# Towards Principled Budget Enforcement in Real-Time Systems

Joseph Goh and James H. Anderson
Department of Computer Science, University of North Carolina at Chapel Hill
{jgoh, anderson}@cs.unc.edu

***Abstract*—The increasing complexity and parallelization of hardware and applications in embedded systems have brought unavoidable uncertainty in determining the worst-case execution times (WCETs) of real-time tasks. While budget enforcement can address this uncertainty by limiting an overrunning task from affecting the rest of the system, less explored is how to determine the rate and pattern of failures resulting from budget overruns. For analysis using probabilistic techniques, one must first consider potential dependence relations across different tasks or jobs of the same task. As a result, prior work on probabilistic WCET (pWCET) distributions seeking to enable independence assumptions has suffered from excessive pessimism and intricate derivation processes. In contrast, industry designs have opted for relatively simple heuristics such as "fudge factors," budgets set by scaling mean or observed worst-case execution times by a constant factor. However, such heuristics do not have a strong analytical foundation. This paper addresses this gap in theory and practice, presenting analysis of a budgeted real-time system's failure rate not reliant on extensive knowledge of a task's execution behavior or independence assumptions, only requiring approximations of the mean execution time and standard deviation. This analysis bounds the rate of deadline failures, particularly those which would violate weakly-hard robustness specifications, to efficiently and optimally allocate budget and to evaluate industry heuristics.**

***Index Terms*—budget allocation, FIT analysis, weakly-hard constraints, real-time systems**

## I. Introduction

Two trends are evident in real-time embedded applications today. First, the applications themselves are growing larger and less deterministic. Second, the underlying hardware platforms have become more parallelized and complex. Even in the face of this complexity, given *worst-case execution-time* (WCET) values, a real-time scheduler, and corresponding schedulability or response-time analysis, one may, in principle, validate whether a given system is guaranteed to meet its timing constraints.

Unfortunately, multiprocessor machines introduce inherent uncertainties to tasks' execution durations, hampering efforts to produce WCET values that are safe yet not excessively pessimistic. Prior work suggests that the complexity of multiprocessor platforms may be a fundamental barrier to producing useful WCET values through *static timing analysis*, which produces the WCET of a program by examining its code structure [1]. On the other hand, *measurement-based timing analysis* (MBTA) may be less pessimistic, but cannot guarantee that the true WCET was observed, and thus cannot provide absolute safety guarantees.

Work was supported by NSF grants CPS 2038960, CPS 2038855, CNS 2151829, and CPS 2333120.

This unavoidable uncertainty in execution times may be addressed through *budget enforcement*, where each task must execute within its provisioned budget. Such systems must be capable of tolerating and responding to some degree of budget *overruns*. Through budget enforcement, a task that overruns its budget can be prevented from affecting the scheduling guarantees for other tasks, allowing correctness of the wider system to be maintained.

It is not clear, however, how to best assign budgets when there is uncertainty in tasks' WCETs such that the likelihood of overruns is low enough to be tolerable without blindly introducing pessimism. Lacking an analytical foundation, it has been commonplace in industry to rely upon simple heuristics such as "fudge factors," i.e., setting budgets equal to average or maximum execution durations observed in testing, multiplied by some constant factor (e.g., 1.5, 2) [2], [3]. Such an approach, while simple to apply, may allocate budget ineffectively and waste capacity. Even more concerning is the possibility that this approach does not actually result in sufficient safety guarantees. Even if budget overruns are difficult to observe, over a large fleet of machines or long periods of operation, they may cause system failures at an unacceptable rate.

**This paper.** We demystify the relationship between task budgets and system failures from a stochastic perspective, giving bounds for the frequency of system failures due to unacceptable patterns of overruns. We do not rely upon precise information about tasks' WCETs, pWCET distributions, or dependency relations, resulting in widely applicable and accessible analysis.

Our approach only requires a priori information of two kinds: safe overapproximations to the mean and standard deviation of each task's execution times, and weakly-hard $(h, k)$ robustness specification, characterizing the degree and pattern of acceptable budget overruns and commonly referred to as an $(m, k)$ constraint. Notably, our work acknowledges the possibility that jobs are not only dependent on one another, but that individual jobs may have distributions and behavior discordant from long-term characteristics of the task.

Our analysis is scheduler-agnostic, provided that budgets are enforced at job-level granularity. We demonstrate how this analysis can be leveraged to assign budgets that minimize the

rate of failures while meeting schedulability requirements and examine how the constant-factor scaling heuristic compares.

**Related work.** Weakly-hard real-time systems have been long studied and used to specify the patterns of deadline misses that an application or system can tolerate [4]–[13]. Studies have also produced schedulability analysis for weakly-hard constraints, providing guarantees of strict adherence to such constraints or bounding the likelihood of constraint violation [5], [14], [15], [15]–[23]. These studies are often reliant on simulation [14], static verification approaches [5], [15]–[19], or scheduler-specific analyses [15], [17]–[20].

Some recent work has taken a probabilistic approach, analyzing deadline misses and weakly-hard constraint violations [14], [21]–[23]. Work such as [21] builds and analyzes rigorous stochastic models for tasks and their execution sequences, while notably also focusing on long-term failure metrics such as mean time to failure (MTTF) and failures in time (FIT). Probabilistic analyses for fault-tolerant systems, particularly those characterized as containing workloads of mixed criticality, have also been explored [24]–[27]. For instance, work such as [26] assumes probabilistic execution times of real-time tasks and characterizes a system's reliability with metrics such as deadline misses per hour. Other work has taken advantage of stochastic execution time models in combination with execution budgets to achieve probabilistic guarantees [28]–[31]. However, to our knowledge, prior work has not placed weakly-hard systems and their failures in the context of effective and efficient budgeting.

The computation of an independent and identically distributed (i.i.d.) *probabilistic WCET* (pWCET), i.e., a safe overapproximation to the cumulative distribution function of a task's execution times, can greatly simplify the derivation of overrun likelihood and ensuing analysis. Prior work has used pWCET distributions to characterize system reliability the under varying definitions of system failures and robustness [26]. Early pWCET characterizations have been insufficient to guarantee soundness of analysis under independence assumptions [32]–[34]. While Bozhko et al. recently proposed meticulous requirements for a pWCET definition to be safe under i.i.d. assumptions, methods for the efficient derivation of pWCET distributions satisfying their requirements are not yet well researched [32]. Furthermore, the degree of pessimism required to mask dependencies may be too great for some applications. Recent work by Marković et al. on the worst-case deadline failure probability under fixed-priority scheduling is notable in that, similar to this paper, they adapt Cantelli's inequality to avoid relying on independence assumptions or detailed knowledge of execution time distributions [20].

**Contributions.** Our contribution is three-fold.

(i) We present a sharp bound on the failures in time (FIT), the expected number of failures over a defined length of operation, of a real-time task system that relies only on knowledge of tasks' mean execution-time requirement and standard deviation.

(ii) We demonstrate how the above analysis may be adapted into an optimization problem that allocates budgets to minimize failures while maintaining schedulability constraints.

