Weakly Hard Schedulability Analysis for Fixed Priority Scheduling of Periodic Real-Time Tasks

Anonymous Author(s)

ABSTRACT

The hard deadline model is very popular in real-time research, but is representative or applicable to a small number of systems. Many applications, including control systems, are capable of tolerating occasional deadline misses, but are seriously compromised by a repeating pattern of late terminations. The weakly hard real-time model tries to capture these requirements by analyzing the conditions that guarantee that a maximum number of deadlines can be possibly missed in any set of consecutive activations. We provide a new weakly hard schedulability analysis method that applies to constrained-deadline periodic real-time systems scheduled with fixed priority and without knowledge of the task activation offsets. The analysis is based on a Mixed Integer Linear Programming (MILP) problem formulation; it is very general and can be adapted to include the consideration of resource sharing and activation jitter. A set of experiments conducted on an automotive engine control application and randomly generated tasksets show the applicability and accuracy of the proposed technique.

1 INTRODUCTION

The notion of a real-time task abstracts the execution of a program (e.g., a thread) triggered repeatedly by a (periodic) clock or a generic event, with a set of applied temporal constraints. Given a model of execution for a set of tasks and a scheduler, *hard* real-time schedulability analysis verifies if a system, that is, all of its tasks, can be safely guaranteed to complete at every activation before a deadline. The hard schedulability problem has been thoroughly investigated in the last decades and solved for many cases of interest.

However, many systems, including control systems, are tolerant to individual deadline misses and treating them as hard real-time would result in unnecessary pessimism and possibly overprovisioning of resources. Of course, uncontrolled response times are also not desirable and even in case of deadline misses the designer may require some guarantees on the timing behavior of the system.

Among the possible options, the *weakly hard* scheduling model has been proposed by several authors (Hamdaoui et al. [24] and Bernat et al. [6]) to check for a relaxed condition, that is, to analyze the number of temporal constraints violations given a time window or a sequence of task activations. This is also called m-K model, since a typical formulation consists in checking that no more than m deadlines are missed over a set of K activations. The rationale behind this model is that each deadline miss will bring the system closer to a faulty condition or state, and each completion in time will move the system back towards a safe working state.

The weakly hard model can be analyzed under several scheduling options, but due to its simplicity and effectiveness, fixed-priority scheduling is nowadays the de-facto standard for industrial real-time systems. This paper focuses on real-time systems consisting of periodic tasks scheduled with fixed priority.

The main problem with the existing analysis methods is that for a weakly hard system, as for any system that can be (at least temporarily) overloaded, the critical instant theorem does not hold and the system becomes much more difficult to analyze. In the original work, this limitation has been overcome by restricting the study to offset-determined systems, that is, systems in which tasks are scheduled with a known initial offset.

This is a possibly serious limitation, not only because the designer may not be able to enforce or determine the system initial offsets, but especially because the analysis of all offset-determined systems is very sensitive to time drifts and errors in the task activation times, which are extremely hard to avoid in real systems.

In this work, we provide a generalized framework for offset-free systems scheduled on uniprocessors. Our formulation is based on a MILP encoding and the problem of the critical instant determination is addressed by letting a variable represent the beginning of the busy period. The MILP formulation can be easily reused to check the system for a large set of m and K values, allowing the designer to explore a wide design range. The developed MILP model serves as an over-approximate analysis, that is, if our weakly hard analysis confirms the m-K property, it is guaranteed that there will be no more than m deadline misses out of any K successive job activations (of the target task), however, if the m-K property is not confirmed, we cannot conclude the opposite.

Main contributions:

- We propose the first weakly hard schedulability analysis for offset-free periodic real-time tasks. The analysis method includes the consideration of resource sharing and activation jitter.
- To solve the possible issues with the large number of integer variables counting the number of task interferences (as used, for example, in [7, 19, 32]), we relaxed these variables to real values, but we added binary variables expressing the possibility of job interferences for reducing or possibly eliminating the introduced pessimism.
- Surprisingly, there is no existing work that can cope with the weakly hard analysis of general (offset-free) periodic tasks, and this prevents a fair comparison between our solution and other relevant works. Thus, we evaluate our analysis method through extensive experiments to show its efficiency (expected runtime) and precision. In the special case in which $m = 1$, the analysis is always accurate as

xxx, xxx

DOI: xxx

long as it validates the *m-K* property. With respect to accuracy, despite the relaxation to real valued job interference counters, the MILP analysis can still return exact results for a very high percentage of the tests.

Paper Organization: Section 2 introduces background results and possible applications that are related with the scheduling in overload conditions, the weakly hard real-time model and its analysis techniques. Next, we formally define the system model and the weakly hard analysis problem in Section 3. Section 4 contains the description of our proposed solution to the problem, which consists of a Mixed Integer Linear Programing (MILP) formulation. As a demonstration of the generality of the solution model, it is then extended in Section 5 to systems having mutual exclusive shared resources, and to tasks with jitter. The evaluation of the proposed technique is conducted in Section 6, and at last, Section 7 contains the conclusions and addresses future issues.

2 STATE OF THE ART

Since the seminal work of Liu and Layland [23], an overwhelming effort in real-time scheduling research has been dedicated to the question on whether there can be *a possible deadline miss* according to the *hard* real-time model. It is hard to completely identify the reasons for this disproportionate interest in hard analysis techniques. It is probably because of the simplicity of the model, its easier understandability and analyzability, the seemingly natural fit to safety-critical systems and, quite possibly, some incorrect judgement on the part of some researchers that believe most real-world systems are of hard-type. The success of hard schedulability analysis also benefits from the existence of the critical instant, an activation scenario where all tasks are simultaneously activated, that leads to the worst-case response time for every task in the system. As a result, in a hard real-time system it is sufficient to investigate the particular task activation pattern that originates from the critical instant. More details on classic hard real-time schedulability analysis can be found in textbooks and surveys like [12] and [28].

A simplistic version of the periodic task model assumes that the activation time of the first instance, or initial offset, of a task is known. For a system of periodic tasks with explicit initial offsets, Leung and Whitehead [22] proved that, starting from any job activation, it is necessary and sufficient to simulate the worst-case execution of the tasks in a bounded time interval to check if there is any deadline miss in the system, as the schedule of periodic tasks will repeat itself. However, as explained in [5], the result of this test is very sensitive to task parameters, including the initial offset.

While it is true that some safety-critical systems are vulnerable to a single violation of the temporal constraints, there are many more that can tolerate timing violations. In these cases, the hard schedulability analysis is too strict. The *weakly* hard real-time schedulability analysis targets the problem of *bounding the maximum number of deadline misses over a number of task activations.* A dynamic priority assignment of priority for streams with m-k requirements is proposed by Hamdaoui et al. [24] to reduce the probability of *m-K* violations in time-sensitive applications. Weakly hard real-time schedulability analysis can be traced back to the work of Bernat et al. [6] on the *m-K* model, in which no more than *m* deadline misses shall occur for any *K* consecutive activations of a task. The analysis in [6] and in other works assumes that there is an explicit initial state of the system, in which the initial offset of each task in the system is known. By restricting the analysis to a periodic activation pattern, the weakly hard analysis can be conducted by checking task activations and interleavings within a large enough time span from the system initialization, so as to verify the *m-K* assumption. Periodic tasksets are quite common in real applications, but the requirement of knowing all activation offsets may be too strict and undermine the robustness of the analysis: given a periodic task system with explicit initial offsets that passes the (weakly) hard test, a slightly change of the initial offset of some task may result in an unexpected time failure. The analysis is also very sensitive to a drift of the task periods.

Recent developments in the study of overloaded systems allow to relax the requirement of knowing the initial system state. The approach consists in the *worst-case analysis* [25] of a system model represented as the superposition of a typical behavior (e.g., of periodic task activations) that is assumed feasible, and a sporadic overload (i.e., a rare event). Under such an assumption, [18] and [33] proposed methods for weakly hard analysis that is composed by two phases: 1) the system is verified to be schedulable under the typical scenario (by the classical hard analysis), and 2) when the system is overloaded, it can be guaranteed that out of *K* successive activations of a task, at most *m* of them will miss the deadline. The sporadic behavior can be abstracted by observing and analyzing the system at runtime, and is characterized as a rare event. A similar approach is considered in [20], where real-time calculus is used to analyze the worst-case busy period (in duration and lateness) that results from a temporary overload because of an error in the timing assumptions on the task worst-case execution times. Both methods require the definition of a task model that may be artificial, since it requires the identification and separation of possible causes of overload. Finally, at least in principle, probabilistic analysis of deadline misses, such as the analysis in [14] could be used to compute the probability of missing m deadlines over k instances, but the model in [14] also assumes known activation offsets and is likely to be computationally extremely expensive when applied to the m-k analysis.

The analysis of overload conditions is also closely related to the co-design of control and CPU-time scheduling [4]. The influence of the response times on the performance of control tasks has been studied in several works such as [34] and [35]. [3] proposed an integrated approach for controller synthesis, selecting the task parameters that meet the expected control performance and guarantee the stability. [2] extended the idea in [3] from uniprocessors to distributed cyber physical systems, and in [17] FlexRay is considered as the communication medium. The *m-K* model has also been investigated in the co-design of controls (with respect to their performance) and scheduling [16, 26], and is used in [10, 11, 30, 31] to define the maximum number of samples (jobs) that can be dropped over any number of consecutive samples (density of dropped samples), to guarantee a minimum level of quality to the controls. However, [11] and [10] do not provide methods for the schedulability analysis of a system and the density of dropped samples needs to be enforced by the operating system or smart sensors.

3 THE SYSTEM MODEL

A periodic real-time task is characterized by a tuple $\tau_i = (C_i, D_i, T_i)$ with $C_i \leq D_i \leq T_i$ such that C_i is its Worst-Case Execution Time (WCET), D_i is the relative deadline and T_i is the period for the activation of τ_i. A task utilization is defined as $U_i = \frac{C_i}{T_i}$.

Each activation (instance) of τ_i is denoted by a job $J_{i,k}$ with $k = 1, 2, \ldots$ representing the job index (of its activations in time). A job $J_{i,k}$ can be further represented by its activation (or arrival) time $a_{i,k}$, its absolute deadline $d_{i,k} = a_{i,k} + D_i$, and its finish time $f_{i,k}$.

A job is schedulable if it can finish its execution before its deadline, i.e., $f_{i,k} \leq d_{i,k}$; and a task τ_i is schedulable if all its jobs are schedulable. The elapsed time between a job finish time and its activation time (i.e., $f_{i,k} - a_{i,k}$) is its *response time*. By definition, a task τ_i is schedulable if and only if the Worst-Case Response Time (WCRT) $R_i = \max_k\{f_{i,k} - a_{i,k}\}$ among all its jobs is not larger than its relative deadline D_i. A task Best-Case Response Time (BCRT) $r_i = \min_k\{f_{i,k} - a_{i,k}\}$ is the minimum time that it takes to complete the execution of the task.

In the periodic activation pattern $a_{k+1} - a_k = T_i$ for any two successive jobs of a task. As we do not require a specific initial offset for a task, the first job activation time $a_{i,1}$ of τ_i is unknown. A job (and the corresponding task) is said to be *active* if it has been activated but has not completed its execution.

The periodic taskset $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ executes upon a uniprocessor platform. Each task in $\mathcal{T}$ is assigned a unique and static priority, and tasks are scheduled by a *fixed priority preemptive scheduler*. Tasks are ordered in $\mathcal{T}$ from higher to lower priority. That is, τ_j has a higher priority than τ_i, if $j < i$. If a task does not always finish before its deadline, it can have multiple active jobs at the same time. In such cases, these jobs are served in FIFO order.

A *level-i busy period* is defined as a time interval during which the processor is always occupied by the execution of tasks with priority higher than or equal to τ_i. For example, in Figure 1, $[s_0, f_2)$ and $[a_3, f_3)$ are level-3 busy periods: Because the focus of this paper is not computing the task WCRT, but analyzing all possible windows with missed deadlines, the definition of busy period extends the maximal level-i busy period as defined in [21].

The execution of any job of τ_i can only be affected by the workload of interfering tasks (including τ_i itself) within the same level-i busy period. According to [21], the WCRT R_i of a task τ_i is found within the longest level-i busy period, which starts at the critical instant (i.e., when all tasks are activated at the same time). In case a task always completes before its next activation, the task schedulability can be easily checked by computing the response time of the first task instance inside it.

