can use this inequality to replace C_k^*

in Eq. (18a) and Eq. (18b). As for t_k^* the two summations are the same we get $t_k^* - 2\sum_{i=1}^{k-1} t_i^* U_i$ in Eq. (18a), and thus to find a minimum value for C_k^* we have to maximize $\sum_{i=1}^{k-1} t_i^* U_i$ as t_k^* is a constant. When we replace C_k^* in Eq. (18b), we get

$$\begin{aligned} & t_k^* - 2\sum_{i=1}^{k-1} t_i^* U_i + \sum_{i=1}^{k-1} t_i^* U_i + \sum_{i=1}^{j-1} t_i^* U_i \\ & = t_k^* - \sum_{i=j}^{k-1} t_i^* U_i \geq t_j^*, \ \forall\ 1 \leq j \leq k-1 \end{aligned} \tag{19}$$

These reformulations result in the following linear programming:

$$\max \sum_{i=1}^{k-1} t_i^* U_i \tag{20a}$$

$$\text{s.t } t_k^* - \sum_{i=j}^{k-1} t_i^* U_i \geq t_j^* \qquad \forall\ 1 \leq j \leq k-1 \tag{20b}$$

$$t_j^* \geq 0, \qquad \forall\ 1 \leq j \leq k-1 \tag{20c}$$

We know that the objective function in Eq. (20a) is bounded as $0 \leq t_i^* \leq t_k^* < \infty$, $\forall\ 1 \leq j \leq k-1$. Thus, the $2(k-1)$ constraints in Eq. (20b) and Eq. (20c) form a polyhedron of feasible solutions as stated in the extreme point theorem for linear programming [18]. This polyhedron is either empty, thus the optimization problem has no feasible solution, or one of the extreme points of the polyhedron is an optimal solution for the optimization problem due to the extreme point theorem [18]. As there are $k-1$ variables, at least $k-1$ of the constraints in Eq. (20b) and Eq. (20c) have to be active, i.e. $\geq$ holds with $=$ in the solution.

We can get one extreme point solution with $t_j^* > 0$, $\forall\ 1 \leq j \leq k-1$ by setting $t_j^* = t_k^* - \sum_{i=j}^{k-1} t_i^* U_i$ with

$$t_{i+1}^* - t_i^* = t_i^* U_i, \ \forall\ 1 \leq j \leq k-1 \tag{21}$$

Thus we know

$$\frac{t_{i+1}^*}{t_i^*} = U_i + 1, \ \forall\ 1 \leq j \leq k-1 \tag{22}$$

and

$$\frac{t_i^*}{t_k^*} = \prod_{j=1}^{k-1} \frac{t_j^*}{t_{j+1}^*} = \frac{1}{\prod_{j=i}^{k-1}(U_j + 1)} \tag{23}$$

From Eq. (18b) with $j = k$ we know that the minimum value of C_k^* is:

$$\begin{aligned} C_k^* &= t_k^* - 2\sum_{i=1}^{k-1} t_i^* U_i \overset{(21)}{=} t_k^* - 2(t_k^* - t_1^*) \\ &\overset{(23)}{=} t_k^* - 2\left(t_k^* - \frac{t_k^*}{\prod_{j=1}^{k-1}(U_j + 1)}\right) \\ &\Rightarrow C_k^* = t_k^*\left(\frac{2}{\prod_{j=1}^{k-1}(U_j + 1)} - 1\right) \end{aligned} \tag{24}$$

The further proof in the Appendix shows that all other possible solutions have a worse objective value than the one we constructed and that is represented in Eq. (24). Thus we conclude that Eq. (17) always holds if $\widehat{C_k'} > C_k^*$. We get

$$\left(\frac{C_k^*}{t_k^*} + 1\right) \prod_{j=1}^{k-1}(U_j + 1) > 2 \tag{25}$$

Thus we know, if

$$\left(\frac{C_k^*}{t_k^*} + 1\right) \prod_{j=1}^{k-1}(U_j + 1) \leq 2 \tag{26}$$

holds, τ_k is schedulable if all higher priority tasks are schedulable. We know that $t_k^* = D_k$. We can replace C_k^* with $\widehat{C_k'}$, as we constructed it to be the minimum of the values Eq. (17) holds for, thus reaching the conclusion of Theorem 1. ■

Instead of optimizing for the smallest C_k^* to ensure Eq. (17) holds, we can also minimize $\widehat{C_k'} + \sum_{i=1}^{k-1} t_i U_i$ to ensure Eq. (17) holds. This leads to another sufficient schedulability test.

Theorem 2. *A task τ_k in a non-preemptive sporadic task system with constrained deadlines can be feasibly scheduled by a fixed-priority scheduling algorithm, if the schedulability for all higher priority tasks has already been ensured and the following condition holds:*

$$\frac{\widehat{C_k'} + \sum_{i=1}^{k-1} t_i U_i}{D_k} \leq \frac{1}{\prod_{\tau_j \in hp_1(\tau_k)}(U_i + 1)} \tag{27}$$

The proof of Theorem 2 is very similar to the proof of Theorem 1 and can be found in the Appendix.

Observation 1. *The schedulability tests in Theorem 1 and Theorem 2 provide the same result.*

The observation above is due to the fact that the optimization problem for both approaches lead to the same linear programming, thus providing the same solution. We use Eq. (18b) with $j = k$ to show the property in Theorem 1, and Eq. (45b) with $j = k$ to show the property in Theorem 2,

and these two equations are the same. As all following steps are without estimations, both will hold if $\widehat{C'_k} \leq C^*_k$ and both will fail for $\widehat{C'_k} > C^*_k$.

The schedulability test in Theorem 1 (as well as the others in Section 4 and Section 6 with a similar treatment) can be implemented to test the schedulability for all the n tasks in the given task set τ under RM-NP and DM-NP in linear time, provided that the orders by their periods and their relative deadlines are given. This is due to the following considerations. The blocking time for all tasks can be computed in $O(n)$ if we compute it starting with the lowest priority task and save the values in an array. For DM-NP, if we move from τ_k to τ_{k+1} we want to analyze the changes in hp_1 and hp_2 for this step. As the relative deadline is increasing with the task priority, no task can ever move from hp_1 to hp_2. Assume τ_k is placed in hp_2. Then all tasks $\tau_i \in hp_2$ with $D_k \leq T_i < D_{k+1}$ will be moved to hp_1.This can be determined in $O(1)$ for each task. If the task in hp_2 are tested in increasing order of their period, we can stop for this step once $T_i < D_{k+1}$ does not hold. Each task is only moved from hp_2 to hp_1 at most once, due to the monotonicity of the deadlines in DM, and for each move $\prod_{\tau_j \in hp_1} (U_j + 1)$ can be computed in $O(1)$. Therefore, the test in Theorem 1 has an amortized cost $O(1)$ resulting in $O(n)$ for τ. The two required orders can be computed in $O(n \log n)$ if not given. Note, that for RM-NP hp_2 is always empty. For the general case this argumentation does not hold, as tasks may be moved from hp_1 to hp_2 as well. In this case the time complexity can be $O(n)$ for each step, resulting in a total complexity of $O(n^2)$.