(iii) We examine industry heuristics and budget overrun strategies to characterize their efficacy and provide updated guidelines.

**Organization.** After covering necessary background and listing analysis assumptions in Sec. II, we analyze overrun and weakly-hard failure rates in Sec. III. We demonstrate how to optimize budgets with respect to our analysis in Sec. IV, evaluating our approach as well as the constant-factor heuristic in Sec. V. Sec. VI concludes the paper.

## II. Preliminaries and Analysis Assumptions

We consider a set $\tau \triangleq \{\tau_1, \tau_2, \ldots, \tau_n\}$ of $n$ sporadic tasks running on a platform of one or more identical processors. Each task $\tau_i$ has a known *period* $T_i$, the minimum separation time between arrivals of its jobs. Jobs of each $\tau_i$ are assigned an *execution budget* of $C_i$ time units, within which each job is expected to finish. Time is modeled as continuous.

We assume a hard real-time scheduler with budget enforcement, i.e., all tasks are guaranteed to meet their deadlines in the absence of budget overruns. The execution of a task is halted whenever it exhausts its budget. We discuss later in this section the different policies a scheduler can apply to unfinished jobs that overrun their budget.

Because each job's budget is enforced and the overrun strategies we examine ensure that any overruns are isolated to a single job or jobs of the same task, the execution budget and any scheduler-specific guarantees for each task is not negatively affected by one another. This allows failures to be analyzed for each task individually, agnostic of the specific scheduling algorithm used or of any interactions with other tasks in the system.

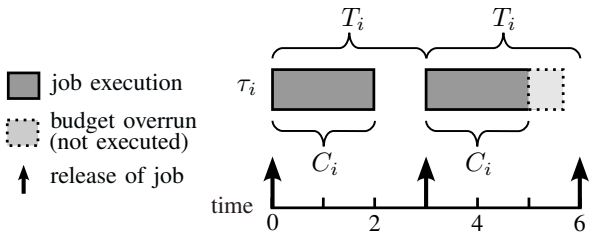


Fig. 1. Timeline of jobs of $\tau_i$ with $T_i = 3$ and $C_i = 2$. At time $t = 5$, the second job depletes its budget and is not allowed to continue executing.

**Weakly-hard specification model.** We examine the $(h, k)$ robustness specification, often referred to as a *weakly-hard* specification, to characterize the patterns of deadline failures that a system can or cannot tolerate. An execution scenario of a task is $(h, k)$ robust if, every window of $k$ consecutive jobs contains at least $h$ successes. We assume each task $\tau_i$ is subject to one such robustness requirement, denoted $(h_i, k_i)$.

We consider only the jobs that finish execution without exceeding their originally assigned budget to be *successful*.

All others, whether they are terminated, allowed to finish with additional budget, or skipped altogether, are considered *failures*. A window of $k_i$ consecutive jobs of $\tau_i$ with fewer than $h_i$ successes is considered as a *system failure*.

Multiple overlapping windows of size $k_i$ violating $(h_i, k_i)$ robustness are counted as at most one system failure event. In other words, each job failure can contribute to at most one system failure. We believe this is appropriate since such a failure would imply continuation of normal execution is unsafe, immediately triggering a separate fallback or recovery mechanism. We discuss later in Sec. IV how one might give more weight to failures of specific tasks, since our failure model inherently implies that constraints specified with a larger $k_i$ value have fewer violations over the same interval.

**Stochastic execution duration model.** The execution durations of jobs are modeled as random variables. We define $X_j^i$ as the non-negative real-valued random variable whose outcome is equal to the full execution duration of the $j^{\text{th}}$ job of task $\tau_i$.

Crucially, we seek to minimize assumptions regarding ground-truth distributions of these random variables, as well as possible dependence relations. This includes the existence of a consistent probability distribution, expected value, or variance in execution durations across jobs of the same task. To see why such an assumption could be problematic, consider the following example.

**Example 1** (Danger of unfounded i.i.d. assumptions)**.** Let $\tau_i$ be a task with $T_i = D_i = 1$ whose jobs, in order of activation, have a full execution duration which repeats the following pattern: $\{1, 1, 3\}$. Suppose that $\tau_i$ given a budget of $C_i = 1$ and must be $(3, 3)$ robust. Clearly, a randomly sampled job of $\tau_i$ has a $\frac{1}{3}$ probability of overrun. If one were to wrongly assume execution durations of $\tau_i$ are i.i.d., they would conclude that a random window of three consecutive jobs has a $(1 - \frac{1}{3})^3 = \frac{8}{27}$ chance of conforming to $(3, 3)$ robustness. In reality, every single window has at least one overrun, violating the robustness specification with a probability of 1.

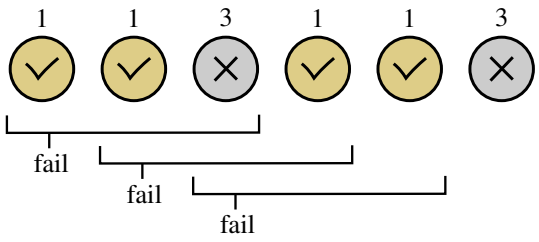


Fig. 2. Illustration of the execution time pattern described in Ex. 1. Despite being possibly characterized as having a $1/3$ probability of overrun, no window of 3 jobs of $\tau_i$ can ever satisfy $(3, 3)$ robustness.

Example 1 illustrates how even a relatively simple correlation or pattern of job execution durations, coupled with unfounded i.i.d. assumptions, can lead to false confidence in the safety and reliability of a system.

Thus, instead of a consistent distribution among jobs of the same task, we only assume the existence of a long-term distribution that jobs converge to in the aggregate. Such a distribution permits ensuing analysis to be applied to tasks whose execution times follow any processes or pathological worst-case patterns. We state our assumption more formally below.

Let $Y_i^j$ be defined as the random variable whose outcome is selected uniformly at random from $\{X_i^1, \ldots, X_i^j\}$, the execution durations of the first $j$ jobs of $\tau_i$. We assume that there exists an independent, real-valued random variable $Y_i$ with finite mean and variance such that $Y_i^j$ converges in probability towards $Y_i$. That is, for all $\varepsilon > 0$,

$$\lim_{\ell \to \infty} \mathbb{P}\left(|Y_i^\ell - Y_i| > \varepsilon\right) = 0.$$

While proving the existence and exact distribution of such $Y_i$ is far from trivial, the lack of such an assumption would preclude most probabilistic analysis aiming to generalize the long-term behavior of any application. Intuitively, $Y_i$ describes the distribution of execution times of $\tau_i$ across a "sufficiently long" interval of execution. For the remainder of the paper, we assume that our analysis interval length is long enough such that the overall deviation from $Y_i$ is negligible. The exact interval lengths used for our analysis metrics are described later in this section and Sec. III.

We denote the expected value and standard deviation of $Y_i$ as $e_i$ and $s_i$, respectively, and assume they are finite and positive. We also assume that it is possible to determine "safe" overapproximations of these values, denoted $\hat{e}_i$ and $\hat{s}_i$, such that $\hat{e}_i \geq e_i$ and $\hat{s}_i \geq s_i$ with high certainty. We require that $C_i \geq e_i$ for our analysis. We consider this requirement to be reasonable, as assigning a budget lower than the mean execution duration would clearly result in excessive overruns.