However, this condition does not hold for weakly hard real-time systems when deadlines can be missed and multiple instances can be active at the same time. In this case, the WCRT of a task τ_i does not necessarily happen for the first job in a level-i busy period. However, the BCRT is still occurring for the last job in a level-i busy period. Algorithms to compute the BCRT can be found in [9, 27]. In this work, we trivially assume that the BCRT of a task does not exceed its period: $r_i \leq T_i$. Otherwise, the task simply misses all its deadlines. Also, we assume that the BCRT r_i and the WCRT R_i of each task are computed in advance using established techniques such as in [9, 21, 27]. Once computed, these values can be used as parameters in the MILP formulation.

We assume the system utilization $U = \sum_{1 \leq i \leq n} U_i$ is lower than 1, meaning that each job is guaranteed to complete its requested execution at some point in time, whether it misses its deadline or not. In addition, we use the result in [5] that shows that fixed priority scheduling analysis is sustainable, that is, if a job is not schedulable, then it will not become schedulable by increasing its execution time. From the analysis point of view, we simply assume that a task requests its WCET every time it is activated.

3.1 The weakly hard model

This subsection formalizes the problem of weakly hard schedulability analysis. The analysis applies to an arbitrary task $\tau_i \in \mathcal{T}$, also defined as *target task*. An arbitrary sequence of K successive activations of τ_i is considered, with the objective of checking whether there are more than m deadline misses for τ_i in this sequence.

For simplicity, the jobs of τ_i in the activation sequence are denoted by $J_1, \ldots, J_k, \ldots, J_K$ (without the task index). Given a job J_k, its activation time and finish time are defined as a_k and f_k, respectively. The time interval $[a_k, a_{k+1}[$ is called the *kth job window* of τ_i, and the *problem window* for the analysis is $[s_0, f_K[$, where s_0 (also considered as the time reference $s_0 = 0$) is the earliest time instant such that the processor is fully occupied by the execution of higher priority tasks from s_0 to a_1.

As an example, consider a system of 3 tasks: $(C_1 = 1, D_1 = 3, T_1 = 3)$, $(C_2 = 3, D_2 = 5, T_2 = 15)$ and $(C_3 = 2, D_3 = 6, T_3 = 6)$, with $K = 3$ and τ_3 is the target task. Figure 1 shows a scenario where 2 (J_1 and J_3) out of 3 jobs in the problem window $[s_0, f_3[$ miss the deadline. In this case, $s_0 = 0$ and $a_1 = 0.5$. If the problem window starts at the critical instant, that is, when all tasks are synchronously activated in s_0, only J_1 misses its deadline.

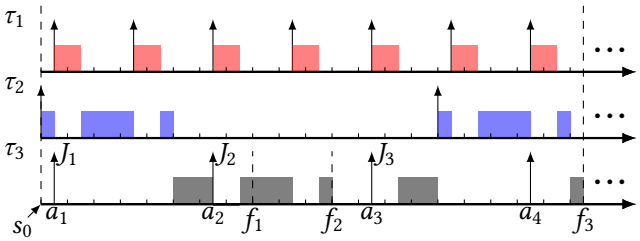


Figure 1: A problem window with 3 job windows

4 THE SOLUTION MODEL

In this section we introduce the Mixed Integer Linear Programming (MILP) formulation for the weakly hard analysis of a set of offset-free periodic tasks under fixed priority scheduling. Two observations allow to reduce the problem space by considering only the problem windows that maximize the number of deadline misses for τ_i.

(**O1**) The worst-case number of deadline misses occurs for problem windows such that the last job of τ_i before the beginning of the problem window (indicated as J_0) is schedulable.

If J_0 is not schedulable, any problem window of k instances starting with J_0 has at least as many misses as the window starting

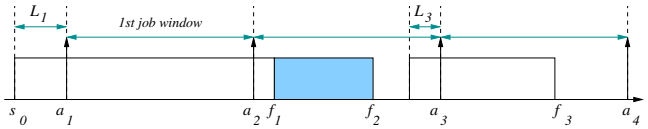


Figure 2: Notation for the definition of a problem window

with the first job J_1. The two windows have all the jobs from J_1 to J_{k-1} in common, but J_0 misses its deadline, therefore, in the best case the two windows have the same number of misses if also J_k is a deadline miss. In Figure 1, consider the problem window with J_2, J_3 and the following instance (not shown) J_4. Depending on the schedulability of J_4, there will be 1 or 2 deadline misses in this window. However, since J_1 is non-schedulable, there are 2 deadline misses for the problem window including J_1, J_2 and J_3.

(**O2**) The worst-case number of deadline misses occurs for problem windows such that the the first job is non-schedulable.

Consider a window of K instances that starts with a set of schedulable jobs of arbitrary length $J_1, \ldots J_n$ (with $n < K$; if $n = K$ the proof is trivial) and the window that starts with J_{n+1}; the latter has at least as many deadlines misses as the window starting with J_1 (the proof is similar to the previous case).

4.1 Variables and basic constraints

In this subsection, we introduce the (real and boolean) variables defined in our MILP model, together with some basic constraints on them. Real valued variables are labeled by $\mathbb{R}$ and boolean variables by $\mathbb{B}$. M is a big enough constant value used to encode conditional constraints (a standard technique known as big-M). A brief summary of all the optimization variables is in Table 1.

Busy periods. Each job of τ_i inside the problem window can be interfered with pending executions of higher priority tasks that are requested before its activation but have not completed. The real valued L_k (as in Figure 2) indicates the portion of the level-i busy period for job J_k that extends to the earliest such activation, when $f_{k-1} \leq a_k$. That is, if J_{k-1} finishes execution not later than a_k (that is, J_{k-1} does not interfere with the execution of J_k), $a_k - L_k$ is the earliest time instant such that the processor in $[a_k - L_k, a_k)$ is fully occupied by higher priority tasks.

The start time $s_0 = 0$ of our problem window is $a_1 - L_1$, and the arrival time a_k of the kth job of τ_i in the problem window is $a_k = L_1 + (k-1) \cdot T_i$. A trivial constraint that applies to all L_k is

$$\forall k \quad 0 \leq L_k \leq T_i - r_i \tag{1}$$

Throughout our analysis, we only use the value of L_k, when $f_{k-1} \leq a_k$. If L_k is larger than $T_i - r_i$ then $f_{k-1} > a_k$ (any job of τ_i needs at least r_i to complete). In Figrue 1, $L_1 = 0.5$ and $L_3 = 0$. Because J_1 interferes with the execution of J_2, L_2 is not relevant to our analysis.

Offsets. The offset of a higher priority task within the problem window refers to its first job activation time with respect to the start time s_0. The first job activation that happens no earlier than s_0 for each higher priority task τ_j is denoted by $\alpha_j \in \mathbb{R}$

$$\forall j < i \quad 0 \leq \alpha_j \leq T_j - r_j \tag{2}$$

Type	Variables	Annotations
$\mathbb{R}$	L_k	Segment of busy execution of higher priority tasks in front of (before) a job window, when $f_{k-1} \leq a_k$.
	α_j	Activation time of the 1st job of each higher priority task τ_i with respect to the start time s_0 of the analysis.
	f_k	Finish time of J_k.
	ι_k	Processor idle time inside the job window.
	$If_{j,k}$	#jobs of τ_j within the time interval $[0, f_k[$.
	$IL_{j,k}$	#job of τ_j inside $[0, a_k - L_k[$.
$\mathbb{B}$	b_k	$b_k = 0$ if J_k is schedulable; $b_k = 1$ if J_k misses its deadline. The number of deadline misses is $\sum_k b_k$.
	β_k	$\beta_k = 0$ if J_k completes before a_{k+1}; otherwise $\beta_k = 1$ and J_k interferes with J_{k+1}.
	$\Gamma f_{j,k,p}$	$\sum_p \Gamma f_{j,k,p}$ is #jobs of τ_j inside: 1) $[a_k - L_k, f_k[$ when $k = 1$ or $\beta_{k-1} = 0$; 2) $[f_{k-1}, f_k[$ when $k > 1$ and $\beta_{k-1} = 1$.
	$\Gamma L_{j,k,p'}$	$\sum_{p'} \Gamma L_{j,k,p'}$ is #jobs of τ_j inside $[f_{k-1}, a_k - L_k[$, if it exists.

Table 1: A summary of variables defined

Assume that the first job $J_{j,1}$ of τ_j in the window arrives at time $s_0 + T_j - r_j + \epsilon$ with $\epsilon > 0$. This implies that the previous job $J_{j,0}$ is activated at time $s_0 - r_j + \epsilon$. Because any job of τ_j needs at least r_j time to finish, $J_{j,0}$ will be still active at time instant s_0, which contradicts the hypothesis that s_0 is the earliest time instant such that from s_0 to a_1 the processor is fully occupied by higher priority tasks. Hence, the upper bound for α_j is $T_j - r_j$.

Finish times. For each job J_k of τ_i, its finish time is denoted by $f_k \in \mathbb{R}$

$$\forall k \quad r_i \leq f_k - a_k \leq R_i$$

Because jobs from the same task are executed sequentially, for any two consecutive jobs of τ_i, this precedence constraint is encoded as

$$\forall k \quad C_i \leq f_{k+1} - f_k \leq R_i + (T_i - r_i) \tag{3}$$

Level-i idle time inside a job window. The level-i processor idle time refers to the time when the processor is not occupied for execution by τ_i or any other higher priority task (in a given time interval). Given an arbitrary job window $[a_k, a_{k+1}[$ of J_k, we define $\iota_k \in \mathbb{R}$ as the amount of processor idle time inside this kth job window

$$\forall k \quad 0 \leq \iota_k \leq a_{k+1} - a_k - r_i$$

Schedulability of each job of τ_i. For each job J_k of τ_i inside the problem window, a boolean variable $b_k \in \mathbb{B}$ indicates whether the job misses its deadline:

- $b_k = 0$ if J_k finishes its execution no later than its deadline;
- $b_k = 1$ otherwise.

The value of b_k is defined by the comparison between the finish time f_k of J_k and its absolute deadline $a_k + D_i$: $b_k = 0 \Leftrightarrow f_k \leq a_k + D_i$, which is encoded by the following linear constraint.

$$\forall k \quad -M \cdot b_k \leq a_k + D_i - f_k < M \cdot (1 - b_k) \tag{4}$$

Being M a very large value, the conditional constraint in (4) forces $b_k = 0$ if the job J_k meets its deadline (i.e., $f_k \leq a_k + D_i$) and $b_k = 1$ otherwise. As in observation **O2** at the beginning of Section 4, we require that J_1 misses its deadline, that is, $b_1 = 1$ (schedulable tasks can be ruled out by simply performing a traditional hard schedulability test in advance).

The *total number of deadline misses* of τ_i inside the problem window is denoted by $\sum_k b_k$.

Interference from the previous jobs of the same task. A job J_k of τ_i interferes with the execution of the next job J_{k+1} in case $f_k > a_{k+1}$. The boolean variable β_k encodes this condition.

- $\beta_k = 0$ if J_k finishes its execution within its own job window;
- $\beta_k = 1$ if J_k completes after a_{k+1}.

Similarly as in (4), the constraint $\beta_k = 0 \Leftrightarrow f_k \leq a_{k+1}$ over β_k, f_k and a_{k+1} can be formulated as

$$\forall k \quad -M \cdot \beta_k \leq a_{k+1} - f_k < M \cdot (1 - \beta_k) \tag{5}$$

If there is idle processor time inside the job window $[a_k, a_{k+1}[$ of J_k, then J_k must terminate within its window and does not interfere with J_{k+1} (i.e., $\beta_k = 1 \Rightarrow \iota_k = 0$).

$$\forall k \quad \iota_k \leq M \cdot (1 - \beta_k)$$

Number of interfering jobs from higher priority tasks. When modeling a schedulability problem in MILP, the major complexity comes from computing the interference from higher priority tasks. A common approach (as in [7, 19, 32]) is to count the number of jobs from each higher priority task that interfere with the execution of the task under analysis. Different from previous works, we explore the relaxation of this integer count to a real value. Table 2 summarizes the variables defined for counting the higher priority interferences for the example system in Figure 1

Given a job J_k of τ_i and a higher priority task τ_j, $If_{j,k}$ is the number of job instances of τ_j within the time interval $[0, f_k[$. By definition, $If_{j,k} = \lceil \frac{f_k - \alpha_j}{T_j} \rceil$ is an integer number. However, we relax the definition of $If_{j,k}$ allowing it to be a real value and we linearize the constraint on $If_{j,k}$ as (by the definition in [21])

$$\forall j < i \quad 0 \leq If_{j,k} - \frac{f_k - \alpha_j}{T_j} < 1 \tag{6}$$

Moreover, we define $IL_{j,k} \in \mathbb{R}$ as the number of jobs of τ_j ($\forall j < i$) within the time interval $[0, a_k - L_k[$, when $\beta_{k-1} = 0$. In this case, If J_k is not interfered with its predecessor J_{k-1}, then the number of jobs from τ_j that interfere with the execution of J_k is $If_{j,k} - IL_{j,k}$. We remind that we only use the value of L_k, and thus the interval $[0, a_k - L_k[$, when $\beta_{k-1} = 0$.