We conclude this section with the following theorems:

Theorem 3. *Suppose that the tasks are indexed such that $T_i \leq T_{i+1}$. If $\gamma = \max_{\tau_i \in lp(\tau_k)} \left\{ \frac{C_i}{C_k} \right\} = \frac{B_k}{C_k}$, then task τ_k is schedulable by RM-NP if*

$$U_{sum} \leq \begin{cases} \left(\left(\frac{2}{1+\gamma} \right)^{\frac{1}{k}} - \frac{1}{1+\gamma} \right) \\ \quad + (k-1) \left(\left(\frac{2}{1+\gamma} \right)^{\frac{1}{k}} - 1 \right) & \text{if } \gamma \leq 1 \\ \frac{1}{1+\gamma} & \text{if } \gamma > 1 \end{cases} \quad (28)$$

Proof: This is proved based on a similar proof of the Liu and Layland bound by using Lagrange Multiplier Method. The details are in the Appendix. ∎

Theorem 4. *Suppose that $\gamma = \max_{\tau_k} \left\{ \max_{\tau_i \in lp(\tau_k)} \left\{ \frac{C_i}{C_k} \right\} \right\}$. A task set can be feasibly scheduled by RM-NP if*

$$U_{sum} \leq \begin{cases} \frac{\gamma}{1+\gamma} + ln\left(\frac{2}{1+\gamma} \right) & \text{if } \gamma \leq 1 \\ \frac{1}{1+\gamma} & \text{if } \gamma > 1 \end{cases} \quad (29)$$

Proof: This follows directly from Theorem 3 by calculating the utilization bound when $k \to \infty$, i.e.,

$$\lim_{k \to \infty} \left(\left(\frac{2}{1+\gamma} \right)^{\frac{1}{k}} - \frac{1}{1+\gamma} \right) + (k-1) \left(\left(\frac{2}{1+\gamma} \right)^{\frac{1}{k}} - 1 \right)$$
$$= k \left(\left(\frac{2}{1+\gamma} \right)^{\frac{1}{k}} - 1 \right) + \left(1 - \frac{1}{1+\gamma} \right) = ln\left(\frac{2}{1+\gamma} \right) + \frac{\gamma}{\gamma+1}$$

for the cases when $\gamma \leq 1$. For $\gamma > 1$ the result is identical to Theorem 3 regardless of k. ∎

The result in Theorem 4 in fact significantly improves the utilization bounds for RM-NP. Prior to this paper, the only existing result was provided by Andersson and Tovar [1]. They show that the utilization bound for non-preemptive RM is $\frac{1}{1+\gamma}$ when $\gamma \geq 2$. They conclude in [1] that the utilization bound of RM-NP for control area network (CAN) 2.0A is 25.8% due to $\gamma \leq \frac{135}{47}$ and for CAN 2.0B is 29.5% due to $\gamma \leq \frac{160}{67}$. The analysis in [1] was too pessimistic, as their utilization bound was only for the extreme cases when $\gamma \geq 2$. With the analysis in Theorem 4, we can conclude that the utilization bound of RM-NP for CAN bus utilization can still be up to 50% if all the tasks have the same execution time, i.e., $\gamma = 1$, meaning that all the messages have the same length. We will further improve the test by using tighter schedulability analysis in Section 6.

5 Speedup Factor for DM-NP

Using the hyperbolic tests in Section 4, we present the speedup factor of DM-NP for constrained-deadline systems with respect to EDF-NP.

Theorem 5. *The speedup factor of non-preemptive deadline monotonic scheduling for task sets with constrained deadline is $\frac{1}{\Omega} \approx 1.76322$ with respect to non-preemptive earliest deadline first scheduling.*

Proof: The lower bound of $\frac{1}{\Omega} \approx 1.76322$ for the speed-up factor was provided by Davis et. al [9]. They construct an example that shows $\frac{1}{\Omega} \approx 1.76322$ is nearly reached for some task sets. This means we only have to show that the upper bound is $\frac{1}{\Omega} \approx 1.76322$ as well, to conclude the proof. This can be done by showing, that all task sets accepted by the exact schedulability test for EDF-NP on a processor with speed 1 will be accepted for DM-NP on a processor with speed $\frac{1}{\Omega} \approx 1.76322$ as well. We will prove this using contrapositive, showing that if a task τ_k is not accepted by our schedulability test in Theorem 1, it will also not be accepted by the exact schedulability test for EDF-NP on a processor with speed Ω.

If $\prod_{i=1}^{k-1}(U_i + 1) \geq 2$ we know that $\sum_{i=1}^{k-1} U_i \geq \ln 2$, resulting in the speed-up factor of $\frac{1}{\ln 2} < 1.76322$ directly. In the second case we have $\prod_{i=1}^{k-1}(U_i + 1) < 2$, which implies $\sum_{i=1}^{k-1} U_i < 1$, and $C^*_k \geq 0$.

We know from the proof of Theorem 1 that we can construct the minimum value C_k^* to ensure the schedulability test fails for a task τ_k by solving the linear programming in Eq. (18). From Eq. (16) we know that for the extreme case

$$\prod_{i=1}^{k-1}(U_i \ + \ 1) \ = \ \frac{2}{\left(\frac{C_k^*}{D_k} \ + \ 1\right)} \qquad (30)$$

By the definition of $B(t)$ for EDF-NP (Section 2.2), we know that $B(D_k) = \max_{\forall i, D_i > t}\{C_i\} = B_k$ when DM-NP is used. If a task is not accepted by our schedulability test in Theorem 1 we know, that

$$\begin{aligned}
\frac{h_k(D_k) + B_k(D_k)}{D_k} &= \frac{B_k + \sum_{i=1}^{k} max\left\{0, \left\lfloor \frac{t-D_i}{T_i} \right\rfloor + 1\right\} C_i}{D_k} \\
&= \frac{B_k + C_k + \sum\limits_{\tau_i \in hp_2(\tau_k)} C_i + \sum\limits_{\tau_i \in hp_1(\tau_k)} max\left\{0, \left\lfloor \frac{t-D_i}{T_i} \right\rfloor + 1\right\} C_i}{D_k} \\
&\geq \frac{\widehat{C_k'} + \sum_{i=1}^{k-1} max\left\{0, \left\lfloor \frac{t-T_i}{T_i} \right\rfloor + 1\right\} C_i}{D_k} \geq \frac{\widehat{C_k'} + \sum_{i=1}^{k-1} t_i U_i}{D_k} \\
&\overset{1}{>} \frac{C_k^* + \sum_{i=1}^{k-1} t_i U_i}{t_k} = \frac{1}{\prod_{i=1}^{k-1}(U_i \ + \ 1)} \ = \ \frac{1 + \frac{C_k^*}{D_k}}{2}
\end{aligned}$$

where $\overset{1}{>}$ comes from Theorem 2 and Observation 1. We denote $\frac{C_k^*}{D_k}$ as x from now on, thus $\frac{1+\frac{C_k^*}{D_k}}{2} \ = \ \frac{1+x}{2}$.