**Overrun strategies.** When a job $J$ of task $\tau_i$ is about to overrun its budget and miss its deadline, the system may respond via one of the following strategies [35]:

(i) *Kill*. Terminate $J$ and undo any changes from the execution of $J$ as necessary.

(ii) *Skip-Next*. Skip the next job of $\tau_i$ and allow $J$ to finish by executing in place of the skipped job, using the now freed budget. Note that skipped jobs always miss their deadlines, and $J$ may not finish, even with additional budget.

(iii) *Queue*. $J$ is allowed to finish executing using budget originally to be allocated to succeeding jobs. Succeeding jobs are not skipped, but also execute to completion with whatever budget is available after $J$ completes. The strategy can be reapplied if necessary.

We analyze strategies (i) and (ii) in this paper, as (iii) can inherently lead to chains of deadline misses under a HRT scheduling policy, from which recovery can be difficult [35], [36]. Any overheads incurred by applying these strategies are considered to be negligible, already accounted for by the underlying system framework, or incorporated into job execution durations.

**Characterizing failure rates.** We wish to characterize the rate of system failures by deriving an upper bound on the *failures in time* (FIT) of each task, defined as the number of expected

number of failures over a large interval of given length. For the purposes of our analysis, each window of $k_i$ consecutive jobs of each $\tau_i$ with fewer than $h_i$ successes is considered a system failure contributing to the FIT metric. We derive a bound for a system's FIT in the following section and later show how to optimize for it.

TABLE I
SUMMARY OF NOTATION

| Symbol | Meaning |
|---|---|
| $n$ | Number of tasks |
| $M$ | Number of processors |
| $\tau$ | Task system |
| $\tau_i$ | $i^{\text{th}}$ task of $\tau$ |
| $T_i$ | Period of $\tau_i$ |
| $C_i$ | Execution budget of $\tau_i$ |
| $(h_i, k_i)$ | $(m.k)$ robustness specification of $\tau_i$ |
| $X_i^j$ | Random var. representing execution time of $j^{\text{th}}$ job of $\tau_i$ |
| $Y_i$ | Long-run distribution of execution times of $\tau_i$ |
| $e_i, s_i$ | Expected value and standard deviation of $Y_i$, respectively |
| $\hat{e}_i, \hat{s}_i$ | Safe overapproximations to $e_i$ and $s_i$, respectively |
| $\rho_i(t)$ | Function providing bound on $\mathbb{P}(Y_i \geq t)$ |
| $P_i$ | Set of tasks assigned to the $i$th processor by a partitioned scheduler |

## III. ANALYSIS

Given a budget assignment and overrun strategy, we derive a FIT bound for the entire system via the following steps:

(i) Derive an upper bound on $\mathbb{P}(Y_i > t)$, which translates to the proportion of jobs requiring more than $t$ time units to complete.
(ii) Use the bound derived in (i) to bound the FIT of a task given its parameters and $(h, k)$ specification, considering each overrun handling strategy.
(iii) Combine per-task FIT bounds into a system-wide or fleet-wide FIT metric.

### A. Bounding Overrun Probability

First, we wish to bound the portion of jobs requiring more than a given time length to finish execution. We utilize an upper bound on $\mathbb{P}(Y_i > t)$ derived by adapting Cantelli's inequality, which we state here.

**Theorem 1** (Cantelli's inequality [37])**.** If $X$ is a real-valued random variable with finite mean value $e$ and standard deviation $s$, for any $\lambda > 0$,

$$\mathbb{P}(X - e \geq \lambda) \leq \frac{s^2}{s^2 + \lambda^2}. \tag{1}$$

**Corollary 1** (adapted from [20])**.** Given $Y_i$ with finite expected value and standard deviation $e_i$ and $s_i$, respectively, for any $t > e_i$ we have

$$\mathbb{P}(Y_i \geq t) \leq \frac{s_i^2}{s_i^2 + (t - e_i)^2}. \tag{2}$$

*Proof.* By substituting $X$ with $Y_i$, $e$ and $s$ with $e_i$ and $s_i$, respectively, in (1), we obtain the desired inequality. □

As stated in Sec. II, given a sufficiently long interval of job executions, the aggregate distribution of execution times is expected to approach the distribution of $Y_i$. Thus, the bound provided in Cor. 1 serves to upper-bound the long-term expected *proportion* of jobs of $\tau_i$ requiring at least $t$ time units to complete execution. Note that Cor. 1 cannot be used to draw conclusions about the execution requirements of any specific job.

The following corollary states that (2) is robust with respect to overapproximations of $e_i$ and $s_i$.

**Corollary 2** (adapted from [20])**.** Given $Y_i$ and overapproximations to the expected value and standard deviation $\hat{e}_i$ and $\hat{s}_i$, for any $t > \hat{e}_i$ we have

$$\mathbb{P}(Y_i \geq t) \leq \frac{\hat{s}_i^2}{\hat{s}_i^2 + (t - \hat{e}_i)^2}. \tag{3}$$

*Proof.* By Cor. 1, we have

$$\begin{aligned}
\mathbb{P}(Y_i \geq t) &\leq \frac{s_i^2}{s_i^2 + (t - e_i)^2} \\
&= 1 - \frac{(t - e_i)^2}{s_i^2 + (t - e_i)^2} \\
&\leq 1 - \frac{(t - e_i)^2}{\hat{s}_i^2 + (t - e_i)^2} \\
&= \frac{\hat{s}_i^2}{\hat{s}_i^2 + (t - e_i)^2} \\
&\leq \frac{\hat{s}_i^2}{\hat{s}_i^2 + (t - \hat{e}_i)^2}.
\end{aligned}$$

□

For brevity of notation in later analysis, we define $\rho_i(t)$ as follows:

$$\rho_i(t) = \frac{\hat{s}_i^2}{\hat{s}_i^2 + (t - \hat{e}_i)^2}. \tag{4}$$

Cantelli's inequality provides a sharp bound, as does Cor. 1, meaning that general improvements to these bounds are not possible without additional knowledge or assumptions about the random variable under analysis, as there exist random variables $X$ (resp. $Y_i$) and values of $\lambda$ (resp. $t$) for which the bound is tight. We illustrate this property with a simple example.

**Example 2** (Sharpness of Cor. 1)**.** Consider a task $\tau_i$ whose job execution times follow a two-point distribution given as follows for some positive $x$:

$$\forall j, \quad \mathbb{P}(X_i^j = t) = \begin{cases} 1 - p & \text{if } t = x \\ p & \text{if } t = 10x \\ 0 & \text{otherwise.} \end{cases}$$

When $p$ is small, $\tau_i$ can be thought of as having an extremely predictable execution time for most activations, but under rare, difficult-to-observe circumstances, exhibits a ten-fold increased execution time. Jobs of $\tau_i$ have an expected execution requirement of $e_i = x + 9px$ with standard deviation $s_i = 9x\sqrt{p(1-p)}$. (Note that the latter can be derived simply by scaling the Bernoulli distribution which takes the value 1 with probability $p$ by $10x - x = 9x$.)