Formally, if J_{k-1} completes before the activation of J_k (i.e., $\beta_{k-1} = 0$), then $IL_{j,k} = \lceil \frac{a_k - L_k - \alpha_j}{T_j} \rceil$. That is, $\beta_{k-1} = 0 \Rightarrow 0 \leq IL_{j,k} - \frac{a_k - L_k - \alpha_j}{T_j} < 1$. In other words, $\forall j < i$

$$-M \cdot \beta_{k-1} \leq IL_{j,k} - \frac{a_k - L_k - \alpha_j}{T_j} < 1 + M \cdot \beta_{k-1} \tag{7}$$

In case $k = 1$, by the definition of the starting time instant $s_0 = 0$, it must be $\forall j < i : IL_{j,1} = 0$.
For simplicity, when $\beta_{k-1} = 1$, we force $IL_{j,k} = If_{j,k-1}$:

$$\forall j < i \quad -M \cdot (1 - \beta_{k-1}) \leq IL_{j,k} - If_{j,k-1} \leq M \cdot (1 - \beta_{k-1}) \tag{8}$$

	$j = 1$			$j = 2$		
	$k = 1$	$k = 2$	$k = 3$	$k = 1$	$k = 2$	$k = 3$
$If_{j,k}$	3	4	7	1	1	2
$IL_{j,k}$	0	3	4	0	1	1
$\Delta_{j,k}$	3	1	3	1	0	1
$\Gamma_{j,k}$	0	0	0	0	0	0

Table 2: The counting of higher priority jobs

Refining the interferences from higher priority tasks. Both $If_{j,k}$ and $IL_{j,k}$ are real variables. This is efficient but inaccurate. To restore a correct formulation, we define two classes of boolean variables to constrain the values of $If_{j,k}$ and $IL_{j,k}$.

Given a job J_k of τ_i and a higher priority task τ_j, an array of boolean variables $\Gamma f_{j,k}[p] \in \mathbb{B}$ counts the number of jobs (i.e., job releases) of τ_j inside the time interval (p indexes these jobs).

- $[a_k - L_k, f_k[$ if J_{k-1} does not interfere with J_k, i.e., $\beta_{k-1} = 0$;
- $[f_{k-1}, f_k[$ if J_{k-1} does interfere with J_k, i.e., $\beta_{k-1} = 1$.

A rough bound for the size of $\Gamma f_{j,k}[\cdot]$ (the number of instances of τ_j in the interval) is

$$\lceil \frac{R_i + T_i - r_i}{T_j} \rceil \tag{9}$$

$\Gamma f_{j,k}[p] = 1$ indicates that the pth job activation of τ_j in the specified time interval can interfere with the execution of J_k (the pth job is activated before J_k completes, otherwise, $\Gamma f_{j,k}[p] = 0$). The total number of activations of jobs of τ_j, interfering with the execution of J_k in the specified time interval is

$$\Delta_{j,k} := \sum_p \Gamma f_{j,k}[p]$$

As shown in Table 2, for the example in Figure 1, when $j = 1$ and $k = 3$ it is $L_3 = 0$ and $\Delta_{1,3}$ jobs from τ_1 within the time interval $[a_3 - L_3, f_3[$. As $\beta_1 = 1$, from f_1 to f_2, $\Delta_{1,2} = 1$.

In case J_{k-1} does not delay the execution of J_k, it is $IL_{j,k} + \Delta_{j,k} = If_{j,k}$. In the other case (i.e., when $\beta_{k-1} = 1$) $If_{j,k-1} + \Delta_{j,k} = If_{j,k}$ and (8) enforces $IL_{j,k} = If_{j,k-1}$. Consequently,

$$\forall j < i \quad IL_{j,k} + \Delta_{j,k} = If_{j,k} \tag{10}$$

If a higher priority job $J_{j,p}$ does not interfere with the job J_k of the target task (i.e., $\Gamma f_{j,k}[p] = 0$), this implies that J_k completes before $J_{j,p}$, then no later job $J_{j,p'}$ ($p' > p$) can interfere with J_k. This results in the precedence constraint between elements in $\Gamma f_{j,k}$.

$$\forall j < i \quad \Gamma f_{j,k}[p+1] \leq \Gamma f_{j,k}[p]$$

Similarly, given a job J_k and a higher priority task τ_j, we define an array of boolean variables $\Gamma L_{j,k}[\cdot]$ to count the number of job instances of τ_j inside the time interval $[f_{k-1}, a_k - L_k)$. The size of $\Gamma L_{j,k}[\cdot]$ can also be bounded (e.g., by $\lceil \frac{T_i - r_i}{T_j} \rceil$), and the total number can be computed as

$$\Lambda_{j,k} := \sum_p \Gamma L_{j,k}[p]$$

$\Lambda_{j,k}$ counts the number of jobs from a higher priority task τ_j that are guaranteed not to interfere with J_{k-1} or J_k since they are activated after J_{k-1} finishes and are not in the same busy period with J_k. For instance, during the interval $[f_2, a_3 - L_3[$ in Figure 1, no higher priority jobs are released: $\Gamma_{1,3} = \Gamma_{2,3} = 0$.

When $\beta_{k-1} = 0$, it is $If_{j,k-1} + \Lambda_{j,k} = IL_{j,k}$:

$$\forall j < i \quad -M \cdot \beta_{k-1} \leq IL_{j,k} - \Lambda_{j,k} - If_{j,k-1} \leq M \cdot \beta_{k-1} \tag{11}$$

In case $\beta_{k-1} = 1$, the interval $[f_{k-1}, a_k - L_k[$ is not relevant to our analysis and we force $\Lambda_{j,k} = 0$. Using the big-M formulation.

$$\forall j < i \quad -M \cdot (1 - \beta_{k-1}) \leq \Gamma L_{j,k}[p] \leq M \cdot (1 - \beta_{k-1}) \tag{12}$$

The constraint between variables in $\Gamma L_{j,k}[\cdot]$ resulting from the execution in FIFO order of the jobs from the same task can be encoded as

$$\forall j < i \;\; \Gamma L_{j,k}[p+1] \leq \Gamma L_{j,k}[p]$$

4.2 Constraints on the idle time and workload

In this subsection, we present the constraints that bound the processor idle times and the time spent executing by the tasks (i.e., workload) inside the problem window and its sub parts (e.g., one or multiple job windows). For short, we first define several terms: $\rho_k = f_k - a_k$, $\lambda_k = f_k - (a_k - L_k)$ and $\lambda'_k = f_k - f_{k-1}$. As an example, in Figure 1, $\rho_1 = 7.5$, $\lambda_1 = 7$ and $\lambda'_2 = 3$.

Minimum level-i idle time. To analyze the target task τ_i under fixed priority scheduling, it is sufficient to consider the taskset composed by τ_i and its higher priority tasks: $\mathcal{T}_i = \{\tau_1, \ldots, \tau_i\}$. Given an arbitrary time interval of length X, we use $\mathsf{minIdle}(\mathcal{T}_i, X)$ to denote the minimum amount of (level-i) processor idle time that is available within it (left unused by the tasks in $\mathcal{T}_i$).

Then, for any number x of consecutive job windows inside the problem window (of length $x \cdot T_i$), the total amount of idle time is lower bounded by $\mathsf{minIdle}(\mathcal{T}_i, x \cdot T_i)$. That is, $\forall 1 \leq x \leq K, 1 \leq y \leq K - x + 1$:

$$\mathsf{minIdle}(\mathcal{T}_i, x \cdot T_i) \leq \sum_{y \leq k \leq y+x-1} \iota_k \tag{C1}$$

To compute $\mathsf{minIdle}(\mathcal{T}_i, X)$, we define a virtual task $\tau_* = (-, X, X)$ that has relative deadline and period equal to the interval length X and lowest priority. $\mathsf{minIdle}(\mathcal{T}_i, X)$ is estimated as the maximum execution time C of τ_* that still guarantees its schedulability: if C is not the minimum level-i idle time, then there should exist a combination of job activations for tasks in $\mathcal{T}_i$ that leads to a deadline miss for τ_* (easily demonstrated by contradiction; slack stealing algorithms [13] provide methods to estimate the processor idle time).

Idle time inside a job window. Consider the job window $[a_k, a_{k+1}[$ of J_k, if $\beta_k = 0$, ι_k is in fact the idle time the interval $[f_k, a_{k+1} - L_{k+1}[$, as exemplified by the 2nd job window in Figure 1. The total amount of higher priority workload in $[f_k, a_{k+1} - L_{k+1}[$ can be represented as

$$\Theta_k := \sum_{j<i} (IL_{j,k+1} - If_{j,k}) \cdot C_j$$

As a result, the idle time in the kth job window is $\iota_k = (a_{k+1} - a_k) - \rho_k - L_{k+1} - \Theta_k$. This equivalence only applies when $\beta_k = 0$ and can be encoded in a MILP formulation with the following constraint (trivially true for $\beta_k = 1$).

$$\forall k \quad -M \cdot \beta_k \leq \iota_k + \Theta_k - (a_{k+1} - a_k - \rho_k - L_{k+1}) \leq M \cdot \beta_k \tag{C2}$$

Formulation of the busy period $[a_k - L_k, f_k)$ when $\beta_{k-1} = 0$. If $\beta_{k-1} = 0$, J_{k-1} does not interfere with the execution of J_k, and $[a_k - L_k, f_k[$ is a busy period with length λ_k. The total amount of workload from higher priority tasks inside $[a_k - L_k, f_k[$ is

$$\Phi_k := \sum_{j<i} (If_{j,k} - IL_{j,k}) \cdot C_j$$

For the first instance $k = 1$, it is

$$\Phi_1 + C_i - \lambda_1 = 0 \tag{C3}$$

Otherwise, $\beta_{k-1} = 0$ implies that $\Phi_k + C_i = \lambda_k$. To apply the constraint only to the case $\beta_{k-1} = 0$, the formulation is

$$\forall k \quad -M \cdot \beta_{k-1} \leq \Phi_k + C_i - \lambda_k \leq M \cdot \beta_{k-1} \tag{C4}$$

Formulation of the busy period $[f_{k-1}, f_k[$ when $\beta_{k-1} = 1$. If $\beta_{k-1} = 1$, the interval between f_{k-1} and f_k is a busy period with length λ'_k. The total amount of workload from higher priority tasks inside $[f_{k-1}, f_k[$ is

$$\Phi'_k := \sum_{j<i} (If_{j,k} - If_{j,k-1}) \cdot C_j$$

Thus, the length λ'_k of busy period $[f_{k-1}, f_k[$ can be represented as $\Phi'_k + C_i$ and the MILP constraint becomes

$$\forall k \quad -M \cdot (1 - \beta_{k-1}) \leq \Phi'_k + C_i - \lambda'_k \leq M \cdot (1 - \beta_{k-1}) \tag{C5}$$

Formulation of f_k by accumulating the idle time and workload. If we consider each job J_k, from $s_0 = 0$ to its finish time f_k, the time interval $[0, f_k[$ consists of multiple busy periods and processor idle times, which can be summed up as in

$$\forall k \quad \sum_{j<i} If_{j,k} \cdot C_j + k \cdot C_i + \sum_{k'<k} \iota_{k'} = f_k \tag{C6}$$

Refining the arrival time of a higher priority job before the beginning or the end of a busy period. At the beginning or the end of a level-i busy period, a higher priority task must have completed any previously requested execution time. As a result, it must be

$$\forall k, j < i \quad \alpha_j + (IL_{j,k} - 1) \cdot T_j + r_j - M \cdot \beta_{k-1} < a_k - L_k \quad \text{(C7)}$$

The latest activation time of a higher priority job (from τ_j) before the beginning of a busy period starting in $a_k - L_k$ is $\alpha_j + (IL_{j,k} - 1) \cdot T_j$. This job must complete before the start of the busy period in $a_k - L_k$ after at least r_j time units. The term $-M \cdot \beta_{k-1}$ is used in the constraint (C7) because L_k is only relevant when $\beta_{k-1} = 0$.