We are looking for the infimum utilization $\sum_{i=1}^{k-1} U_i$ to ensure $\prod\limits_{i=1}^{k-1}(U_i+1) > \frac{2}{x+1}$. Due to the fact that the arithmetic mean is always larger than or equal to the geometric mean, we know that $\left(\frac{\sum_{i=1}^{k-1} U_i}{k-1} + 1\right)^{k-1} \geq \prod\limits_{i=1}^{k-1}(U_i+1)$. Moreover, we have the fact $\left(\frac{\sum_{i=1}^{k-1} U_i}{k-1} + 1\right)^{k-1} \leq e^{\sum_{i=1}^{k-1} U_i}$, where the right-hand side is the case when k goes to ∞ and e is the Euler number. From $\prod\limits_{i=1}^{k-1}(U_i+1) > \frac{2}{x+1}$, we reach

$$\sum_{i=1}^{k-1} U_i \ > \ ln\left(\frac{2}{1+x}\right). \qquad (31)$$

As $ln\left(\frac{2}{1+x}\right)$ is a decreasing function of x, while $\frac{1+x}{2}$ is an increasing function of x, we know that

$$\inf_{0 \leq x < 1} \left\{ \max\left\{ \frac{x+1}{2}, \ ln\left(\frac{2}{1+x}\right) \right\} \right\} = \Omega \qquad (32)$$

which happens at the intersection of these two functions, i.e., $\frac{2}{x \ + \ 1} = ln\left(\frac{2}{1+x}\right)$. As a result, we conclude our proof by

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

■

The following corollary is a straightforward extension of Theorem 5.

Corollary 1. *The speedup factor of non-preemptive rate monotonic scheduling for task sets with implicit deadline is 1.76322 with respect to non-preemptive earliest deadline first.*

6 Tighter Hyperbolic Schedulability Test

The sufficient schedulability test in Lemma 1 is pessimistic, as the concept behind it still allows task τ_k to be preempted. If we consider that task τ_k cannot be preempted as long as it starts, we merely have to verify whether a job of task τ_k, arriving at time t, can be started before $t+D_k-C_k$. Therefore, we can determine the schedulability of τ_k by[2]

$$\exists t \ \in \ (0, \ D_k - C_k] \text{ with } B_k \ + \ \sum_{i=1}^{k-1} \left\lceil \frac{t}{T_i} \right\rceil C_i \ \leq \ t \qquad (34)$$

thus ensuring that there is enough time for τ_k to start executing.

However, testing Eq. (34) *alone is not safe enough, as the worst case response time of a task may happen to a later job, due to the self-pushing phenomenon [5].* Fortunately, by adopting the following lemma from Yao, Buttazzo, and Bertogna [23], we can still use the test in Eq. (34) under certain conditions.

Lemma 2 (Yao, Buttazzo, and Bertogna, 2010)**.** *The worst-case response time of a non-preemptive task occurs in the first job if the task is activated at its critical instant and the following two conditions are both satisfied:*

1) the task set is feasible under preemptive scheduling;

2) the relative deadlines are less than or equal to periods.

[2]In the literature, e.g., [23], [5], when considering a tight blocking time with $B_k = \max_{\tau_i \ \in \ lp(\tau_k)}\{C_i - \Delta\}$ (instead of a strict upper bound) they have to use $\left\lfloor \frac{t}{T_i} \right\rfloor + 1$ (instead of $\left\lceil \frac{t}{T_i} \right\rceil$) in Eq. (34). The simplication by setting B_k to $\max_{\tau_i \ \in \ lp(\tau_k)}\{C_i\}$ instead of $\max_{\tau_i \ \in \ lp(\tau_k)}\{C_i - \Delta\}$ with $\Delta > 0$ allows us to put $\leq$ instead of $<$ in the condition.

Therefore, we can combine Lemma 2 and Eq. (34), which results in the following Lemma:

Lemma 3. *A task τ_k is schedulable by a fixed priority non-preemptive scheduling (FP-NP) algorithm A^{NP}, if all higher priority tasks are schedulable, and the following two conditions hold:*

1) *the first job of τ_k will be executed before its deadline: $\exists t \in (0,\ D_k - C_k]$ with $B_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t$.*
2) *the task set is schedulable by A^P (FP-P): $\exists t \in (0,\ D_k]$ with $C_k + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t$.*

Proof: This follows directly from the previous considerations and Lemma 2. ∎

From Lemma 3, we are going to construct a tighter sufficient schedulability test based on two hyperbolic equations. This construction will be similar to the one in Theorem 1, thus we will only state the differences here.

As we are considering different time intervals $(0,\ D_k]$ and $(0,\ D_k - C_k]$ for the preemptive and the non-preemptive test, the sets $hp_1(\tau_k)$ and $hp_2(\tau_k)$ will not necessarily be identical for both tests, i.e., a task $\tau_i \in hp(\tau_k)$ with $D_k - C_k \leq T_i < D_k$ will be in $hp_1(\tau_k)$ for the preemptive case and in $hp_2(\tau_k)$ for the non-preemptive case. Thus we denote these sets $hp_1^P(\tau_k)$ and $hp_2^P(\tau_k)$ for the preemptive case and $hp_1^{NP}(\tau_k)$ and $hp_2^{NP}(\tau_k)$ for the non-preemptive case. As the examined deadline differs, the order of the jobs in $hp_1^P(\tau_k)$ and $hp_1^{NP}(\tau_k)$ may differ as well if the last release time t_i of τ_i is in $(D_k - C_k,\ D_k]$, and thus the permutation of the $\tau_i \in hp_1^P(\tau_k)$ according to the last release of τ_i may differ from the permutation of the $\tau_i \in hp_1^{NP}(\tau_k)$ as well. We denote those last release times t_i^P and t_i^{NP} and the resulting permutations as π_k^P and π_k^{NP}.