Applying the aforementioned distribution and $t = 10x$ to (2) gives us

$$\begin{aligned}
\mathbb{P}(Y_i \geq t) &\leq \frac{s_i^2}{s_i^2 + (10x - e_i)^2} \\
&= \{\text{Using } e_i = x + 9px \text{ and } s_i = 9x\sqrt{p(1-p)}\} \\
&\quad \frac{\left(9x\sqrt{p(1-p)}\right)^2}{\left(9x\sqrt{p(1-p)}\right)^2 + (10x - (x + 9px))^2} \\
&= \frac{(9x)^2 p(1-p)}{(9x)^2 p(1-p) + (9x(1-p))^2} \\
&= \frac{p}{p + (1-p)} \\
&= p.
\end{aligned}$$

This is the exact probability that $X_i^j = 10x$, illustrating how sharpness of our bound can be observed even under somewhat mundane execution-time distributions, and showing that Cor. 1 cannot be improved upon without additional knowledge about the underlying task's execution time distribution.

### B. Bounding System Failures

We now wish to bound the total number of system failures, i.e., we wish to derive the maximum number of $(h, k)$ robustness violations. Because we do not make any assumptions about dependence relations between execution times of different jobs, failures may occur in any sequence or pattern so long as the bound given in Cor. 1 is respected in the long run. Therefore, instead of attempting to reason about the likelihood that a random window of $k_i$ jobs of $\tau_i$ violates $(h_i, k_i)$ robustness, we focus on the worst-case number of violations that can be incurred by overrunning jobs.

We consider an analysis interval of $\ell$ time units. As stated earlier, we assume $\ell$ is large enough such that any deviations from Cor. 1 are negligible. We examine two overrun strategies separately, Kill and Skip-Next, as the worst-case overrun behavior for each are different, affecting the resulting FIT bounds.

*1) Kill:* For tasks whose overruns are handled by a Kill strategy, i.e., aborting the job about to overrun and leaving subsequent jobs unaffected, the worst-case number of failures can be derived rather easily. The following lemma bounds the number of system failures under a Kill strategy.

**Lemma 1.** Within an analysis interval of $\ell$ time units, the expected number of system failures from jobs of a task $\tau_i$ whose overruns are handled by a Kill strategy is at most

$$\frac{\rho_i(C_i)}{k_i - h_i + 1} \cdot \lceil \ell / T_i \rceil. \tag{5}$$

*Proof.* Under a Kill strategy, only jobs that overrun their budget and are subsequently aborted can fail to meet their deadlines.

By Cor. 2 and (4), the expected proportion of jobs of $\tau_i$ in the analysis interval that would overrun their budgets is at most

$$\mathbb{P}(Y_i > C_i) \leq \mathbb{P}(Y_i \geq C_i) \leq \rho_i(C_i).$$

The number of jobs released by $\tau_i$ within the analysis interval is at most $\lceil \ell / T_i \rceil$. Note that we use the number of jobs released instead of jobs released *and* whose deadlines fall within the analysis interval, as the latter depends on the deadline model of the task system and is bounded by the former. Thus, the maximum expected number of jobs that would overrun their budget is at most

$$\rho_i(C_i) \cdot \lceil \ell / T_i \rceil. \tag{6}$$

Since each job failure can contribute to at most one system failure, and at least $k_i - h_i + 1$ failures are required to violate $\tau_i$'s robustness specification, the expected total number of system failures is at most

$$\frac{\rho_i(C_i)}{k_i - h_i + 1} \cdot \lceil \ell / T_i \rceil$$

as desired. □

*2) Skip-Next:* Tasks whose overruns are handled by a Skip-Next strategy are somewhat more complicated, as an overrunning job not only misses its own deadline, but can cause one or more subsequent jobs to be skipped, resulting in a variable number of failures.

Assuming no added delays from applying our overrun strategy, a job that executes for more than its allotted budget will cause the subsequent job to be skipped, then executes in place of the skipped job consuming its now freed budget.

A job that executes for more than $zC_i$ time units for some $z \in \mathbb{N}$ causes at least $z + 1$ job failures including itself. To guard against indefinitely long overruns, the number of times the Skip-Next policy can be applied to the same job may be limited to some constant integer, which we denote $z_i$. $z_i$ should be at most $k_i - h_i - 1$, as applying Skip-Next $k_i - h_i$ or more times would prevent any window containing the overrunning job from having $h_i$ successes.

The following lemma bounds the maximum number of job failures under a Skip-Next strategy.

**Lemma 2.** Let $\tau_i$ be a task whose overruns are handled by a Skip-Next strategy allowing each overrunning job to skip at most $z_i$ succeeding jobs. Within an analysis interval of $\ell$ time units, the expected number of failed jobs of $\tau_i$ is at most

$$\left(\rho_i(C_i) + \sum_{z=1}^{z_i} \rho_i(zC_i)\right) \cdot \lceil \ell / T_i \rceil \tag{7}$$

*Proof.* By induction on $z_i$. Every overrunning job is considered to miss its own deadline and cause at least one succeeding job to be skipped. The worst-case pattern of job execution times occurs when all jobs which overrun their budget are allowed to execute, contributing to failures, instead of being skipped due to other overrunning jobs.

**Base case.** Consider the case that $z_i = 1$. Each job requiring more than $C_i$ time units, skips at most one succeeding job each, contributing to at most two job failures per overrunning job. There are at most $\rho_i(C_i) \cdot \lceil \ell / T_i \rceil$ such jobs, as shown in

the proof of Lem. 1 and stated as (6). Thus, the number of job failures is at most

$$2\rho_i(C_i) \cdot \lceil \ell/T_i \rceil = \rho_i(C_i) \cdot \lceil \ell/T_i \rceil + \rho_i(C_i) \cdot \lceil \ell/T_i \rceil$$
$$= \left( \rho_i(C_i) \cdot + \sum_{z=1}^{1} \rho_i(zC_i) \right) \cdot \lceil \ell/T_i \rceil$$

as desired.

**Inductive step.** Suppose that the lemma holds for $z_i = x$. When $z_i = x+1$, jobs that execute for longer than $(x+1)C_i$ time units each skip exactly one more job than when $z_i = x$, contributing up to one additional job failure. By Cor. 2, the proportion of such jobs is at most $\rho_i((x+1)C_i)$, which we can multiply by the maximum number of jobs released by $\tau_i$ in the analysis interval $\lceil \ell/T_i \rceil$, giving us at most

$$\rho_i((x+1)C_i) \cdot \lceil \ell/T_i \rceil$$

additional job failures. Thus, the total number of job failures is at most

$$\left( \rho_i(C_i) + \sum_{z=1}^{x} \rho_i(zC_i) \right) \cdot \lceil \ell/T_i \rceil + \rho_i((x+1)C_i) \cdot \lceil \ell/T_i \rceil$$
$$= \left( \rho_i(C_i) + \sum_{z=1}^{x+1} \rho_i(zC_i) \right) \cdot \lceil \ell/T_i \rceil.$$

Hence, the lemma holds for all $z_i$. □

Using Lem. 2, the following lemma bounds the maximum number of system failures under a Skip-Next strategy.