Likewise, at the end of a busy period

$$\forall k, j < i \quad \alpha_j + (If_{j,k} - 1) \cdot T_j + r_j < f_k \quad \text{(C8)}$$

Length of a busy period. We use BP to denote the length of longest level-i busy period, and $N_i = \lceil \frac{BP}{T_i} \rceil$ is the number of jobs of τ_i within that busy period. As long as there is a busy period that spans N_i jobs of τ_i, the total task execution within it cannot exceed BP. Therefore, $\forall 1 \le x \le K - N_i + 1$:

$$L_x + (a_{x+1} - a_x) - M \cdot (1 - \beta_x) + \sum_{x < k < x+N_i-1} (a_{k+1} - a_k - M \cdot (1 - \beta_k)) + \rho_{x+N_i-1} \le BP \quad \text{(C9)}$$

For arbitrarily N_i successive jobs inide an arbitrary problem window, we do not know if they are inside the same busy period, however, $\beta_k = 1$ is a sufficient condition for two jobs J_k and J_{k+1} to be in the same busy period (the same for $\beta_x = 1$) and this explains the big-M terms in (C9). For the scenario in Figure 1 where $BP = 11$ and $N_3 = 2$, there is a busy period $[s_0, f_2)$ that spans two jobs J_1 and J_2: $L_1 + (a_2 - a_1) + \rho_2 = 11.5 \le BP$.

4.3 Weakly hard schedulability analysis

Given an arbitrary sequence of K successive jobs of τ_i inside the problem window, the weakly hard property specifies that the maximum number of deadline misses (among these K jobs) should be bounded by m ($< K$). The total number of deadline misses can be computed as

$$\mathsf{nDmiss} := \sum_k b_k \quad \text{(C10)}$$

The number of deadline misses of τ_i within the problem window is bounded by m, *if the addition of the constraint*

$$m + 1 \le \mathsf{nDmiss} \quad \text{(C11)}$$

makes the formulation non feasible.

Another option is to use the formulation of the number of deadline misses in (C10) as an optimization (maximization) function, and check what is the maximum number of misses for a given number of activations, or even the other way around, to find what is the minimum value of K given a number of deadline misses.

5 EXTENSIONS OF THE SOLUTION MODEL

The weakly-hard analysis framework proposed in Section 4 can be easily adapted to a more general task model. In particular, to shared resources and tasks with jitter.

5.1 Shared resources

In this part, we show an extension to the case of resource sharing using the Immediate Priority Ceiling Protocol (PCP) [29] as used in the OSEK and AUTOSAR operating system standards.

A set of shared resources $\mathcal{R}_1, \ldots, \mathcal{R}_G$ are accessed by tasks in mutual exclusive mode. For any task τ_i and for any resource $\mathcal{R}_g$, $\mathcal{S}_{i,g} = \{cs_{i,g,1}, cs_{i,j,2}, \ldots\}$ is a finite **multiset** (a set that allows multiple instances of its elements) of worst case execution times for the critical sections executed by τ_i on $\mathcal{R}_g$.

The *priority ceiling* $pc(\mathcal{R}_g) := \min\{i : \mathcal{S}_{i,g} \neq \emptyset\}$ of $\mathcal{R}_g$ is defined as the highest priority of any task that accesses it. Every time a task accesses $\mathcal{R}_g$, its priority is boosted to the priority ceiling of $\mathcal{R}_g$. In this way, any job of τ_i can be blocked at most once by one lower priority job executing a critical section on a resource with priority ceiling $pc(\mathcal{R}_g) \le i$. This guarantees a predictable worst-case blocking time.

For simplicity, in the following, we will assume the Rate Monotonic (RM) system such that τ_i has a higher priority than τ_j if $T_i < T_j$ and for any τ_i, $D_i = T_i$.

An arbitrary sequence of x consecutive job activations of τ_i, can be (directly or indirectly) blocked by at most x critical section executions on resources with ceiling higher than or equal to the priority of the job.

$$\mathcal{S}_i^x := \bigcup_{pc(\mathcal{R}_g) \le i} \bigcup_{i \le j} \overbrace{\mathcal{S}_{j,g} \cup \cdots \cup \mathcal{S}_{j,g}}^{\lceil \frac{BP + x \cdot T_i}{T_j} \rceil \text{ times}}$$

Hence, for any x consecutive job windows of τ_i, the maximum blocking time is defined as the sum of the x largest elements in the multiset $\mathcal{S}_i^x$:

$$\mathcal{B}_i^x := \sum_{\text{the } x \text{ largest}} \mathcal{S}_i^x$$

To apply these blocking times to the MILP model in Section 4, we follow the common approach that adds the blocking time to the execution time when considering the possible interference.

For any $1 \le k \le K$, the real variable $c_{i,k}$ indicates the execution time which includes the blocking time that the kth job of τ_i, within the problem window, can suffer.

$$C_i \le c_{i,k} \le C_i + \mathcal{B}_i^1 \quad (13)$$

For any number of consecutive job windows, we can bound the sum of all these execution variables: $\forall 1 < x \le K \ \forall 1 \le y \le K - x + 1$

$$\sum_{y \le k \le y+x-1} c_{i,k} - x \cdot C_i \le \mathcal{B}_i^x \quad \text{(C12)}$$

To extend the original problem formulation to the case of resource sharing, all instances of C_i in constraints (C3)~(C6) should be replaced by the corresponding variable $c_{i,k}$ Also the definition of the minimum processor idle time needs to be modified and the constraint (C1) is then updated as follows

$$\mathsf{minIdle}(\mathcal{T}_i, x \cdot T_i) - \mathcal{B}_i^x \le \sum_{y \le k \le y+x-1} \iota_k \quad \text{(C1*)}$$

5.2 Jitter

The jitter [9] of a periodic task represents the maximum possible delay of the task actual activation times with respect to the ideal periodic activations. Given a periodic task $\tau_l = (C_l, D_l, T_l)$, we denote its jitter as $\mathcal{J}_l$ with $\mathcal{J}_l + C_l \leq D_l$.

Because of jitter, the distance between the activation times of two jobs J_k and J_{k+1} of the target task τ_i inside the problem window is not a fixed value T_i, but can be any value within the range $[T_i - \mathcal{J}_i, T_i + \mathcal{J}_i]$. More generally, there is $\forall 1 \leq k < K, 1 \leq N \leq K-k$,

$$N \cdot T_i - \mathcal{J}_i \leq a_{k+N} - a_k \leq N \cdot T_i + \mathcal{J}_i$$

The jitter of a higher priority task τ_j also affects its interference upon the task under analysis. For example, the number of jobs of τ_j that arrive before the finish time of the kth job of τ_i within the problem window becomes: $\lceil \frac{f_k - \alpha_j - \mathcal{J}_j}{T_j} \rceil \leq If_{j,k} \leq \lceil \frac{f_k - \alpha_j + \mathcal{J}_j}{T_j} \rceil$. This is encoded by the constraint below, as a replacement of (6).

$$\forall j < i \quad \frac{f_k - \alpha_j - \mathcal{J}_j}{T_j} \leq If_{j,k} < \frac{f_k - \alpha_j + \mathcal{J}_j}{T_j} + 1 \qquad (3^*)$$

For $IL_{j,k}$, when $\beta_k = 0$, it is now $\lceil \frac{a_k - L_k - \alpha_j - \mathcal{J}_j}{T_j} \rceil \leq IL_{j,k} \leq \lceil \frac{a_k - L_k - \alpha_j + \mathcal{J}_j}{T_j} \rceil$, and the big-$M$ constraint in (7) is updated to $\forall j < i$

$$\frac{a_k - L_k - \alpha_j - \mathcal{J}_j}{T_j} - M \cdot \beta_{k-1} \leq IL_{j,k} \qquad (4.a^*)$$

$$IL_{j,k} < \frac{a_k - L_k - \alpha_j + \mathcal{J}_j}{T_j} + 1 + M \cdot \beta_{k-1} \qquad (4.b^*)$$

To take into account jitter, several equations in the MILP formulation also need to be updated (the jitter mostly result in a modifier applied to periods). Summarizing (the full justification is omitted for space reasons), Equations (1), (2), (3), (4), (4.1), (C1) (C7), (C8) are replaced with the following

$$\forall k \quad 0 \leq L_k \leq T_i - r_i + \mathcal{J}_i \quad (17)$$

$$\forall j < i \quad 0 \leq \alpha_j \leq T_j - r_j + \mathcal{J}_j \quad (18)$$

$$\forall k \quad C_i \leq f_{k+1} - f_k \leq R_i + (T_i - r_i) + \mathcal{J}_i \quad (19)$$

$$\forall k \quad -M \cdot b_k \leq a_k + D_i - f_k - \mathcal{J}_i < M \cdot (1 - b_k) \quad (20)$$

$$\lceil \frac{R_i + T_i - r_i + \mathcal{J}_j}{T_j} \rceil \quad (21)$$

$$\mathsf{minIdle}(\mathcal{T}_i, x \cdot T_i - \mathcal{J}_i) \leq \sum_{y \leq k \leq y+x-1} \iota_k \quad (22)$$

$$\forall k, j < i \quad \alpha_j + (IL_{j,k} - 1) \cdot T_j + r_j - M \cdot \beta_{k-1} - \mathcal{J}_j < a_k - L_k \quad (23)$$

$$\forall k, j < i \quad \alpha_j + (If_{j,k} - 1) \cdot T_j + r_j - \mathcal{J}_j < f_k \quad (24)$$

6 EXPERIMENTS

In the section, we apply the proposed weakly hard schedulability analysis to an automotive engine control application and a set of randomly generated system configurations. All experiments are conducted on a machine with 8 GB memory and 8 cores: Intel(R) Xeon(R) CPU X3460 @ 2.80GHz, using CPLEX 12.6.3 as the MILP solver. The MILP formulation is encoded in C++ using the CPLEX library and is available for download[1].

[1] https://github.com/m-k-wsa/

6.1 The fuel injection case study

At first, we apply the MILP weakly hard schedulability analysis with the shared resource extension (Section 5), to the fuel injection application described in [8].

According to the AUTOSAR standard, an automotive application is composed by a set of functions called runnables, which are executed by tasks scheduled by fixed priority. The runnable-to-task mapping and the task scheduling are defined at the system integration phase.

For the fuel injection application in [8], a heuristic strategy is applied to allocate approximately 1000 runnables to tasks with 280 critical sections. The resulting taskset has 15 tasks with priorities assigned according to the Rate Monotonic rule (all times in microseconds)

Task	C_i	T_i	Task	C_i	T_i
τ_1	1015.83	$2 \cdot 10^4$	τ_8	5296.84	$1 \cdot 10^5$
τ_2	2309.5	$2 \cdot 10^4$	τ_9	325.64	$2 \cdot 10^5$
τ_3	1148.64	$2.5 \cdot 10^4$	τ_{10}	3285.24	$2 \cdot 10^5$
τ_4	2419.6	$3 \cdot 10^4$	τ_{11}	208.67	$5 \cdot 10^5$
τ_5	287.5	$5 \cdot 10^4$	τ_{12}	539.5	$5 \cdot 10^5$
τ_6	51.072	$6 \cdot 10^4$	τ_{13}	47616.3	$1 \cdot 10^6$
τ_7	2318.42	$1 \cdot 10^5$	τ_{14}	799006	$2 \cdot 10^6$
			τ_{15}	$1.0005 \cdot 10^6$	$1 \cdot 10^7$

Table 3: An automotive case study

Due to the blocking from τ_{15}, τ_{14} is not (hard real-time) schedulable.

To verify the weakly hard schedulability property, we tested a series of m-K parameters: $\{(1,5),(2,5),(2,10),(3,10),(3,15),(4,15)\}$. According to our weakly hard schedulability analysis, it is guaranteed that there will be at most $m = 2$ (resp. 3 and 4) deadline misses out of any $K = 5$ (resp. 10 and 15) consecutive jobs of τ_{14}.

Regarding the runtime cost, except for the case $m = 3$ and $K = 15$, all tests complete within 2 minutes. It takes the CPLEX solver almost 30 minutes to make a decision when $m = 3$ and $K = 15$.

6.2 Runtime performance

In this subsection, we apply the weakly hard real-time analysis in Section 4 to a set of randomly generated tasksets for an empirical evaluation of the runtime performance, with a large variety of configurations: $n \in \{10 \sim 15, 20, 30, 50\}$, $U \in \{0.8, 0.85, 0.9, 0.95\}$, $m \in \{1, 2, 3\}$ and $K \in \{5, 10 \sim 15, 20\}$. Each configuration in the experiment is specified by a tuple (n, U, m, K), where n is the taskset size, U is the taskset utilization, and m-K is the weakly hard property to be checked.

Overall, 6253 task systems are tested. For each taskset with a pair n and U: 1) the utilization U_i of tasks is generated using the Randfixedsum algorithm in [15]; 2) the task period T_i is uniformly sampled in the range $[10, 1000]$; each task has an implicit deadline, i.e., $D_i = T_i$; 3) and each task WCET is computed as $C_i = T_i \cdot U_i$.