Theorem 6. *A task τ_k is schedulable by a fixed priority non-preemptive scheduling algorithm A^{NP} if all higher priority tasks are schedulable and the following two conditions hold:*

$$\left(\frac{B_k + \sum_{\tau_i \in hp_2^{NP}(\tau_k)} C_i}{D_k - C_k} + 1 \right) \prod_{\tau_j \in hp_1^{NP}(\tau_k)} (U_j + 1) \leq 2 \quad (35)$$

$$\left(\frac{C_k + \sum_{\tau_i \in hp_2^{P}(\tau_k)} C_i}{D_k} + 1 \right) \prod_{\tau_j \in hp_1^{P}(\tau_k)} (U_j + 1) \leq 2 \quad (36)$$

Proof: We need to show that if both conditions in this theorem hold, Lemma 3 holds as well. The proof for Eq. (36) is similar to the one of Theorem 1. Instead of looking for the smallest $\widehat{C_k'}$, we are looking for the smallest C_k.

For the proof of Eq. (35) the sets $hp_1^{NP}(\tau_k)$ and $hp_2^{NP}(\tau_k)$ may differ from $hp_1^P(\tau_k)$ and $hp_2^P(\tau_k)$, i.e., a tasks τ_i with $D_k - C_k \leq T_i < D_k$ is moved from $hp_1^P(\tau_k)$ to $hp_2^{NP}(\tau_k)$, and thus $|hp_1^{NP}(\tau_k)| \leq |hp_1^P(\tau_k)|$. The tasks in $hp_1^{NP}(\tau_k)$ will be ordered according to

$$t_i^{NP} = \left\lfloor \frac{D_k - C_k}{T_i} \right\rfloor T_i \ \forall \tau_i \in hp_1^{NP}(\tau_k) \text{ and } t_k = D_k \quad (37)$$

We create an optimization problem, again looking for the smallest $B_k^* > B_k' = B_k + \sum_{\tau_i \in hp_1^{NP}(\tau_k)} C_i$ to ensure that τ_k can not start. The conditions for the optimization problem are the same as in Eq. (18b) with B_k^* instead of C_k^*, if we replace $>$ with $\geq$ to look for the minimum instead of the infimum. With $B_k^* \geq t_k^* - \sum_{i=1}^{k-1} t_i^* U_i - \sum_{i=1}^{k-1} t_i^* U_i$ we get a maximization problem of the $\sum_{i=1}^{k-1} t_i^* U_i$ with the same constraints as in Eq. (18), and thus getting the same solution for B_k^*. ∎

The following theorem provides an interesting property of the blocking time. If the blocking time is not too long under certain conditions, the blocking time has no impact on the schedulability test in Lemma 3.

Theorem 7. *The schedulability of task τ_k under a non-preemptive fixed priority scheduling A^{NP} solely depends on the schedulability of τ_k under its preemptive version A^P if the following two conditions both hold:*

$$T_i \leq D_k - C_k, \qquad \forall \tau_i \in hp(\tau_k) \quad (38a)$$

$$B_k \leq \left(1 - \frac{C_k}{D_k}\right) C_k \quad (38b)$$

The proof of Theorem 7 is in the Appendix.

We conclude this section with the following theorems for the total utilization bounds of RM-NP with respect to γ. These bounds are tighter than the results in Theorems 3 and 4.

Theorem 8. *Suppose that the tasks are indexed such that $T_i \leq T_{i+1}$. If $\gamma = \max_{\tau_i \in lp(\tau_k)} \left\{ \frac{C_i}{C_k} \right\} = \frac{B_k}{C_k} > 0$, then task τ_k is schedulable by RM-NP if*

~~$$\sum_{i=1}^{k} U_i \leq \min\{k(2^{\frac{1}{k}} - 1), H(k,\gamma)\}$$~~

$$\sum_{i=1}^{k} U_i \leq \min\left\{ k(2^{\frac{1}{k}} - 1), \frac{1}{1+\gamma} \right\} \quad (39)$$

~~where~~

$$H(k,\gamma) = \begin{cases} (k-1)(2^{\frac{1}{k-1}}-1) & \text{if } \gamma \le (\frac{1}{2})^{\frac{1}{k-1}} \\ \frac{(\frac{2}{\gamma})^{\frac{1}{k}}-\frac{1}{\gamma}}{1+(\frac{2}{\gamma})^{\frac{1}{k}}-\frac{1}{\gamma}} + (k-1)((\frac{2}{\gamma})^{\frac{1}{k}}-1) & \text{if } (\frac{1}{2})^{\frac{1}{k-1}} < \gamma \le 2 \\ \frac{1}{1+\gamma} & \text{if } \gamma > 2 \end{cases} \quad (40)$$

Proof: The utilization bound of Eq. (36) for RM-NP is the well-known Liu and Layland bound $k(2^{\frac{1}{k}}-1)$, as shown in [17], [4]. We only have to focus on the utilization bound of Eq. (35). ~~which is denoted by $H(k,\gamma)$.~~ Due to RM-NP, we know $T_i \le T_k$ for any higher-priority task τ_i, which means $\frac{C_i}{T_k} \le U_i$. Therefore, a more pessimistic test than Eq. (35) is to test whether

$$\left(\frac{\gamma U_k + \sum_{\tau_i \in hp_2^{NP}(\tau_k)} U_i}{1-U_k} + 1 \right) \prod_{\tau_j \in hp_1^{NP}(\tau_k)} (U_j+1) \le 2 \quad (41)$$

The utilization bound can be proven by finding the infimum $\sum_{i=1}^{k} U_i$ such that Eq. (41) does not hold. We first show that the condition in Eq. (41) can be simplified. Suppose that $U_k + \sum_{\tau_i \in hp_2^{NP}(\tau_k)} U_i$ is specified, denoted as f. It is not difficult to see that $\frac{\gamma U_k + f - U_k}{1-U_k}$ is maximized either when U_k is 0 or U_k is f. As a result, we only have to consider two cases when U_k is 0 or U_k is f.

~~When U_k is 0, this problem is reduced to the case with at most $k-1$ tasks in RM-P scheduling, in which the utilization bound is $(k-1)(2^{\frac{1}{k-1}}-1) > k(2^{\frac{1}{k}}-1)$. We focus on the remaining case when $U_k > 0$. This is done by using the Lagrange Multiplier Method~~ We need to find the infimum $\sum_{i=1}^{k} U_i$ such that $\left(\frac{\gamma U_k}{1-U_k}+1\right)\prod_{j=1}^{k-1}(U_j+1) > 2$. The detailed proof to show ~~that $H(k,\gamma)$ is~~ the infimum happens for one of the boundary values of $U_1 \in [0;\ 2^{\frac{1}{k-1}}-1]$ is in the Appendix. The utilization bound for $U_1 = 0$ is $\frac{1}{1+\gamma}$. If $U_1 = (2^{\frac{1}{k-1}}-1)$ the utilization bound is $(k-1)\cdot(2^{\frac{1}{k-1}}-1) > k(2^{\frac{1}{k}}-1)$. Therefore, we reach the conclusion. ■

With the property in Theorem 8, we can now formulate the utilization bounds of RM-NP with respect to γ for sporadic real-time tasks with implicit deadlines.