**Lemma 3.** Let $\tau_i$ be a task whose overruns are handled by a Skip-Next strategy allowing each overrunning job to skip at most $z_i$ succeeding jobs. Within an analysis interval of $\ell$ time units, the expected number of system failures from jobs of a task $\tau_i$ is at most

$$\frac{\rho_i(C_i) + \sum_{z=1}^{z_i} \rho_i(zC_i)}{k_i - h_i + 1} \cdot \lceil \ell/T_i \rceil. \tag{8}$$

*Proof.* Similar to the proof of Lem. 1, we observe that each system failure requires at least $k_i - h_i + 1$ failed jobs, with each job failure contributing to at most one system failure. Dividing (7), the upper bound on the expected number of failed jobs from Lem. 2, by $k_i - h_i + 1$ gives us the desired bound. □

We define

$$\mathsf{FIT}(\ell, \tau_i, C_i, 0) = \frac{\rho_i(C_i)}{k_i - h_i + 1} \cdot \lceil \ell/T_i \rceil \tag{9}$$

as short-hand for the FIT bound given by Lem. 1 for tasks under a Kill strategy. We also define $\forall z_i \in \{1 \dots k_i - h_i - 1\}$,

$$\mathsf{FIT}(\ell, \tau_i, C_i, z_i) = \frac{\rho_i(C_i) + \sum_{z=1}^{z_i} \rho_i(zC_i)}{k_i - h_i + 1} \cdot \lceil \ell/T_i \rceil, \tag{10}$$

as short-hand for the FIT bound given by Lem. 3 for tasks under Skip-Next.

We are now prepared to describe the FIT of an entire task system.

**Theorem 2.** For each $\tau_i$ whose overruns are handled by Kill, let $z_i = 0$, and for each $\tau_i$ handled by Skip-Next, let $z_i$ be equal to the number of skipped jobs allowed per overrunning job. The FIT of task system $\tau$, measured over $\ell$ time units is at most

$$\sum_{\tau_i \in \tau} \mathsf{FIT}(\ell, \tau_i, C_i, z_i). \tag{11}$$

*Proof.* The FIT of task system $\tau$ is given by the sum of the FIT of each $\tau_i \in \tau$. The proof follows directly from Lem. 1, Lem. 3, and the definition of FIT from (9) and (10). □

The expression given by Thm. 2 may also be summed across multiple task systems. Such a summation may be useful in scenarios such as bounding the FIT of an airline's entire aircraft fleet. In the next section, we demonstrate how a system designer may determine budget assignments that optimally minimize the FIT bound given by Thm. 2.

## IV. Optimizing Budget Allocation

In this section, we demonstrate via a case study how one may allocate budgets to tasks effectively and efficiently to minimize the FIT of the system with respect to our analysis in Sec. III. We examine partitioned EDF scheduling on a multiprocessor with $M$ identical cores. While the analysis presented in Sec. III is scheduler-agnostic, the schedulability conditions of partitioned EDF are especially well suited for efficient optimization. We consider a task set whose overruns are managed by Kill, but the steps outlined in this section are easily modified for tasks handled by Skip-Next.

We present two optimization problems with the FIT bound presented in Thm. 2 as their objective. The first problem can be solved to find an assignment of $C_i$ values that minimize FIT while respecting schedulability constraints. The second problem uses the "fudge factor" heuristic, assigning a constant $c$ such that each $C_i$ is set equal to $c\hat{e}_i$. The problems are intentionally constructed to be efficiently solvable, enhancing the accessibility and utility of our analysis.

For the remainder of the paper, $C_i$ values are treated as variables, whereas other task parameters ($T_i$, $e_i$, $s_i$, etc.) and the interval length $\ell$ are considered constants.

### A. Budget Allocation via Convex Optimization

We first present an optimization to find the optimal budget allocation that minimizes system FIT with respect to our analysis in Sec. III. Budget values are subject to minimal constraints in this problem, with the addition of one constraint set that allows the problem to be a convex optimization problem, and thus efficiently solvable by methods such as Newton's method.

**Initial constraints.** We begin by adapting our FIT bounds into per-task variables and constraints.

**Constraint Set 1.** The following constraints bound the FIT of each task, with $F_i$ representing the FIT bound for $\tau_i$ under the Kill strategy given in Lem. 1 and (9).

$$\forall i \in \{1, \dots n\} : F_i = \mathsf{FIT}(\ell, \tau_i, C_i, 0)$$

Next, we must state constraints that assure that the resulting system is schedulable under partitioned EDF. We assume that our processor partitions are predetermined and let $P_i$ denote the set of tasks assigned to the $i^{\text{th}}$ processor. Each processor is schedulable, in the absence of budget overruns, if and only if its utilization is at most 1 [38]. Furthermore, the utilization of each task must not exceed 1.

**Constraint Set 2.** The following constraints correspond to our partitioned EDF schedulability condition.

$$\forall i \in \{1, \dots M\} : \sum_{\tau_j \in P_i} C_j/T_j \leq 1$$
$$\forall i \in \{1, \dots n\} : C_i/T_i \leq 1$$

While Constraint Sets 1 and 2 encompass most of the constraints needed to state our optimization problem, as is, the resulting objective function, characterizing the FIT bound for the entire system, would be non-convex and relatively inefficient to solve. To address this, we introduce a modification to make the objective convex within our solution space.

**Constraints for convexity.** With only our original requirement that $C_i$ is greater than $e_i$, the FIT bound given by Lem. 1 is non-convex. Thus, we wish to find the range of each budget variable $C_i$ within which our $F_i$ variables are convex functions of $C_i$. This allows us to derive and define constraints that make our objective, the sum of $F_i$ values, convex as well.

The following lemma identifies the values of $C_i$ for which $\rho_i(C_i)$ is convex.

**Lemma 4.** The function $\rho_i(t)$ is convex when $t > \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i$.

*Proof.* Taking the second derivative of $\rho_i(t)$ with respect to $t$ gives us

$$\begin{aligned}
\frac{d^2}{dt^2}\rho_i(t) &= \frac{d^2}{dt^2}\frac{\hat{s}_i^2}{\hat{s}_i^2 + (t-\hat{e}_i)^2} \\
&= \frac{d}{dt}\left(-\frac{2\hat{s}_i^2(t-\hat{e}_i)}{(\hat{s}_i^2 + (t-\hat{e}_i)^2)^2}\right) \\
&= \frac{8\hat{s}_i^2(t-\hat{e}_i)^2(\hat{s}_i^2 + (t-\hat{e}_i)^2)}{(\hat{s}_i^2 + (t-\hat{e}_i)^2)^4} \\
&\quad - \frac{2\hat{s}_i^2(\hat{s}_i^2 + (t-\hat{e}_i)^2)^2}{(\hat{s}_i^2 + (t-\hat{e}_i)^2)^4} \\
&= \hat{s}_i^2 \cdot \frac{8(t-\hat{e}_i)^2 - 2(\hat{s}_i^2 + (t-\hat{e}_i)^2)}{(\hat{s}_i^2 + (t-\hat{e}_i)^2)^3} \\
&= \hat{s}_i^2 \cdot \frac{6(t-\hat{e}_i)^2 - 2\hat{s}_i^2}{(\hat{s}_i^2 + (t-\hat{e}_i)^2)^3}.
\end{aligned}$$