Tasks are assigned priorities according to the Rate Monotonic rule. If the lowest priority task τ_n in the taskset is schedulable, the taskset is abandoned; otherwise, we proceed with the weakly hard

real-time analysis on τ_n. This configuration is designed to stress the weakly hard analysis, since even if the lowest priority task τ_n is schedulable there may exist other non-schedulable tasks with a smaller number of interfering higher priority tasks for the m-K analysis.

In the analysis of each taskset we defined a runtime limit of 1800 seconds: if the analysis takes more than 1800 seconds without terminating, we stop and report a failure of the m-K analysis.

Deadline misses in a row. The m-K model discussed so far concerns the upper bound on the number of deadline misses (m) out of any K consecutive task activations. Another popular pattern for weakly hard schedulability analysis is to check if there are **more than m deadline misses in a row**, which is equivalent to analyze the m-K model with $K = m + 1$.

In the following, we evaluate the number of cases in which there are $K = 2$ and $K = 3$ consecutive deadline misses, when $U = 0.95$ and $n \in \{10, 20, 30, 50\}$. Results are shown in Table 4 and for every test case, the MILP solver returns its decision in less than a minute. Consecutive deadline misses seldom happen even when the total utilization is as high as 0.95. Another observation is that the fraction of cases with consecutive deadline misses is not sensitive with respect to the number of tasks in the set.

	$K = 2$	$K = 3$		$K = 2$	$K = 3$
$n = 10$	80.3%	98.6%	$n = 30$	85.1%	99.9%
$n = 20$	84.8%	99.6%	$n = 50$	84.9%	99.9%

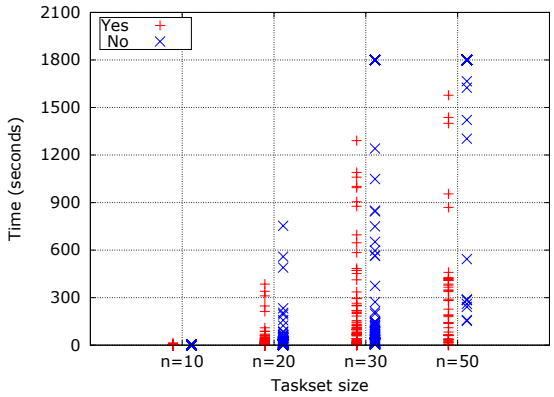
Table 4: Percentage of sets with K consecutive deadline misses

Varying the taskset size. Table 5 and Figure 3 show the experimental results with a variable taskset size, when $U = 0.85$. The weakly hard analysis confirms that a large portion of these non-schedulable tasks will never miss more than 1 deadline out of any 5 its consecutive activations. For example, when the taskset size is 20 or 30, the percentage of *1-5* feasible sets is around 50%. When m is increased to 2, more than 90% of the tested tasksets satisfy the specified m-K property in all cases.

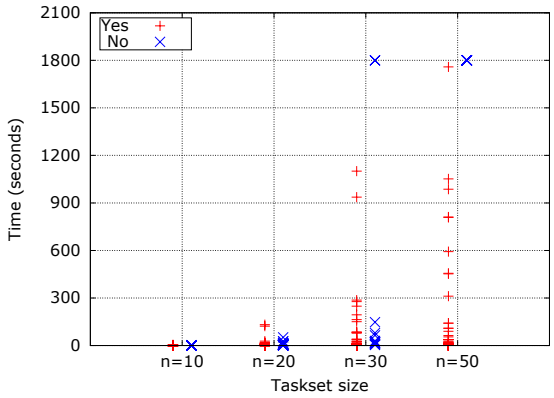
On the other side, when the taskset size is very large ($n = 50$), for $m = 1$ a significant amount of tests exceed the runtime limit of 1800 seconds, which implies that a longer runtime is needed for such cases. Figure 3 depicts the time spent on the weakly hard analysis of each taskset: *Yes* labels that the corresponding m-K property is verified (*No* in the other case). The majority of analyses return the decision within 10 minutes.

	$m = 1, K = 5$		$m = 2, K = 5$	
	confirmed	n/a	confirmed	n/a
$n = 10$	42.8%	0%	90.6%	0%
$n = 20$	49.0%	0%	91.9%	0%
$n = 30$	54.4%	6.5%	92.9%	1.2%
$n = 50$	42.7%	41.9%	94.0%	6.0%

Table 5: Experiments that confirm the m-K property and run out of time limit (n/a) with variable n



(a) $m = 1$



(b) $m = 2$

Figure 3: Runtime results for $K = 5$

	$m = 2$	$m = 3$		$m = 2$	$m = 3$
$K = 11$	54.4%	84.8%	$K = 14$	47.0%	73.6%
$K = 12$	51.3%	81.5%	$K = 15$	45.8%	66.4%
$K = 13$	49.2%	76.5%	$K = 20$	33.6%	56.1%

Table 6: Percentage of valid m-K property with variable K

Varying the problem window size. Table 6 contains the experiment results when varying the problem window size K, with $n = 10$ and $U = 0.85$. The problem window size K is a dominant factor with respect to the complexity of the analysis. Still, the results are promising and for more than one third of the tasksets, the number of deadline misses is bounded by at most $m = 2$.

	$m = 2, K = 10$	$m = 3, K = 10$
$U = 0.80$	90.8%	99.1%
$U = 0.85$	60.8%	86.1%
$U = 0.90$	30.2%	50.6%
$U = 0.95$	6.8%	17.6%

Table 7: Percentage of valid m-K property with variable U

Varying taskset utilization. Table 7 shows the percentage of tasksets that satisfy the m-K property with variable taskset utilization levels and a fixed taskset size $n = 10$.

Even when the taskset utilization is very high ($U = 0.90$), more than 30% of the non-schedulable tasks will not miss more than $m = 2$ deadlines within any sequence of $K = 10$ successive task activations. If we further increase m to 3, the tasksets satisfying the weakly hard property become half of the generated sets.

7 CONCLUSIONS

In this work, we propose a weakly hard schedulability analysis technique for fixed priority preemptive scheduling of a set of periodic tasks. Our approach applies to offset-free systems and is more general than previous works. The analysis is formulated as an MILP encoding, and its performance (regarding both the analysis precision and runtime efficiency) have been confirmed by extensive experiments.

A possible extension of the approach proposed in this paper could target multiprocessor systems with partitioned scheduling and shared resources as applicable to several automotive applications [1, 32].

REFERENCES

[1] Zaid Al-bayati, Youcheng Sun, Haibo Zeng, Marco Di Natale, Qi Zhu, and Brett Meyer. 2015. Task placement and selection of data consistency mechanisms for real-time multicore applications. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*. 172–181.

[2] Amir Aminifar, Petru Eles, Zebo Peng, and Anton Cervin. 2013. Control-quality driven design of cyber-physical systems with robustness guarantees. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 1093–1098.

[3] Amir Aminifar, Soheil Samii, Petru Eles, Zebo Peng, and Anton Cervin. 2012. Designing high-quality embedded control systems with guaranteed stability. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd.* 283–292.

[4] Karl-Erik Årzén, Anton Cervin, Johan Eker, and Lui Sha. 2000. An introduction to control and scheduling co-design. In *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, Vol. 5. 4865–4870.

[5] Sanjoy Baruah and Alan Burns. 2006. Sustainable scheduling analysis. In *2006 27th IEEE International Real-Time Systems Symposium (RTSS'06)*. 159–168.

[6] Guillem Bernat, Alan Burns, and Albert Liamosi. 2001. Weakly hard real-time systems. *Computers, IEEE Transactions on* 50, 4 (2001), 308–321.

[7] Enrico Bini, Marco Di Natale, and Giorgio Buttazzo. 2008. Sensitivity analysis for fixed-priority real-time systems. *Real-Time Systems* 39, 1-3 (2008), 5–30.

[8] Alessandro Biondi, Marco Di Natale, Youcheng Sun, and Stefania Botta. 2016. Moving from single-core to multicore: initial findings on a fuel injection case study. (2016).

[9] Reinder J Bril, Johan J Lukkien, and Rudolf H Mak. 2013. Best-case response times and jitter analysis of real-time tasks with arbitrary deadlines. In *Proceedings of the 21st International conference on Real-Time Networks and Systems*. ACM, 193–202.

[10] Tobias Bund and Frank Slomka. 2014. Controller/platform co-design of networked control systems based on density functions. In *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*. 11–14.

[11] Tobias Bund and Frank Slomka. 2015. Worst-case performance validation of safety-critical control systems with dropped samples. In *Proceedings of the 23rd International Conference on Real Time and Networks Systems*. 319–326.

[12] Giorgio Buttazzo. 2011. *Hard real-time computing systems: predictable scheduling algorithms and applications*. Vol. 24. Springer Science & Business Media.

[13] Robert I Davis, Ken W Tindell, and Alan Burns. 1993. Scheduling slack time in fixed priority pre-emptive systems. In *Real-Time Systems Symposium, 1993., Proceedings*. IEEE, 222–231.

[14] J.L. Diaz, D. F. Garcıa, K. Kim, C.-G. Lee, L. Lo Bello, J. M. Lopez, S. L. Min, and O. Mirabella. 2002. Stochastic analysis of periodic real-time systems. In *22nd IEEE Real-Time Systems Symposium., Austin, TX. USA.*

[15] Paul Emberson, Roger Stafford, and Robert I Davis. 2010. Techniques for the synthesis of multiprocessor tasksets. In *proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010)*. 6–11.

[16] Goran Frehse, Arne Hamann, Sophie Quinton, and Matthias Woehrle. 2014. Formal analysis of timing effects on closed-loop properties of control software. In *Real-Time Systems Symposium (RTSS), 2014 IEEE*. 53–62.

[17] Dip Goswami, Reinhard Schneider, and Samarjit Chakraborty. 2011. Co-design of cyber-physical systems via controllers with flexible delay constraints. In *Proceedings of the 16th Asia and South Pacific Design Automation Conference*. IEEE Press, 225–230.

[18] Zain AH Hammadeh, Sophie Quinton, and Rolf Ernst. 2014. Extending typical worst-case analysis using response-time dependencies to bound deadline misses. In *Proceedings of the 14th International Conference on Embedded Software*. ACM, 10.

[19] Jung-Eun Kim, Tarek F. Abdelzaher, and Lui Sha. 2015. Budgeted generalized rate monotonic analysis for the partitioned, yet globally scheduled uniprocessor model. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium, Seattle, WA, USA, April 13-16, 2015*. 221–231.

[20] Pranaw Kumar and Lothar Thiele. 2012. Quantifying the effect of rare timing events with settling-time and overshoot. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd.* 149–160.

[21] John P Lehoczky. 1990. Fixed Priority Scheduling of Periodic Task Sets with Arbitrary Deadlines.. In *RTSS*, Vol. 90. 201–209.

[22] Joseph Y-T Leung and Jennifer Whitehead. 1982. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance evaluation* 2, 4 (1982), 237–250.

[23] Chung Laung Liu and James W Layland. 1973. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM (JACM)* 20, 1 (1973), 46–61.

[24] M. M Hamdaoui and P. Ramanathan. 1995. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. In *IEEE Transactions on Computers*.

[25] Sophie Quinton, Matthias Hanke, and Rolf Ernst. 2012. Formal analysis of sporadic overload in real-time systems. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 515–520.

[26] Parameswaran Ramanathan. 1999. Overload management in real-time control applications using (m, k)-firm guarantee. *IEEE Transactions on Parallel and Distributed Systems* 10, 6 (1999), 549–559.

[27] Ola Redell and Martin Sanfridson. 2002. Exact best-case response time analysis of fixed priority scheduled tasks. In *Real-Time Systems, 2002. Proceedings. 14th Euromicro Conference on*. IEEE, 165–172.

[28] Lui Sha, Tarek Abdelzaher, Karl-Erik Årzén, Anton Cervin, Theodore Baker, Alan Burns, Giorgio Buttazzo, Marco Caccamo, John Lehoczky, and Aloysius K Mok. 2004. Real time scheduling theory: A historical perspective. *Real-time systems* 28, 2-3 (2004), 101–155.

[29] Lui Sha, Ragunathan Rajkumar, and John P Lehoczky. 1990. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Transactions on computers* 39, 9 (1990), 1175–1185.

[30] Damoon Soudbakhsh, Linh TX Phan, Anuradha M Annaswamy, and Oleg Sokolsky. 2016. Co-design of arbitrated network control systems with overrun strategies. *IEEE Transactions on Control of Network Systems* (2016).

[31] Damoon Soudbakhsh, Linh TX Phan, Oleg Sokolsky, Insup Lee, and Anuradha Annaswamy. 2013. Co-design of control and platform with dropped signals. In *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*. 129–140.