Theorem 9. *Suppose that* $\gamma = \max_{\tau_k}\left\{\max_{\tau_i \in lp(\tau_k)}\left\{\frac{C_i}{C_k}\right\}\right\}$. *A task set can be feasibly scheduled by RM-NP if*

$$U_{sum} \le \begin{cases} ln(2) \approx 0.693 & \text{if } \gamma \le 1 \\ \frac{\gamma-1}{2\gamma-1} + ln(\frac{2}{\gamma}) & \text{if } 1 < \gamma \le 2 \\ \frac{1}{1+\gamma} & \text{if } 2 < \gamma \end{cases}$$

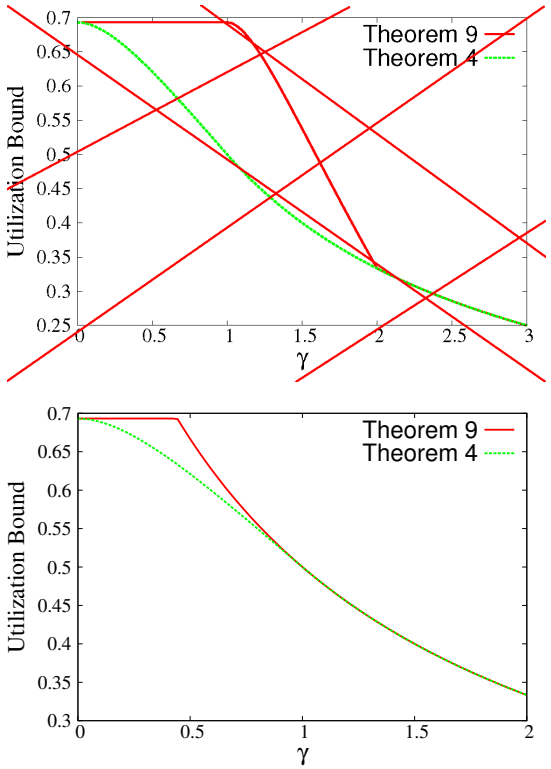

Fig. 1: Comparison of the total utilization bound of RM-NP with respect to $\gamma = \max_{\tau_k}\left\{\max_{\tau_i \in lp(\tau_k)}\left\{\frac{C_i}{C_k}\right\}\right\}$ provided by Theorem 4 and Theorem 9

$$U_{sum} \le \begin{cases} ln(2) \approx 0.693 & \text{if } \gamma \le \frac{1-ln(2)}{ln(2)} \\ \frac{1}{1+\gamma} & \text{if } \gamma > \frac{1-ln(2)}{ln(2)} \end{cases} \quad (42)$$

Proof: This follows directly from Theorem 8 by calculating the utilization bound when $k \to \infty$. ~~, i.e., $\lim_{k\to\infty}(\frac{2}{\gamma})^{\frac{1}{k}} = 1$ and $\lim_{k\to\infty}(\frac{1}{2})^{\frac{1}{k-1}} = 1$. We know that $\lim_{k\to\infty} k(2^{\frac{1}{k}}-1) = \lim_{k\to\infty}(k-1)(2^{\frac{1}{k-1}}-1) = ln(2)$ and $\lim_{k\to\infty} \frac{(\frac{2}{\gamma})^{\frac{1}{k}}-\frac{1}{\gamma}}{1+(\frac{2}{\gamma})^{\frac{1}{k}}-\frac{1}{\gamma}} + (k-1)((\frac{2}{\gamma})^{\frac{1}{k}}-1) = \frac{1-\frac{1}{\gamma}}{2-\frac{1}{\gamma}} + ln(\frac{2}{\gamma}) = \frac{\gamma-1}{2\gamma-1} + ln(\frac{2}{\gamma})$.~~ ■

The result in Theorem 9 further improves the result in Theorem 4. With the analysis in Theorem 9, we can conclude that the utilization bound of RM-NP with respect to γ can still be up to 69.3%, if $\gamma \le \frac{1-ln(2)}{ln(2)} \approx 0.44269$. ~~if none of the lower priority tasks has a larger execution time, i.e., $\gamma \le 1$. As long as the lower-priority tasks do not have much longer execution time than the higher-priority tasks, e.g., $\gamma \le 1.59$, the utilization bound can still be more than 50%.~~ We illustrate the results of Theorems 4 and Theorem 9 in Figure 1.

7 Limited-Preemptive Scheduling

Limited preemptive techniques try to combine the advantages of preemptive and non-preemptive scheduling by limiting the number of preemptions, e.g., [23]. Our proposed schedulability tests can also be easily extended to limited-preemptive scheduling, as long as the pseudo-polynomial time schedulability test can be constructed with the similar forms in Section 3, 4 and 6. Here, we demonstrate how to apply them for the Task Splitting model in [23]. As shown in [23], we can compute the strict upper bound of the blocking time for task τ_k as

$$B_k = \max_{\tau_i \in lp(\tau_k)} \{ \max_{j \in np(\tau_i)} \{C_{i,j}\} \} \quad (43)$$

where $np(\tau_i)$ is the number of non-preemptive regions in τ_i and $C_{i,j}$ is the WCET of the j-th non-preemptive region of τ_i. With the above upper-bounded blocking time, we can revise the TDA-based schedulability test in [6] for task τ_k by using the definition of B_k in Eq. (43) to replace B_k in Eq. (4) in Section 4. Therefore, we can easily apply all our schedulability tests in Theorem 1 and Theorem 2 for limited preemption.

If a task ends with a non-preemptive interval, a tighter schedulability test has been proposed by Yao et al. [23] by considering the last non-preemptive execution interval. Let $C_{k,f}$ be the length of this final non-preemptive section of τ_k and let $C_{k,s} = C_k - C_{k,f}$ be the WCET of τ_k without the final section. We need to ensure that the upper-bounded blocking time, all higher priority tasks $\tau_i \in hp_2^{NP}(\tau_k)$, and the part of τ_k represented by $C_{k,s}$, can be executed before the last non-preemptive section of τ_k. Therefore, the first condition in Lemma 3 is changed to verify whether there exists $t \in (0, D_k - C_{k,f}]$ with $B_k + C_{k,s} + \sum_{\tau_i \in hp(\tau_k)} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t$, as shown in Theorem 2 in [23]. Thus we can reformulate Eq. (35) in Theorem 6 as

$$\left(\frac{B_k + C_{k,s} + \sum_{\tau_i \in hp_2^{NP}(\tau_k)} C_i}{D_k - C_{k,f}} + 1 \right) \prod_{\tau_j \in hp_1^{NP}(\tau_k)} (U_j + 1) \leq 2 \quad (44)$$

if $hp_2^{NP}(\tau_k)$ and $hp_1^{NP}(\tau_k)$ are constructed accordingly.