From $t > \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i$, we get

$$\begin{aligned}
& t > \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i \\
\Rightarrow\ & t - \hat{e}_i > \frac{\sqrt{3}}{3}\hat{s}_i \\
\Rightarrow\ & \{\text{Squaring and multiplying both sides by } 6\} \\
& 6(t-\hat{e}_i)^2 > 2\hat{s}_i^2 \\
\Rightarrow\ & 6(t-\hat{e}_i)^2 - 2\hat{s}_i^2 > 0 \\
\Rightarrow\ & \{\text{Because } \hat{s}_i^2 > 0 \text{ and } \hat{s}_i^2 + (t-\hat{e}_i)^2 > 0\} \\
& \hat{s}_i^2 \cdot \frac{6(t-\hat{e}_i)^2 - 2\hat{s}_i^2}{(\hat{s}_i^2 + (t-\hat{e}_i)^2)^3} > 0.
\end{aligned}$$

Since $\frac{d^2}{dt^2}\rho_i(t)$ is positive when $t > \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i$, $\rho_i(t)$ must be convex for all $t$ satisfying the stated inequality. □

**Corollary 3.** The function $\mathsf{FIT}(\ell, \tau_i, C_i, 0)$, the FIT bound for a task under the Kill strategy, is convex when $C_i > \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i$.

*Proof.* The proof of Lem. 4 applies near equivalently, as $\mathsf{FIT}(\ell, \tau_i, C_i, 0)$ is a positive constant multiple of $\rho_i(C_i)$. □

Lem. 4 and Cor. 3 show that by requiring budget values at least $\frac{\sqrt{3}}{3}$ standard deviations greater than the mean, the FIT bound of $\tau_i$ under the Kill strategy can be made convex. Setting this requirement for each task's budget variable $C_i$ results in the system FIT bound being the sum of convex functions, making it also convex.

While we do not derive the exact thresholds herein, if analyzing tasks under a Skip-Next strategy, range of values $C_i$ for which the FIT bound given by Lem. 3 and (10) is convex may be determined by solving the following inequality for $C_i$ for each possible value of $z_i$:

$$\frac{\partial^2}{\partial C_i^2}\mathsf{FIT}(\ell, \tau_i, C_i, z_i) > 0.$$

When $t = \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i$, by Cor. 2 we have that

$$\begin{aligned}
\mathbb{P}(Y_i \geq t) &\leq \frac{\hat{s}_i^2}{\hat{s}_i^2 + (t-\hat{e}_i)^2} \\
&= \frac{\hat{s}_i^2}{\hat{s}_i^2 + \hat{s}_i^2/3} = \frac{3}{4},
\end{aligned}$$

a typically unacceptable probability of overrun. Even under a commonly assumed normal distribution, as opposed to the bound given by Cor. 2, $t = \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i$ results in up to an approximately 0.28 chance of overrun. Thus, we consider this added constraint to be reasonable and include it in our optimization problem as follows.

**Constraint Set 3.** The following constraints correspond to requirements for convexity of the objective as given by Cor. 3.

$$\forall i \in \{1, \dots n\} : C_i \geq \hat{e}_i + \frac{\sqrt{3}}{3}\hat{s}_i.$$

Note that Constraint Set 3 implies $C_i > e_i > 0$ as originally required by our analysis.

We may finally state our convex optimization problem.

**Optimization Problem 1.** The assignment of budget values $C_i$ for each $\tau_i$ obtaining the system FIT bound that is nearly optimal with respect to Thm. 2 is given by the solution to the following convex optimization problem:

$$\begin{array}{ll} \text{Minimize:} & \sum_{i=1}^{n} F_i \\ \text{Subject to:} & \text{Constraint Sets 1-3} \end{array}$$

For a solution that is guaranteed optimal with respect to Thm. 2, one may remove Constraint Set 3 from Optimization

Problem 1. However, the resulting problem would be non-convex and may be more computationally intensive to solve.

### *B. Budget Allocation with "Fudge Factors"*

We now wish to modify Optimization Problem 1 to use each budget value $C_i$ to be a constant multiple of $\hat{e}_i$, the expected execution time requirement of jobs of $\tau_i$. This allows us to apply and evaluate the efficacy of the aforementioned "fudge factor" heuristic. While we start by presenting a new optimization problem, the optimal solution, if this problem is feasible, has a simple, exact derivation which we show.

We begin by introducing the "fudge factor" variable $c$ along with the following constraints:

**Constraint Set 4.** The following constraints represent adherence to a "fudge factor" heuristic.

$$\forall i \in \{1, \dots n\} : C_i = c \cdot \hat{e}_i$$
$$c > 1$$

We can state our modified optimization problem as follows:

**Optimization Problem 2.** The constant factor $c$ obtaining the system FIT bound that is optimal with respect to Thm. 2 under a "fudge factor" heuristic is given by the solution to the following optimization problem:

Minimize: $\sum_{i=1}^{n} F_i$
Subject to: Constraint Sets 1, 2, and 4

While this problem may result in budget allocation that is less than optimal, a simple derivation for the optimal "fudge factor" $c$ can be shown.

The following two lemmas are used to show an intuitive property that, within the bounds of schedulability, a larger "fudge factor" yields lower FIT bounds.

**Lemma 5.** $\mathsf{FIT}(\ell, \tau_i, C_i, z_i)$ is monotonically decreasing with respect to $C_i$ when $C_i > \hat{e}_i$.

*Proof.* We first observe that $\rho_i(t)$ is monotonically decreasing when $t > \hat{e}_i$. Given arbitrary $a, b$ satisfying $\hat{e}_i < a < b$, we have

$$\begin{aligned}\rho_i(a) &= \frac{\hat{s}_i^2}{\hat{s}_i^2 + (a - \hat{e}_i)^2} \\ &> \{\text{Because } 0 < a - \hat{e}_i < b - \hat{e}_i\} \\ &\quad \frac{\hat{s}_i^2}{\hat{s}_i^2 + (b - \hat{e}_i)^2} = \rho_i(b).\end{aligned}$$

Now, again for arbitrary $a, b$ satisfying $\hat{e}_i < a < b$, consider the following cases:

(i) $z_i = 0$, i.e., $\tau_i$ is under a Kill strategy.
By (9), we have

$$\begin{aligned}\mathsf{FIT}(\ell, \tau_i, a, 0) &= \frac{\rho_i(a) \cdot \lceil \ell / T_i \rceil}{k_i - h_i + 1} \\ &> \{\text{Because } \rho_i(a) > \rho_i(b)\} \\ &\quad \frac{\rho_i(b) \cdot \lceil \ell / T_i \rceil}{k_i - h_i + 1} = \mathsf{FIT}(\ell, \tau_i, b, 0).\end{aligned}$$