[32] Alexander Wieder and Björn B Brandenburg. 2013. Efficient partitioning of sporadic real-time tasks with shared resources and spin locks. In *Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on*. 49–58.

[33] Wenbo Xu, Zain AH Hammadeh, Alexander Kroller, Rolf Ernst, and Sophie Quinton. 2015. Improved Deadline Miss Models for Real-Time Systems Using Typical Worst-Case Analysis. In *Real-Time Systems (ECRTS), 2015 27th Euromicro Conference on*. IEEE, 247–256.

[34] Yang Xu, Karl-Erik Årzén, Enrico Bini, and Anton Cervin. 2014. Response time driven design of control systems. *IFAC Proceedings Volumes* 47, 3 (2014), 6098–6104.

[35] Yang Xu, Karl-Erik Årzén, Anton Cervin, Enrico Bini, and Bogdan Tanasa. 2015. Exploiting job response-time information in the co-design of real-time control systems. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2015 IEEE 21st International Conference on*. 247–256.

A RESPONSES TO THE REVIEWERS

A.1 Reviewer 1

1 Reviewer's comment:

> *Due to many assumptions made in such an approach (e.g. schedulability overhead is assumed to be minimal or zero, and even just assuming that one can give reasonable worst case execution times is not clear), the applicability in industry is not immediately clear. Also, it might be worth comparing to probabilistic methods for schedulability analysis.*

We added a reference stating that probabilistic methods culd in principle be used for checking the probability of m-k misses. However, to our knowledge probabilistic methods typically assume knowledge of the activation offsets and are even less scalable than our proposed method. In addition, while it is true that overheads are not included, this is true for most papers in which these overheads are simply assumed ot be negligible with respect to the task execution times.

---

2 Reviewer's comment:

> *Despite the many equations and formalisms, it is not difficult to follow the analysis described in the work. The paper also includes a well-written summary of schedulability analysis methods proposed in the past.*
>
> *I could not verify the experiments since the website was anonymized. With blind reviews, the authors might consider hosting the code in a temporary location (e.g. a github repository) that cannot be traced back to the authors, to allow reviewers access.*

A repository has been made available for the code

---

3 Reviewer's comment:

> *Some formulations could be improved. For instance, the following is incomplete: "if j_k is not interfered by its predecessor". To be correct, it should read "if j_k is not interfered with by its predecessor". However, the use of the active voice might improve readability.*

Fixed.

---

4 Reviewer's comment:

> *Minor issues: "inidicates" → "indicates" "elemnts" → "elements"*

Fixed.

A.2 Reviewer 2

5 Reviewer's comment:

> *The paper is not easy to read, in particular the MILP formulation description could be improved with some examples and a more simplified expression (no use of big-M constant for instance). The experiments are nice.*

An example system, as in Figrue 1, is added to the paper for helping readers understand the MILP formulation. As suggested, big-M constraints are now annotated with high level operators.

---

6 Reviewer's comment:

> *Remarks:*
>
> *The considered model is (m,k)-firm model, why not referring to the seminal work [HR95]?*
>
> *[HR95] A dynamic priority assignment technique for streams with (m, k)-firm deadlines. M Hamdaoui, P Ramanathan - IEEE transactions on Computers 1995*

Added.

---

7 Reviewer's comment:

> *job window = $[a_k, a_{k+1}[$; what are a busy period? a problem window $[0, f_k[$?*

We clarified these concepts in the revision, and concrete examples are given.

In Section 3 (The System Model): "A level-i busy period *is defined as a time interval during which the processor is always occupied by the execution of tasks with priority higher than or equal to τ_i. For example, in Figure 1, $[s_0, f_2)$ and $[a_3, f_3)$ are two level-3 busy periods: Because the focus of this paper is not the single task WCRT, our definition of a busy period does not strictly follow its original meaning in [21], where a busy period corresponds to the maximal level-i busy period in this paper."*

In Section 3.1 (The weakly hard model): "The time interval $[a_k, a_{k+1})$ is called the kth job window *of τ_i, and the* problem window *for the analysis starts from a time instant s_0 (let us say $s_0 = 0$), which is the earliest time instant such that from s_0 to a_1 the processor is fully occupied by the execution of higher priority tasks, till the finish of the Kth job, i.e., $[s_0, f_K)$."*

---

8 Reviewer's comment:

> *The problem minimizes the number of deadline misses of τ_i for all $\alpha_j \leq T_j - r_j$. Why not $\alpha_j < T_j$?*

In the weakly hard analysis, "An arbitrary sequence of K successive activations of τ_i is considered, with the objective of checking whether there are more than m deadline misses for τ_i in this sequence." Alternatively, "Another option is to use the formulation of the number of deadline misses in (C10) *as an optimization (maximization) function, and check what is the maximum number of misses for a given number of activations"*

Regarding $\alpha_j \leq T_j - r_j$, "Assume that the first job $J_{j,1}$ of τ_j in the window arrives at time $s_0 + T_j - r_j + \epsilon$ with $\epsilon > 0$. This implies the precedent job (let us say) $J_{j,0}$ is activated at time $s_0 - r_j + \epsilon$. Because any job of τ_j needs at least r_j time to finish, $J_{j,0}$ will be still active at time instant s_0, which contradicts the fact that s_0 is the earliest time instant that from s_0 to a_1 the processor is fully occupied by higher priority task execution. This small proof explains why the upper bound for α_j is $T_j - r_j$. "

---

9 Reviewer's comment:

> *big-M method is classic to encode high level operators such as $\Longrightarrow$. But for the sake of readability, it is better to give the constraints with high level operators. e.g. (1) $b_k = 0$ <=> f_k <= $a_k + D_i$ (2) $\beta_k = 0$ <=> f_k <= a_{k+1} (3) $l_k > 0$ => $\beta_k = 0$*

Added.

---

10 Reviewer's comment:

Again, for the sake of readability: first explaining the interfering constraint with integer and then with real would help. Indeed, these are two distinct questions: 1. how to model the schedulability problem; 2. how to implement it efficiently with efficient solver (for instance by using real).

There are two integer to real conversions in the paper. As suggested by the reviewer, we added the "high level operator" to improve the readability. In the end, they are rather straightforward.

- $If_{j,k} = \lceil \frac{f_k - \alpha_j}{T_j} \rceil$ *is linearized as* $0 \leq If_{j,k} - \frac{f_k - \alpha_j}{T_j} < 1$.
- $\beta_{k-1} = 0 \Rightarrow IL_{j,k} = \lceil \frac{a_k - L_k - \alpha_j}{T_j} \rceil$ *is linearized as* $\beta_{k-1} = 0 \Rightarrow 0 \leq IL_{j,k} - \frac{a_k - L_k - \alpha_j}{T_j} < 1$ *The reason for* $\beta_{k-1} = 0$ *is that "that we only use the value of* L_k*, and thus the interval* $[0, a_k - L_k)$*, when* $\beta_{k-1} = 0$."

---

11 Reviewer's comment:

An example also would greatly help.

An example system with 3 tasks is added as in Figure 1 and Table 2. This simple 3-task system is also used for the illustration of linear constraints at each necessary step of the paper.

---

12 Reviewer's comment:

If $\tau = (C, T, P)$*, P being the priority and D=T (in this case* $\beta_k = b_k$*).* $\tau_1 = (4, 80, 2)$*,* $\tau_2 = (5, 20, 3)$*,* $\tau_3 = (15, 40, 1)$ *If all* $\alpha = 0$ *and if we analyze* τ_2*: problem window* $[0, 40]$*,* $b_0 = 1$ *and* $b_1 = 1$*;* $L_1 = 4$*;* $If_{1,0} = 1, If_{1,1} = 1, IL_{1,0} = 1, IL_{1,1} = 0$.

Then I don't understand "For simplicity, when $\beta_{k-1} = 1$*, we enforce* $IL_{j,k} = If_{j,k-1}$*" because in my example above,* $IL_{1,1}$ *would be set to 1.*

"**Throughout our analysis, we will only use the value of** L_k**, when** $f_{k-1} \leq a_k$**.**"

"We remind that we only use the value of L_k, and thus the interval $[0, a_k - L_k)$, when $\beta_{k-1} = 0$."

"For the particular scenario in Figrue 1, $L_1 = 0.5$ and $L_3 = 0$. Because J_1 does interfere the execution of J_2, L_2 is not relevant to our analysis and its value can be any number constrained."

As a result, in your example, $b_0 = 1$. In this case the value of L_1 is not even used in our analysis and of course it has no effect on the value of $IL_{1,1}$. In other words, the L_1 and $IL_{1,1}$ can be any value in this case. "For simplicity, when $\beta_{k-1} = 1$, we force $IL_{j,k} = If_{j,k-1}$"

Table 2 summarizes all these counting values based on the example system.

---

13 Reviewer's comment:

Continuing with the example above, if now $\alpha_1 = 19$*,* $\alpha_3 = 19$*, the behaviour will be very similar but for* J_2 *(with -1 because 19 and not 20) What do we gain be exploring all offsets? compared to the critical instant? an example would be good. Does the solver provide the optimal offsets of tasks to minimize the deadline misses?*

Figure 1 shows a problem window, where 2 out of its 3 jobs miss the deadline. However, if we consider all tasks simultaneously activated (i.e., the critical instant), only 1 out of 3 jobs miss the deadline.

The solver checks if there will be more than m deadline misses out of K job activations. If there is, the solver will provide tasks' offsets (w.r.t. s_0*) that lead to the situation such that more than m deadline misses happen in the problem window.*

A.3 Reviewer 3

14 Reviewer's comment:

Points in favor: • Weakly-hard schedulability analysis is an interesting topic and it needs more efforts to be more mature. • The schedulability analysis of offset-free periodic tasks in weakly-hard real-time system is not addressed in existing work. • The paper carefully addressed all related work to their contribution. • Successfully, the schedulability problem of weakly-hard real-time systems was formulated as MILP. • The paper is provided with an empirical evaluation with a variety of configurations: number of tasks, utilization, m and K.

---

15 Reviewer's comment:

Points against • The paper is hard to follow.

In the revision, we focus on improving the readability of the paper. Especially,

- *we formalize key concepts in Section 3 (The System Model) and Section 3.1 (The weakly hard model)*
- *an example system, as in Figure 1, is added*
  - *to demonstrate that the critical instant does not lead to maximum deadline misses in the problem window*
  - *to illustrate key concepts on weakly hard analysis*
  - *to help explain linear constraints at each necessary step*
- *high level operators are used to annotate these big-M constraints*

---

16 Reviewer's comment:

• No formal definitions.

We formalize relevant definitions in Section 3, where Section 3.1 speciflizes concepts for the weakly hard analysis.

---

17 Reviewer's comment:

• A short section to explain the proposed approach to compute m is missing. The proposed approach is shown implicitly through the MILP.

In the weakly hard analysis, "An arbitrary sequence of K successive activations of τ_i *is considered, with the objective of checking whether there are more than m deadline misses for* τ_i *in this sequence." Alternatively, "Another option is to use the formulation of the number of deadline misses in* (C10) *as an optimization (maximization) function, and check what is the maximum number of misses for a given number of activations"*

Each linear constraint in this paper corresponds to an inherent property for jobs (of τ_i *or higher priority tasks) within the problem window. Some of our constraints even relax such properties, for example, we use real-valued variables to count the number of interfering higher priority jobs. This means our linear constraints model accepts more behaviours than what can indeed happen inside the problem window and thus as long as there is a particular scenario such that more than m deadline misses happen in the problem window, this*

scenario must also exist inside the developed linear constraints model. In other words,

"The developed MILP model serves as an over-approximate analysis, that is, if our weakly hard analysis confirms the m-K property, it is guaranteed that there will be no more than m deadline misses out of any K successive job activations (of the target task), however, if the m-K property is not confirmed, we cannot conclude the opposite."

---

18 Reviewer's comment:

> *• The complexity is high. The proposed analysis is applicable for relatively small $K <= 20$. Large K appears more interesting (K=1000) for applications like image processing but, there the proposed analysis is not applicable.*

Image processing applications are outside the scope of this paper, which mostly refers to control (such as engine control) applications. The number of tasks in engine control applications (see the ECRTS WATERS challenge, for example) is easily within the applicability range of our method. However, the method may be still applied indirectly (with some degree of pessimism) to large K. If we first conduct the weakly hard analysis with a smaller window (let us say $K' = 20$) and find that at most (let us say) m' jobs miss their deadline. Then, for a window of $K = 1000$ jobs, the number of deadline misses will be bounded by $50 \cdot m'$.

---

19 Reviewer's comment:

> *Other points to be covered: • In the abstract, authors claim that their analysis is "very general", but in the system model we see that the analysis is restricted to periodic tasks (no jitter, no sporadic), constrained deadlines and independent tasks.*

Our solution focuses on offset-free periodic task systems, and it can be applied to the shared resource problem and tasks with jitter (as added in this revised version). Regarding existing works on periodic task systems, tasks' initial offsets must be known and our solution is thus more general.