8 Conclusion

In this paper we provide, to our knowledge, the first schedulability tests for non-preemptive fixed priority scheduling with a hyperbolic structure based on the blocking factor. We lower the upper bound of the speed-up factors of RM-NP and DM-NP in comparison to EDF-NP to $\frac{1}{\Omega} \approx 1.76322$, which closes the gap for implicit-deadline and constrained-deadline systems. We also provide the utilization bound for RM-NP based on the ratio $\gamma > 0$, which has significantly improved previous results in [1].

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).

References

[1] B. Andersson and E. Tovar. The utilization bound of non-preemptive rate-monotonic scheduling in controller area networks is 25%. In *IEEE Fourth International Symposium on Industrial Embedded Systems - SIES*, pages 11–18, 2009.

[2] S. K. Baruah, R. R. Howell, and L. Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.

[3] S. K. Baruah, A. K. Mok, and L. E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *In Proceedings of the 11th Real-Time Systems Symposium*, pages 182–190, 1990.

[4] E. Bini, G. Buttazzo, and G. Buttazzo. A hyperbolic bound for the rate monotonic algorithm. In *Real-Time Systems, 13th Euromicro Conference on, 2001.*, pages 59–66, 2001.

[5] R. J. Bril, J. J. Lukkien, and W. F. Verhaegh. Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption. *Real-Time Systems*, 42(1-3):63–119, 2009.

[6] A. Burns. Preemptive priority-based scheduling: An appropriate engineering approach. In *Advances in Real-Time Systems, chapter 10*, pages 225–248. Prentice Hall, 1994.

[7] J. Chen, W.-H. Huang, and C. Liu. k2U: A general framework from k-point effective schedulability analysis to utilization-based tests. *CoRR*, abs/1501.07084, 2015.

[8] R. Davis, T. Rothvo, 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 Networks and Systems Conference*, 2009.

[9] R. I. Davis, L. George, and P. Courbin. Quantifying the sub-optimality of uniprocessor fixed priority non-pre-emptive scheduling. *International Conference on Real-Time and Network Systems (RTNS'10)*, 2010.

[10] R. I. Davis, T. Rothvoß, S. K. Baruah, and A. Burns. Exact quantification of the suboptimality of uniprocessor fixed priority pre-emptive scheduling. *Real-Time Systems*, 43, 2009.

[11] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *IFIP Congress'74*, pages 807–813, 1974.

[12] L. George, P. Muhlethaler, N. Rivierre, et al. Optimality and non-preemptive real-time scheduling revisited. 1995.

[13] L. George, N. Rivierre, M. Spuri, et al. Preemptive and non-preemptive real-time uniprocessor scheduling. 1996.

[14] R. Howell and M. Venkatrao. On non-preemptive scheduling of recurring tasks using inserted idle times. *Information and Computation*, 117(1):50 – 62, 1995.

[15] 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'89*, pages 166–171, 1989.

[16] J. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic real-time tasks. *Performance Evaluation*, 2:237–250, 1982.

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

[18] D. G. Luenberger and Y. Ye. *Linear and nonlinear programming*, volume 116. Springer, 2008.

[19] A. K. Mok. Fundamental design problems of distributed systems for the hard-real-time environment. Technical report, Cambridge, MA, USA, 1983.

[20] M. Nasri, S. K. Baruah, G. Fohler, and M. Kargahi. On the optimality of RM and EDF for non-preemptive real-time harmonic tasks. In *RTNS*, page 331, 2014.

[21] A. Thekkilakattil, R. Dobrin, and S. Punnekkat. Quantifying the sub-optimality of non-preemptive real-time scheduling. In *Euromicro Conference on Real-Time Systems, ECRTS*, pages 113–122, 2013.

[22] Y. Wang and M. Saksena. Scheduling fixed-priority tasks with preemption threshold. In *International Conference on Real-Time Computing Systems and Applications*, RTCSA '99, pages 328–, 1999.

[23] G. Yao, G. Buttazzo, and M. Bertogna. Feasibility analysis under fixed priority scheduling with fixed preemption points. In *Embedded and Real-Time Computing Systems and Applications (RTCSA)*, pages 71–80, 2010.

Appendix

Proof of Optimality of the extreme point solution in Theorem 1. We have to show, that there is no feasible solution that results in a smaller value for C_k^* then the one derived in Eq. (24). An extreme point solution for $k-1$ variables must

have at least $k-1$ active constraints. This means that $k-1$ constraints out of the Eqs. (20b) and (20c) hold with equality.

We first show that in an extreme point solution for each $\tau_j \in \{\tau_1, \ldots, \tau_{k-1}\}$ either Eq. (20b) or Eq. (20c) is active but not both by assuming that there is a task $\tau_p \in \{\tau_1, \ldots, \tau_{k-1}\}$ with $0 = t_p^* = t_k^* - \sum_{i=p}^{k-1} t_i^* U_i$. Let $\tau_q \in \{\tau_{p+1}, \ldots, \tau_{k-1}\}$ be the next task in the extreme point solution with $t_q^* > 0$, thus $t_p^* = t_{p+1}^* = \ldots = t_{q-1}^* = 0$. If no such τ_q exists, we set $q = k^*$ and $t_q^* = t_k^*$. With these two conditions, we get the contradiction that $0 = t_p^* = t_k^* - \sum_{i=p}^{k-1} t_i^* U_i = \sum_{i=q}^{k-1} t_i^* U_i \geq t_q^* > 0$ for $q \leq k-1$ and $0 = t_p^* = t_k^* - \sum_{i=p}^{k-1} t_i^* U_i = t_k^* > 0$ if $q = k$. Thus we can represent a feasible solution of Eq. (20) by partitioning $\{\tau_1, \ldots, \tau_{k-1}\}$ into two tasks sets $\mathbf{T}_1$ and $\mathbf{T}_2$, where $\tau_j \in \mathbf{T}_1$ if $t_j^* = 0$ (Eq. (20c) holds) and $\tau_j \in \mathbf{T}_2$ if $t_j^* = t_k^* - \sum_{i=j}^{k-1} t_i^* U_i > 0$ (Eq. (20b) holds). As we maximize $\sum_{i=1}^{k-1} t_i^* U_i$ we can simply drop all tasks in $\mathbf{T}_1$ and only use the tasks in $\mathbf{T}_2$ thus leading to the an objective function as in Eq. (24) where only $\tau_j \in \mathbf{T}_2$ are considered. As $\prod_{\tau_j \in \mathbf{T}_2}(U_j + 1) \leq \prod_{j=1}^{k-1}(U_j + 1)$ we maximize the objective in Eq. (20) if all higher priority tasks are in $\mathbf{T}_2$. □