(ii) $z_i > 0$, i.e., $\tau_i$ is under a Skip-Next strategy.
Similar to case (i), by (10), we have

$$\begin{aligned}\mathsf{FIT}(\ell, \tau_i, a, z_i) &= \frac{\rho_i(a) + \sum_{z=1}^{z_i} \rho_i(za)}{k_i - h_i + 1} \cdot \lceil \ell / T_i \rceil \\ &> \{\text{Because } \forall z \geq 1,\ \rho_i(za) > \rho_i(zb)\} \\ &\quad \frac{\rho_i(b) + \sum_{z=1}^{z_i} \rho_i(zb)}{k_i - h_i + 1} \cdot \lceil \ell / T_i \rceil \\ &= \mathsf{FIT}(\ell, \tau_i, b, z_i).\end{aligned}$$

Thus, in all cases, $\mathsf{FIT}(\ell, \tau_i, a, z_i) > \mathsf{FIT}(\ell, \tau_i, b, z_i)$ and the lemma holds. □

**Lemma 6.** The objective value of Optimization Problem 2 is monotonically decreasing with respect to feasible values of $c$.

*Proof.* Let $c_1, c_2$ be arbitrary values satisfying $1 < c_1 < c_2$. From Constraint Sets 1 and 4, we have

$$\begin{aligned}\sum_{i=1}^{n} F_i &= \sum_{i=1}^{n} \mathsf{FIT}(\ell, \tau_i, C_i, z_i) \\ &= \sum_{i=1}^{n} \mathsf{FIT}(\ell, \tau_i, c\hat{e}_i, z_i).\end{aligned}$$

Because $c_1 \hat{e}_i < c_2 \hat{e}_i$, by Lem. 5 we have

$$\sum_{i=1}^{n} \mathsf{FIT}(\ell, \tau_i, c_1 \hat{e}_i, z_i) > \sum_{i=1}^{n} \mathsf{FIT}(\ell, \tau_i, c_2 \hat{e}_i, z_i),$$

showing that the objective is monotonically decreasing with respect to feasible $c$. □

Using the results of Lem. 6, we can state the exact optimal value of $c$ in closed form. The following theorem states that the optimal "fudge factor" is one that is as large as possible without over-utilizing any processors.

**Theorem 3.** If Optimization Problem 2 is feasible, its objective is optimally minimized when

$$c = \min_{i \in \{1, \dots, M\}} \left\{ \left( \sum_{\tau_j \in P_i} \hat{e}_j / T_j \right)^{-1} \right\}. \tag{12}$$

*Proof.* A value of $c$ any larger than that given in (12) would violate one of the constraints in Constraint Set 2, since $C_i = c\hat{e}_i$ by Constraint Set 4. Note that if the value given by (12) is at most 1, then Constraint Set 4 cannot be satisfied, the problem is infeasible, and the theorem does not apply.

Thus, (12) must be the maximum feasible assignment of $c$ which, by Lem. 6, minimizes the objective. □

We implement and evaluate Optimization Problem 1 and the "fudge factor" heuristic solution given by Thm. 3 in Sec. V.

**Possible modifications to Optimization Problem 1.** While we do not evaluate them in this paper, possible modifications to the presented optimization problems include task-specific constraints on failure rates. For instance, tasks with a higher criticality level may be required not to exceed a given FIT value, or each $F_i$ may be multiplied by a preassigned weight

$w_i$ in the objective, with weights determined by how much impact a system failure caused by $\tau_i$ has.

If tighter bounds than those presented in Sec. III are available for a subset of tasks, perhaps stemming from additional information on some tasks' execution time distributions, it may be possible to replace corresponding constraints on $F_i$ in Constraint Set 1 with the improved bound expressions. However, doing so may render the resulting optimization problem non-convex or non-continuous, significantly impacting the computational complexity required to solve it.

When examining a scheduling policy other than partitioned EDF, Constraint Set 2 may be replaced with constraints corresponding to schedulability conditions for the scheduler under analysis. Note that such a modification is likely to affect the resultant optimization problem's computational complexity.

## V. Evaluation

In this section, we evaluate the analysis of Sec. III with experiments based on the case study described in Sec. IV. Using randomly generated task sets, we optimized budget allocations with respect to the FIT bound given by Thm. 2, comparing the results of solving Optimization Problem 1 versus applying the "fudge factor" heuristic based on Optimization Problem 2 and Thm. 3.

### A. Experimental Design

**Task generation.** We generated 200 task sets for each permutation of the following configurations. Periods were sampled log-uniformly from the range $[10\text{ ms}, 1000\text{ ms}]$. Targeting a partitioned EDF scheduler as examined in Sec. IV with $M$=4 identical processors, we sampled the number of tasks per processor uniformly at random from the integer range $[8, 32]$. Then, using Stafford's Randfixedsum algorithm [39] as suggested by [40], we generated expected utilization values for each task, given by $e_i/T_i$, summing up to each target value from $\{0.4, 0.5, \ldots, 0.9\}$ for each processor.

Expected values $\hat{e}_i$ for execution times were calculated based on each task's expected utilization value, while standard deviations $\hat{s}_i$ were selected by multiplying $e_i$ by a value selected from $[0.1, 0.5]$ uniformly at random. Note that because we select a consistent expected utilization value across processors, by Thm. 3, optimal "fudge factors" are determined at this point.

$(h, k)$ robustness specifications were generated with the window size set to $k = 5$, $k = 10$, or $k = 20$ across each task set. Each task set was either assigned a constant value of $h$ across all tasks or assigned random per-task $h$ values. Constant values of $h$ tested are $h = \lfloor k \cdot 0.6 \rfloor$ or $h = \lfloor k \cdot 0.8 \rfloor$, and $h = k - 1$. Task sets with mixed $h$ values had $h$ sampled from $[\lfloor k \cdot 0.6 \rfloor, k - 1]$.

**Budget allocation and FIT bound derivation.** For each task set, budget allocations were optimized using Optimization Problem 1 and the "fudge factor" heuristic based on Optimization Problem 2. The former was solved using the SciPy [41] implementation of a trust-region method [42], while the analytical solution from Thm. 3 was used directly for the latter. Optimization was performed on an AMD Ryzen 5 2600 6-core CPU. Resulting FIT bounds were recorded with the interval length $\ell$ set to 1 billion hours or $3.6 \times 10^{15}$ in milliseconds, i.e., the worst-case number of failures occurring over 1 billion operating hours. For brevity, we refer to the results of Optimization Problem 1 as ConvexFIT and that of Optimization Problem 2 as Fudge.

### B. Results and Observations

The results of our experiments are plotted in Fig. 3 and 4. Based on the obtained results, we make the following observations.