For the weakly hard analysis of the sporadic case, as we state mention in Section 2 (State of the Art), it is addressed by the line of works based on typical worst-case analysis. *Meanwhile, "This line of works and the method in this paper are orthogonal and the two extend the weakly hard real-time system analysis at different directions. Combined use of the two (in the future) can make the safety analysis and system design more comprehensive."*

---

20 Reviewer's comment:

> *•Response time is not considered to be a variable, how do you compute it? R_i is used to bound the finish time f_i (which is a continuous variable).*

We added a paragraph to clarify this point. We assume standard worst case response time analysis (as in Lehoczky's paper [21]) is performed in advance to determine the WCRT R_i of each task τ_i.

Once this is done, R_i becomes a parameter (a known value)

---

21 Reviewer's comment:

> *• The problem window is presented to be $[s_0, a_K + T_i)$. But J_K will not finish before f_K, the problem window should be then $[s_0, f_K)$. The problem window should be formally defined.*

We added a formal definition in Section 3.1: "the problem window *for the analysis starts from a time instant s_0 (let us say $s_0 = 0$), which is the earliest time instant such that from s_0 to a_1 the processor is fully occupied by the execution of higher priority tasks, till the finish of the Kth job, i.e., $[s_0, f_K)$." An example is also given afterwards.*

---

22 Reviewer's comment:

> *O1, O2 are not clear. How do you know in advance that a job is schedulable or not?*

We added concrete examples to help understand.

O1 and O2 aim to show that, in order to maximize the number of deadline misses in the window, we should start the analysis from a non-schedulable job J, but the job J' just before J must be schedulable. Given any task activation pattern (let us say Scenario A) that does respect O1 or O2, we can always find a Scenario B that satisfies O1 and O2, and the number of deadline misses in Scenario B does not decrease (w.r.t Scenario A).

---

23 Reviewer's comment:

> *• C9 seems to be true only for non schedulable tasks ($N_i > 1$).*

No. (C9) *always holds. $N_i = 1$ means that even the longest level-i busy period only contains one job of τ_i. Thus, every job inside the problem window will finish before the next job release, which implies that $\beta_x = 0$ in* (C9)*, and the big-M term $-M \cdot (1 - \beta_x)$ will guarantee the right side of* (C9) *to be a small enough value.*

As a matter of fact, an implicit pre-condition for the weakly hard analysis is that a task is not schedulable, otherwise, there will be no deadline miss in any case. As stated in the paper "schedulable tasks can be ruled out by simply performing a traditional hard schedulability test in advance".

---

24 Reviewer's comment:

> *I think in C9: $-M.(\beta_x - 1)$ is not needed. For fixed priority preemptive with $D_i <= T_i$ it is simple to prove that $\beta_x = 0$ is true only for $x = N_i$ where $x \in 1, ..., N_i$, which means that $\beta_1 = 1$ and the third term is not needed. Furthermore, the counter in the fourth term already excludes the last activation in the busy period and its response time is added in the fifth term in C9.*

"For arbitrarily N_i successive jobs inside an arbitrary problem window, we do not know if they are inside the same busy period, however, $\beta_k = 1$ is a sufficient condition for two jobs J_k and J_{k+1} to be in the same busy period (the same for $\beta_x = 1$) and this explains these big-M terms in (C9)*."*

---

25 Reviewer's comment:

> *As far as I understood the analysis, K (problem window) is the bottleneck of the proposed analysis which means the largest the K the highest the complexity. It is better to state that clearly in the paper.*

Added. "The problem window size K is a dominant factor with respect to the complexity of the analysis."

---

26 Reviewer's comment:

Every higher priority task will be represented in the MILP with 3 variables. Increasing the number of tasks in the system will increase the complexity exponentially. It is better to state that clearly in the paper.

The increase is not exponential, but depends on many factors, such as the ratio between the maximum and minimum period (which determines the lenght of the time interval and the number of jobs to be considered). All variables in the MILP are summarized in Table 1. there are at most variables with 3 indexes (for which their number increases in the worst case with the third power with respect to the number of entities on each index). Then, it is of course well known that each and every MILP is in essence of exponential complexity wrt the number of binary variables.

---

27 Reviewer's comment:

In section 4.3: "should bounded by m(< K)". It is incorrect, it should be "<= K". The authors, however, did not prove that in the paper.

*In the weakly hard analysis, "An arbitrary sequence of K successive activations of τ_i is considered, with the objective of checking whether there are more (note **strictly more**) than m deadline misses for τ_i in this sequence."*

In case $m = K$, the answer is surely no.

Alternatively, "Another option is to use the formulation of the number of deadline misses in (C10) *as an optimization (maximization) function, and check what is the maximum number of misses for a given number of activations"*

Each linear constraint in this paper corresponds to an inherent property for jobs (of τ_i or higher priority tasks) within the problem window. Some of our constraints even relax such properties, for example, we use real-valued variables to count the number of interfering higher priority jobs. This means our linear constraints model accepts more behaviours than what can indeed happen inside the problem window and thus as long as there is a particular scenario such that more than m deadline misses happen in the problem window, this scenario must also exist inside the developed linear constraints model. In other words,

"The developed MILP model serves as an over-approximate analysis, that is, if our weakly hard analysis confirms the m-K property, it is guaranteed that there will be no more than m deadline misses out of any K successive job activations (of the target task), however, if the m-K property is not confirmed, we cannot conclude the opposite."

---

28 Reviewer's comment:

The MILP is not illustrated in the experiments, I think the synthetic task set in section 6.2 should be shown as an illustrative example.

We added the example system as in Figure 1 to illustrate constraints at each necessary step.

---

29 Reviewer's comment:

What do you mean by "is very general and it can be easily adapted to a more general task model."? Does it mean that the proposed analysis can be adapted easily to periodic tasks with jitter, sporadic tasks and/or dependent tasks?

Our solution focuses on offset-free periodic task systems, and it can be applied to the shared resource problem and the task model with jitters. Regarding existing works on periodic task systems, tasks' initial offsets must be known and our solution is thus more general.

For the weakly hard analysis of the sporadic case, as we state in Section 2 (State of the Art), it is addressed by the line of works based on typical worst-case analysis. *Meanwhile, "This line of works and the method in this paper are orthogonal and the two extend the weakly hard real-time system analysis at different directions. Combined use of the two (in the future) can make the safety analysis and system design more comprehensive."*

---

30 Reviewer's comment:

How many task sets have been generated? Did you repeat the experiments to guarantee that the results are confident. That should be clarified.

Added. "Overall, 6253 task systems are tested."

---

31 Reviewer's comment:

Which motivations are behind the experiments? Why are you interested in the "Deadline misses in a raw"?

From the authors' perspective, we want to demonstrate that even the proposed analysis is an over-appxorimation, it can return "confirmed" results in many (if not most) cases within reasonable time, which is 1800 seconds.

"The developed MILP model serves as an over-approximate analysis, that is, if our weakly hard analysis confirms the m-K property, it is guaranteed that there will be no more than m deadline misses out of any K successive job activations (of the target task), however, if the m-K proeprty is not confirmed, we cannot conclude the opposite."

"Deadline misses in a raw" is a special case of m-K model such that $m = K - 1$, that is, are there more than $K - 1$ deadline misses out of the K jobs?

---

32 Reviewer's comment:

In the experiment of "Deadline misses in a raw", for K=2 , n=10 the 19.7% are schedulable or $m > 1$?

As stated above, "Deadline misses in a raw" is a special case of m-K model such that $m = K-1$, that is, are there more than $K-1$ deadline misses out of the K jobs? Thus, when $K = 2$, there is $m = 1$.

Table 4 reports the statistics of "No K deadline misses in a row". Thus, 19.7% is percentage of the tested tasksets that there exist 2 deadline misses in a row. "These results reveal the fact that several deadline misses in a row rarely happen"

---

33 Reviewer's comment:

Figure 3 is more interesting than the tables 4,5,6,7, it tells the reader about the scalability.

Implicitly, we set up the runtime threshold of the solver to 1800 seconds and they apply to all experiments in the paper. This threshold is already very realistic for many (if not most) temporal safety analysis senarios in avionics and automobiles.

---

34 Reviewer's comment:

Showing the "n/a" in Table 5 is more interesting than showing the "confirmed".

In my opinion, the majority of the experiments shown in 6.3 are not interesting, they do not test the proposed analysis.

From the authors' perspective, we want to demonstrate that even the proposed analysis is an over-appxorimation, it can return "confirmed" results in many (if not most) cases within reasonable time, which is 1800 seconds.

"The developed MILP model serves as an over-approximate analysis, that is, if our weakly hard analysis confirms the m-K property, it is guaranteed that there will be no more than m deadline misses out of any K successive job activations (of the target task), however, if the m-K proeprty is not confirmed, we cannot conclude the opposite."

---

35 Reviewer's comment:

In the experiments, it is interesting to: compute the "cut-off" values of utilization, task set size and K where the run time limit will be exceeded. show how much the computed m is overestimated.

It is extremely hard to conclude such "cut-off" values for each individual parameter. There are systems with very high utilization that can be anlyzed easily, and a bigger K does not necessarily lead to higher runtime. As a result, the emphasis of experiments is to demonstrate the applicability of the proposed method through a variety of combinations of n, U, m and K (subject to the runtime upper bound 1800 seconds).

Regarding the overestimation of m, due to the lack of (exact or over-approximate) solutions in literaute for weakly hard analysis of offset-free periodic task systems, and it makes unclear how we can evaluate the overestimation.

A.4 Reviewer 4

---

36 Reviewer's comment:

The paper proposes a schedulability analysis for weakly hard real-time systems (systems for which a bounded number of deadline misses out of a sequence of task activations can be tolerated). The authors consider uniprocessor systems running periodic task sets with implicit deadlines according to a fixed priority preemptive policy. The specificity of the model they address is that task offsets are unknown. The schedulability problem is formulated as a mixed integer linear programming (MILP) problem which constitutes the main part of the paper. An implementation of the solution, for which the source code is available, is evaluated on a case study and a set of randomly generated task sets.

---

37 Reviewer's comment:

The problem addressed in this paper appears to be relevant and it is well motivated by the authors. Unfortunately the paper does not address the question of whether the MILP formulation is sound or exact, in the sense that it cannot provide false positives or false negatives.

In the revision, we added explicitly that, "The developed MILP model serves as an over-approximate analysis, that is, if our weakly hard analysis confirms the m-K property, it is guaranteed that there will be no more than m deadline misses out of any K successive job activations (of the target task), however, if the m-K proeprty is not confirmed, we cannot conclude the opposite."

The correctness of the solution is implicitly verified by the linear constraints model developed in this paper. Each linear constraint in this paper corresponds to an inherent property for jobs (of τ_i or higher priority tasks) within the problem window. Some of our constraints even relax such properties, for example, we use real-valued variables to count the number of interfering higher priority jobs. This means our linear constraints model accepts more behaviours than what can indeed happen inside the problem window and thus as long as there is a particular scenario such that more than m deadline misses happen in the problem window, this scenario must also exist inside the developed linear constraints model. In other words,

---

38 Reviewer's comment:

There are very few figures and no example to help the reader understand the numerous notations which are introduced. In addition, the core part of the paper consists of a long list of constraints but there is no overall discussion to argue the correctness and precision of the proposed analysis. Is your analysis exact or not?

An example system, as in Figure 1, is added

- *to demonstrate that the critical instant does not lead to maximum deadline misses in the problem window*
- *to illustrate key concepts on weakly hard analysis*
- *to help explain linear constraints at each necessary step*

Yes, the core part of the paper is the MILP formulation, which consists of a list of linear constraints.

As stated above for Comment 37, the MILP formulation serves as a sound over-approximation of the exact weakly hard model. Its correctness can be implicitly verified by the linear constraints developed. Its precision is examined through experiments with a variety of configurations.

---

39 Reviewer's comment:

An example illustrating that activating all tasks at the same time at the beginning of the problem window does not necessarily yield the maximum number of deadline misses in the window would improve a lot the motivation of the problem.

We added a simple example system, as in Figure 1, and it illustrates that activating all tasks at the same time does not maximize the number of deadline misses in the problem window.

---

40 Reviewer's comment:

Quite a few definitions remain unclear to me, which are listed below. This prevents me from being really convinced that the paper is entirely correct.

Another point: The paper heavily relies on the fact that deadlines are constrained but this is explicitly stated only once in the definition of the system model, and not in the introduction.