Proof of Theorem 2. We use the contrapositive to prove this theorem as well, showing that if Eq. (15) is not satisfied, Eq. (27) will not be satisfied. We construct a linear programming to find the minimum of $C_k^* + \sum_{i=1}^{k-1} t_i^* U_i$ to ensure Eq. (15) is not satisfied:

$$\inf\ C_k^* + \sum_{i=1}^{k-1} t_i^* U_i \tag{45a}$$

$$\text{s.t } C_k^* + \sum_{i=1}^{k-1} t_i^* U_i + \sum_{i=1}^{j-1} t_i^* U_i > t_j^*, \quad \forall\ 1 \leq j \leq k-1 \tag{45b}$$

$$t_j^* \geq 0, \quad \forall\ 1 \leq j \leq k-1 \tag{45c}$$

where $t_1^*, \ldots, t_{k-1}^*$ and C_k^* are variables and t_k^* is defined as t_k for notational brevity. We replace $>$ with $\geq$ again as infimum and minimum are the same if $\geq$ is used.

When considering Eq. (45b) with $j = k$, we get $C_k^* + \sum_{i=1}^{k-1} t_i^* U_i \geq t_k^* - \sum_{i=1}^{k-1} t_i^* U_i$, thus we can switch to the maximization problem with $\sum_{i=1}^{k-1} t_i^* U_i$ as the objective function. We replace $C_k^* + \sum_{i=1}^{k-1} t_i^* U_i$ with $t_k^* - \sum_{i=1}^{k-1} t_i^* U_i$ in Eq. (45b) resulting in

$$\begin{aligned} & t_k^* - \sum_{i=1}^{k-1} t_i^* U_i + \sum_{i=1}^{j-1} t_i^* U_i \\ = \ & t_k^* - \sum_{i=j}^{k-1} t_i^* U_i \geq t_j^*, \ \forall\ 1 \leq j \leq k-1 \end{aligned} \tag{46}$$

The result is the same linear programming as in Eq. (20), thus finding the same optimal solution as in Theorem 1 with the same properties. From Eq. (45b) for $j = k$ we get

$$\begin{aligned} & C_k^* + \sum_{i=1}^{k-1} t_i^* U_i \geq t_k^* - \sum_{i=1}^{k-1} t_i^* U_i \\ \overset{(21)}{=} \ & t_k^* - (t_k^* - t_1^*) = t_1^* \overset{(23)}{=} \frac{t_k^*}{\prod_{i=1}^{k-1}(U_i + 1)} \\ \Rightarrow \ & \frac{C_k^* + \sum_{i=1}^{k-1} t_i^* U_i}{D_k} = \frac{1}{\prod_{i=1}^{k-1}(U_i + 1)} \end{aligned} \tag{47}$$

We can replace C_k^* with $\widehat{C_k'}$ as we constructed C_k^* as the minimum value to ensure Eq. (15) is not satisfied under the worst case setting of the t_i^* determined by the linear programming. Thus if Eq. (27) holds, Eq. (15) holds as well and the task set is schedulable. □

Proof of Theorem 3. With RM-NP scheduling, we only have to consider the case that $hp_2(\tau_k)$ is empty. This is due to the fact, that a task τ_l can only be in $hp_2(\tau_k)$ if $T_l = T_k$ if RM is used. In this case the value for $\left(\frac{\widehat{C_k'}}{D_k} + 1\right) \prod_{\tau_j \in hp_1(\tau_k)} (U_j + 1)$ only gets smaller as $1 + x + y < 1 + x + y + xy = (1+x)(1+y)$ if $x > 0,\ y > 0$ and $C_i > 0\ \forall i$. The utilization bound can be proved by using Lagrange Multiplier to find the infimum $U_k + \sum_{i=1}^{k-1} U_i$ such that $((1+\gamma) \cdot U_k + 1) \prod_{j=1}^{k-1} (U_j + 1) > 2$. By using the same observation (the arithmetic mean is larger than or equal to the geometric mean) in the proof of Theorem 5, we know that the infimum $\frac{C_k}{T_k} + \sum_{i=1}^{k-1} U_i$ happens when $U_1 = U_2 = \cdots = U_{k-1}$. Thus, there are only two variables U_k and U_1 to minimize $U_k + (k-1)U_1$ such that $((1+\gamma) \cdot U_k + 1)(U_1 + 1)^{k-1} \geq 2$.

Let λ be the Lagrange Multiplier and G be $U_k + (k-1)U_1 - \lambda\left(((1+\gamma) \cdot U_k + 1)(U_1 + 1)^{k-1} - 2\right)$. The minimum $U_k + (k-1)U_1$ happens when

$$\frac{\partial G}{\partial U_1} = (k-1) - \lambda(k-1)\left((1+\gamma) \cdot U_k + 1\right)(U_1 + 1)^{k-2} = 0$$

$$\frac{\partial G}{\partial U_k} = 1 - \lambda(1+\gamma)(U_1 + 1)^{k-1} = 0$$

This implies that $\lambda = \frac{1}{(1+\gamma)(U_1+1)^{k-1}}$. Therefore, by Lagrange Multiplier the above non-linear programming is minimized when U_1 is $U_k + \frac{1}{1+\gamma} - 1$ and

$$2 = \left((\gamma+1)U_k + 1\right)(U_1 + 1)^{k-1} = (1+\gamma)(U_k + \frac{1}{1+\gamma})^k.$$

Therefore, by solving the above equality, we have $U_k = (\frac{2}{1+\gamma})^{\frac{1}{k}} - \frac{1}{1+\gamma}$ and, hence U_1 is $(\frac{2}{1+\gamma})^{\frac{1}{k}} - 1$.

This solution with Lagrange Multiplier will allow U_1 to be negative when $\gamma > 1$. Therefore, we should set U_1 to 0 and U_k to $\frac{1}{1+\gamma}$ when $\gamma > 1$. By putting these two cases together, we reach the conclusion of the proof. □

Proof of Theorem 7. We need to show that under the given assumptions, if Eq. (36) holds, Eq. (35) holds as well. Since $T_i \leq D_k - C_k$ for all $\tau_i \in hp(\tau_k)$, we know, that $hp_2^P(\tau_k)$ and $hp_2^{NP}(\tau_k)$ are both empty, and thus $hp_1^P(\tau_k) = hp_1^{NP}(\tau_k) = hp(\tau_k)$ and $\prod_{\tau_j \in hp_1^N(\tau_k)} (U_j + 1)$ and $\prod_{\tau_j \in hp_1^{NP}(\tau_k)} (U_j + 1)$ are the same. By Eq. (38b), we know that $\frac{B_k}{D_k - C_k} \leq \frac{C_k}{D_k}$, which implies that the success of Eq. (36) implies the success of Eq. (35). □

Proof of ~~Lagrange Multiplier for the proof of~~ Theorem 8. By using the same observation (the arithmetic mean is larger than or equal to the geometric mean) in the proof of Theorem 5, the infimum $\frac{C_k}{T_k} + \sum_{i=1}^{k-1} U_i$ happens when $U_1 = U_2 = \cdots = U_{k-1}$. Thus, there are only two variables U_k and U_1 to minimize $G = U_k + (k-1)U_1$ such that $(\frac{\gamma U_k}{1-U_k} + 1)(U_1 + 1)^{k-1} \geq 2$.

~~Let λ be the Lagrange Multiplier and G be $U_k + (k-1)U_1 - \lambda\left((\frac{\gamma U_k}{1-U_k} + 1)(U_1+1)^{k-1} - 2\right)$. The minimum $U_k + (k-1)U_1$ happens when~~

~~$$\frac{\partial G}{\partial U_1} = (k-1) - \lambda(k-1)\left(\frac{\gamma U_k}{1-U_k} + 1\right)(U_1+1)^{k-2} = 0$$~~

~~$$\frac{\partial G}{\partial U_k} = 1 - \lambda(U_1+1)^{k-1}\frac{\gamma}{(1-U_k)^2} = 0$$~~

~~This implies that $\lambda = \frac{(1-U_k)^2}{\gamma(U_1+1)^{k-1}}$. Therefore, by Lagrange Multiplier the above non-linear programming is minimized when U_1 is $\frac{(1-U_k)(1+(\gamma-1)U_k)}{\gamma} - 1$ and~~

~~$$\begin{aligned} 2 &= \left(\frac{\gamma U_k}{1-U_k} + 1\right)(U_1+1)^{k-1} \\ &= \gamma\left(\frac{U_k}{1-U_k} + \frac{1}{\gamma}\right)\left(\frac{\gamma U_k + 1 - U_k}{\gamma(1-U_k)}\right)^{k-1} \\ &= \gamma\left(\frac{U_k}{1-U_k} + \frac{1}{\gamma}\right)^k \end{aligned}$$~~

~~Therefore, by solving the above equality, we have $U_k = \frac{(\frac{2}{\gamma})^{\frac{1}{k}} - \frac{1}{\gamma}}{1 + (\frac{2}{\gamma})^{\frac{1}{k}} - \frac{1}{\gamma}}$, and, hence, U_1 is $(\frac{2}{\gamma})^{\frac{1}{k}} - 1$.~~

~~The above solution with Lagrange Multiplier will allow U_1 to be negative when $\gamma > 2$. Therefore, by adopting the Kuhn-Tucker condition, we should set U_1 to 0 and U_k to $\frac{1}{1+\gamma}$ when $\gamma > 2$. It also allows U_k to be negative when $\gamma < (\frac{1}{2})^{\frac{1}{k-1}}$. Therefore, by adopting the Kuhn-Tucker condition, we should set U_1 to $(2^{\frac{1}{k-1}} - 1)$ and U_k to 0 when $\gamma < (\frac{1}{2})^{\frac{1}{k-1}}$. By combining these three cases together, we reach our result.~~

For the minimum total utilization this equation holds with equality. We donate $\ell = k - 1$ and get

$$U_k = \frac{(\frac{2}{1+U_1})^\ell - 1}{\gamma + (\frac{2}{1+U_1})^\ell - 1} = 1 - \frac{\gamma}{\gamma + (\frac{2}{1+U_1})^\ell - 1}$$

We use this value to replace U_k in G, thus getting a function of only one variable U_1. To find the minimum value for $G = \ell U_1 + 1 - \frac{\gamma}{\gamma + (\frac{2}{1+U_1})^\ell - 1}$ we calculate the first order derivative:

$$\begin{aligned} \frac{\partial G}{\partial U_1} &= \ell - \frac{\ell\gamma 2(1+U_1)^{-\ell-1}}{(\gamma - 1 + 2(1+U_1)^{-\ell})^2} \\ &= \ell\left[1 - \frac{\gamma 2(1+U_1)^{-\ell-1}}{(\gamma + 2(1+U_1)^{-\ell} - 1)^2}\right] \end{aligned}$$

We know that $U_1 \in [0;\ 2^{\frac{1}{\ell}} - 1]$, as if $U_1 > 2^{\frac{1}{\ell}} - 1$ then $U_k < 0$. We now prove, that the minimal value happens for one of the boundaries of U_1. For $U_1 = 0$ we get $\frac{\partial G}{\partial U_1}(0) = \ell(1 - \frac{2\gamma}{(\gamma+1)^2}) > 0$ as $\gamma > 0$. We further analyze the second order derivative:

$$\begin{aligned} \frac{\partial^2 G}{\partial U_1} &= \frac{-\ell(-\ell-1)\gamma 2(1+U_1)^{-\ell-2}2(1+U_1)^{-\ell}}{(\gamma - 1 + 2(1+U_1)^{-\ell})^4} \\ &\quad + \frac{\ell\gamma 2(1+U_1)^{-\ell-1}2(\gamma - 1 + 2(1+U_1)^{-\ell})(-2\ell)(1+U_1)^{-\ell-1}}{(\gamma - 1 + 2(1+U_1)^{-\ell})^4} \\ &= \frac{2\gamma\ell(1+U_1)^{-\ell-2}[(\ell+1)(\gamma-1) + 2(\ell-1)(1+U_1)^{-\ell}]}{(\gamma - 1 + 2(1+U_1)^{-\ell})^3} \end{aligned}$$

We know that the demoninator is always positive as $U_1 \geq 0$ and $\gamma > 0$. In the numerator the same argument holds for the multiplied term outside the bracket. The first term in the bracket is a constant. $2(\ell-1)(1+U_1)^{-\ell}$ is a decreasing function with respect to U_1. So we can conclude that the second order derivative of G with respect to U_1 in all the values in the range of $[0;\ 2^{\frac{1}{\ell}} - 1]$ is either (1) always positive $\forall U_1 \in [0;\ 2^{\frac{1}{\ell}} - 1]$, (2) always negative $\forall U_1 \in [0;\ 2^{\frac{1}{\ell}} - 1]$, or (3) changing from positive to negative at a certain value U_1^* for $U_1 \in [0;\ 2^{\frac{1}{\ell}} - 1]$. For the first case the minimum happens when $U_1 = 0$. In the second case and the third case the minimum is in one of the boundary conditions, since $\frac{\partial G}{\partial U_1} = 0$ happens when $\frac{\partial^2 G}{\partial U_1} < 0$ and in this case we get a local maximum. □