Now, we explicitly point out this at the abstraction. "We provide a new weakly hard schedulability analysis method that applies to constrained-deadline periodic real-time systems"

---

41 Reviewer's comment:

> *Another assumption, which should be more clearly stated, is that the scheduler does not react to a deadline miss and lets tasks run to completion.*

The weakly hard analysis problem is formally defined at Section 3.1 (The weakly hard model), where "We assume the system utilization $U = \sum_{1 \le i \le n} U_i$ is lower than 1, meaning that each job is guaranteed to complete its requested execution at some point in time, whether it misses its deadline or not."

Also, in the Section 3 (The System Model), "If a task does not always finish before its deadline, it can have multiple active jobs at the same time. In such cases, these jobs are served in FIFO order, meaning that the latter job cannot execute until all its predecessors are completed."

---

42 Reviewer's comment:

> *Finally, the task model under consideration is quite restricted (independent, periodic tasks with constrained deadlines). You should at least mention how your approach could be generalized to sporadic tasks etc. If that is not the case it is difficult to see how your approach could be applicable.*

The weakly hard analysis in this paper focuses on offset-free periodic task systems. We demonstrate its applicability to the resource sharing problem, and in this revision, we further show its extensibility to task model with jitters.

For the weakly hard analysis of the sporadic case, as we state in Section 2 (State of the Art), it is addressed by the line of works based on typical worst-case analysis. *Meanwhile, "This line of works and the method in this paper are orthogonal and the two extend the weakly hard real-time system analysis at different directions. Combined use of the two (in the future) can make the safety analysis and system design more comprehensive."*

---

43 Reviewer's comment:

> *All in all, the paper has some merit but its limitations regarding the scope of the model taken into account and the presentation of the work make it fall below the level of quality required at EMSOFT. If however my comments regarding correctness of the analysis and extendability of the solution to handle more complex systems can be addressed then I would not be against acceptance.*

Its extensibility is shown by the application to the shared resource problem and tasks with jitters.

Its correctness is implicitly verified through the set of linear constraints developed in this paper. More details are given as the response for Comment 37.

---

44 Reviewer's comment:

> *Other comments: - A level-i busy period is a *maximal* interval such that...*
>
> *- Please add a subsection before 3.1 to explain in more detail and illustrate issues related to level-i busy periods.*

In the revision, we clarified the concept of a level-i period in Section 3 (The System Model), and an example is also added.

"A level-i busy period *is defined as a time interval during which the processor is always occupied by the execution of tasks with priority higher than or equal to τ_i. For example, in Figure 1, $[s_0, f_2)$ and $[a_3, f_3)$ are two level-3 busy periods: Because the focus of this paper is not the single task WCRT, our definition of a busy period does not strictly follow its original meaning in [21], where a busy period corresponds to the maximal level-i busy period in this paper."*

---

45 Reviewer's comment:

> *- You should formally define what you mean by problem window.*

The problem window is now formally defined at Section 3.1 (The weakly hard model), together with other concepts for the weakly hard analysis problem. "the problem window *for the analysis starts from a time instant s_0 (let us say $s_0 = 0$), which is the earliest time instant such that from s_0 to a_1 the processor is fully occupied by the execution of higher priority tasks, till the finish of the Kth job, i.e., $[s_0, f_K)$."*

An example is also added for the illustration purpose.

---

46 Reviewer's comment:

> *For me observations (O1) and (O2) are not understandable due to this lack of formalization. What I understand is that a problem window should start with a non-schedulable job J_1 but I really don't see why.*

O1 and O2 aim to show that, in order to maximize the number of deadline misses in the window, we should start the analysis from a non-schedulable job J, but the job J' just before J must be schedulable. Given any task activation pattern (let us say Scenario A) that does respect O1 or O2, we can always find a Scenario B that satisfies O1 and O2, and the number of deadline misses in Scenario B does not decrease (w.r.t Scenario A).

We added concrete examples to help understand.

---

47 Reviewer's comment:

> *- Figures 1 and 2 should be explained and commented. In fact, you should show higher priority jobs.*

We upgraded Figure 1, and inlined examples based on it for each necessary linear constraint. As an example, "For the particular scenario in Figrue 1, $L_1 = 0.5$ and $L_3 = 0$."

Figrue 2 only aims to show several shortcut notations: "For short, we first define several terms: $\rho_k = f_k - a_k$, $\lambda_k = f_k - (a_k - L_k)$ and $\lambda'_k = f_k - f_{k-1}$." To avoid confusing, we deleted FIgure 2. Instead, "As an example, in Figure 1, $\rho_1 = 7.5$, $\lambda_1 = 7$ and $\lambda'_2 = 3$. "

---

48 Reviewer's comment:

> *- "inidicates" −− > "indicates"*

Fixed.

---

49 Reviewer's comment:

> *- The right-hand side bound of the equation related to offsets should be justified.*

"Assume that the first job $J_{j,1}$ of τ_j in the window arrives at time $s_0 + T_j - r_j + \epsilon$ with $\epsilon > 0$. This implies the precedent job (let us say) $J_{j,0}$ is activated at time $s_0 - r_j + \epsilon$. Because any job of τ_j needs at least r_j time to finish, $J_{j,0}$ will be still active at time instant s_0, which contradicts the fact that s_0 is the earliest time instant that from s_0 to a_1 the processor is fully occupied by higher priority task execution. This small proof explains why the upper bound for α_j is $T_j - r_j$."

---

50 Reviewer's comment:

> *- Please recall what M is in (1), (2) etc. (it has been introduced quite a long time before).*

Added.

"We define M as a big enough constant value that will be used in the model to encode conditional constraints, and this is well known as the big-M method."

"We remind that the M is supposed to be a big enough constant."

"Again, we apply the big-M method."

"the big-M constraint (7) *is updated to"*

"and this explains these big-M terms in (C9)*."*

Also, we added high level operators to help understand the big-M encoding. E.g., $b_k = 0 \Leftrightarrow f_k \leq a_k + D_i$, $\beta_k = 0 \Leftrightarrow f_k \leq a_{k+1}$, $\beta_k = 1 \Rightarrow \iota_k = 0$, "Otherwise, $\beta_{k-1} = 0$ implies that $\Phi_k + C_i = \lambda_k$".

---

51 Reviewer's comment:

> *- In the paragraph on the number of interfering jobs from higher priority tasks, what is the number of jobs within a time interval? The number of jobs released?*

Yes.

"Given a job J_k of τ_i and a higher priority task τ_j, we define an array of boolean variables $\Gamma f_{j,k}[p] \in \mathbb{B}$ to count the number of jobs (i.e., job releases) of τ_j inside the time interval (p is used to index these jobs).

- *$[a_k - L_k, f_k)$ if J_{k-1} does not interfere with J_k, i.e., $\beta_{k-1} = 0$;*
- *$[f_{k-1}, f_k)$ if J_{k-1} does interfere with J_k, i.e., $\beta_{k-1} = 1$.*

"

Such "time intervals" are always in form of $[a_k - L_k, \ldots)$ or $[f_{k-1}, \ldots)$, which indicates that there is no pending workload from higher priority tasks when entering such intervals, and numer of jobs within them are in fact number of job released within them.

---

52 Reviewer's comment:

> *- The paragraph on refining the interferences from higher priority tasks is confusing and must be clarified.*

In the revision, we added Table 2) to show concrete variables' values during the refining procedure.. "As shown in Table 2, for the example in Figrue 1, when $j = 1$ and $k = 3$ there is $L_3 = 0$ and $\Delta_{1,3}$ jobs from τ_1 within the time interval $[a_3 - L_3, f_3)$. As $\beta_1 = 1$, from f_1 to f_2, $\Delta_{1,2} = 1$ job is released."

"For instance, during the interval $[f_2, a_3 - L_3)$ in Figure 1, no higher priority jobs are released: $\Gamma_{1,3} = \Gamma_{2,3} = 0$."

More importantly, we improved the readability for concepts like $If_{j,k}$ and $IL_{j,k}$. After all, the refining procedure is for refining the values of If and IL, which are linearized from their original integer values

- *$If_{j,k} = \lceil \frac{f_k - \alpha_j}{T_j} \rceil$ is linearized as $0 \leq If_{j,k} - \frac{f_k - \alpha_j}{T_j} < 1$.*
- *$\beta_{k-1} = 0 \Rightarrow IL_{j,k} = \lceil \frac{a_k - L_k - \alpha_j}{T_j} \rceil$ is linearized as $\beta_{k-1} = 0 \Rightarrow 0 \leq IL_{j,k} - \frac{a_k - L_k - \alpha_j}{T_j} < 1$ The reason for $\beta_{k-1} = 0$ is that "that we only use the value of L_k, and thus the interval $[0, a_k - L_k)$, when $\beta_{k-1} = 0$."*

---

53 Reviewer's comment:

> *- (C9) needs to be clarified. For example, what happens for schedulable tasks ($N_i = 1$)? I did not understand why the term $-M.(1 - \beta_x)$ is needed. For fixed priority preemptive with $D_i <= T_i$ it is simple to prove that $\beta_x = 0$ is true only for $x = N_i$ where $x \in 1, \ldots, N_i$, which means that $\beta_1 = 1$ and the third term is not needed. Furthermore, k in the fourth term already excludes the last activation in the busy period and its response time is added in the fifth term in (C9).*

(C9) *always holds. $N_i = 1$ means that even the longest level-i busy period only contains one job of τ_i. Thus, every job inside the problem window will finish before the next job release, which implies that $\beta_x = 0$ in* (C9)*, and the big-M term $-M \cdot (1 - \beta_x)$ will guarantee the right side of* (C9) *to be a small enough value.*

As a matter of fact, an implicit pre-condition for the weakly hard analysis is that a task is not schedulable, otherwise, there will be no deadline miss in any case. As stated in the paper "schedulable tasks can be ruled out by simply performing a traditional hard schedulability test in advance".

"For arbitrarily N_i successive jobs inside an arbitrary problem window, we do not know if they are inside the same busy period, however, $\beta_k = 1$ is a sufficient condition for two jobs J_k and J_{k+1} to be in the same busy period (the same for $\beta_x = 1$) and this explains these big-M terms in (C9)*."*

---

54 Reviewer's comment:

> *- In the case study (Section 6.1), it would be better to illustrate the MILP by showing how the constraints look like for this example.*

We added an example system as in Figure 1 and use it to illustrate constraints at each necessary step.

A.5 Reviewer 5

55 Reviewer's comment:

> *The paper provides a new weakly hard schedulability analysis method for periodic real-time systems scheduled with fixed priority and without knowledge of the task activation offsets. The schedulability problem is abstracted to a Mixed Integer Linear Programming problem which can compute an upper bound for the number of deadline misses of an arbitrary K successive activations of a task.*
>
> *The paper improves the current weakly hard analysis for fixed priority periodic tasks systems by relaxing the constraint of systems analyzed from offset-determined to offset-free. Furthermore, an extension of the analysis method for shared resources*

is proposed. Also, the effectiveness of the method is well explored in the experiments section.

---

56 Reviewer's comment:

The method is not scalable as is showed in the experiments section.

As shown in the experiments, depending on the system configurations, the proposed MILP weakly hard analysis is scalable to tens of tasks when the problem window size K is relatively small, on the other hand, even if we increase the problem window size to be as big as 20, the method is still eligible to handle tasksets with more than 10 tasks. Implicitly, we set up the runtime threshold of the solver to 1800 seconds and they apply to all experiments in the paper. This threshold is already very realistic for many (if not most) temporal safety analysis senarios in avionics and automobiles.

---

57 Reviewer's comment:

With the increasing of taskset size, the percentage of the configurations that can obtain the results in reasonable time might decrease considerably.

In case there are 50 tasks in a system, we experienced a significant amount of tests that exceed the runtime threshold 1800 seconds. However, even in the case, there are still a significant amount of tests that get confirmed results: 42.7% when $m = 1$ and 94.0% when $m = 2$.

---

58 Reviewer's comment:

Furthermore,the presentation in section 4 is not clear enough and hard to read. More illustrations should be added.

More ilustrations are added in the revision.

In the revision, we focus on improving the readability of the paper. Especially,

- *we formalize key concepts in Section 3 (The System Model) and Section 3.1 (The weakly hard model)*
- *an example system, as in Figure 1, is added*
  - *to demonstrate that the critical instant does not lead to maximum deadline misses in the problem window*
  - *to illustrate key concepts on weakly hard analysis*
  - *to help explain linear constraints at each necessary step*
- *high level operators are used to annotate these big-M constraints*

---

59 Reviewer's comment:

Typo

1. Section 4.1, 4th paragraph, 4th line. indicates the portion of the level-i busy period. 2. Section 5, 5th paragraph, 6th line. largest elements in the multiset.

Fixed.