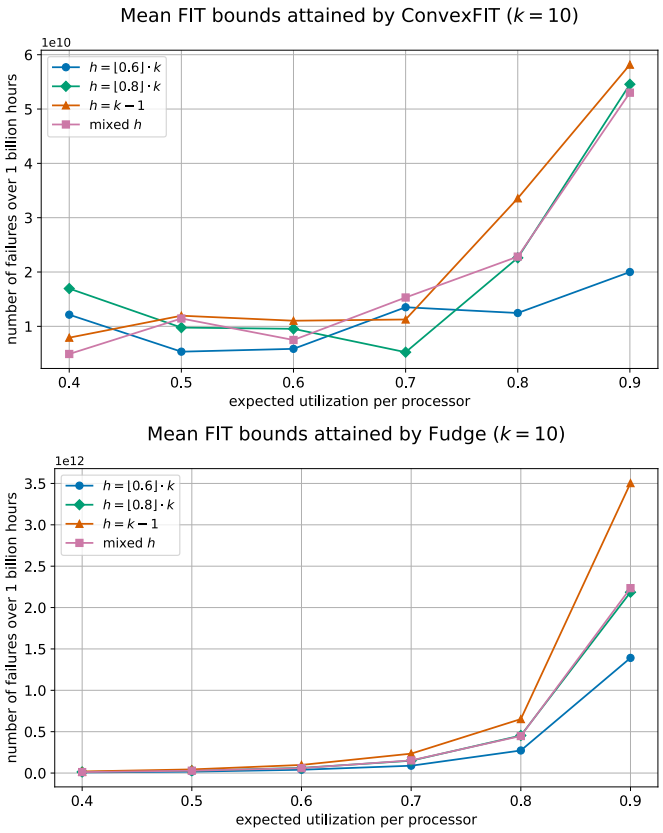


Fig. 3. Mean FIT bounds obtained from ConvexFIT and Fudge for task sets with $k = 10$.

**Observation 1.** Budget allocation through ConvexFIT, on average, achieved FIT bounds orders of magnitude lower than those obtained by Fudge. As shown in Fig. 4, the log difference between bounds from ConvexFIT and Fudge is consistently high, with the average difference ranging from about 4.0 to as high as 8.2 for some configurations. While it is unsurprising that optimization without restrictive constraints performs better than a relatively naïve heuristic, the observed degree of improvement suggests that, where possible, system designers should avoid using "fudge factors" as their only method of budget assignment.

**Observation 2.** While Fudge generally under-performs against ConvexFIT, our results include a small but significant

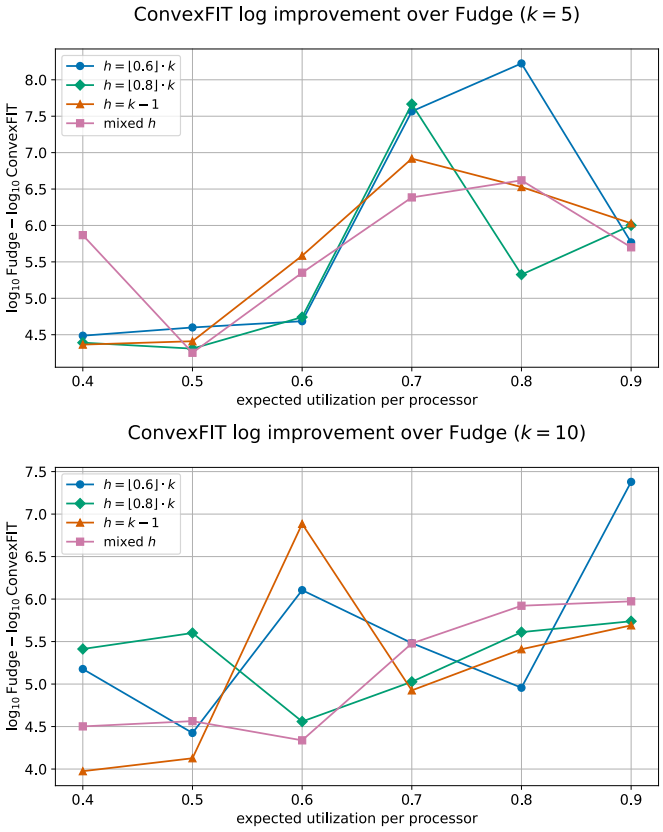


Fig. 4. Log (base 10) difference of mean FIT bounds obtained by ConvexFIT versus Fudge for task sets with $k = 5$ and $k = 10$.

portion of task sets for which Fudge achieves a FIT bound less than twice as large as ConvexFIT. This implies that certain task sets and parameters may indeed be effectively optimized by "fudge factors."

**Observation 3.** Higher per-processor expected utilization values generally correlate with higher FIT bounds as shown in Fig. 3. FIT bounds obtained through ConvexFIT, however, does not seem to consistently adhere to this trend. While the "jaggedness" in our ConvexFIT results may be due to insufficient sample size, the contrasting predictability of bounds obtained through Fudge indicate that the performance of ConvexFIT is inherently less predictable.

**Observation 4.** Computation time to derive FIT bounds via ConvexFIT was short, requiring roughly 1.5 seconds of computation per task set on average. Computation time for Fudge was negligible.

Based on these results, we believe ConvexFIT to be a meaningful improvement to the use of "fudge factors".

## VI. Conclusion

In this paper, we have explored how the reliability of real-time systems may be analyzed without access to reliable WCET values or detailed execution time distributions. We have shown the likelihood of overruns in a task system through first principles, relying only upon a priori knowledge of each task's long-term expected execution time and standard deviation values. With overrun probability as a foundation, we have determined a bound on the expected failures in time of a task system.

We demonstrated via a case study and corresponding experiments how our analysis may be leveraged to efficiently and effectively allocate task budgets in a principled manner. Furthermore, we have examined how "fudge factors" fare against our tighter optimizations, showing that, while being an intuitive and accessible heuristic, use of "fudge factors" may lead to significantly higher FIT of a system.

In future work, we plan to investigate how to further adapt our analysis to different scheduling algorithms and task models. We also hope to assess and improve FIT bounds under our analysis for systems employing various multiprocessor partitioning schemes.

## References


[1] R. Wilhelm, "Real time spent on real time," in *2020 IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 1–2.

[2] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," *The Computer Journal*, vol. 29, no. 5, pp. 390–395, 01 1986. [Online]. Available: https://doi.org/10.1093/comjnl/29.5.390

[3] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *28th IEEE International Real-Time Systems Symposium (RTSS 2007)*, 2007, pp. 239–243.

[4] P. Ramanathan, "Overload management in real-time control applications using (m, k)-firm guarantee," *IEEE Transactions on Parallel and Distributed Systems*, vol. 10, no. 6, pp. 549–559, 1999.

[5] C. Huang, W. Li, and Q. Zhu, "Formal verification of weakly-hard systems," in *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 197–207. [Online]. Available: https://doi.org/10.1145/3302504.3311811

[6] M. Hamdaoui and P. Ramanathan, "A dynamic priority assignment technique for streams with (m, k)-firm deadlines," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1443–1451, 1995.

[7] M. Caccamo and G. Buttazzo, "Exploiting skips in periodic tasks for enhancing aperiodic responsiveness," in *Proceedings Real-Time Systems Symposium*, 1997, pp. 330–339.

[8] G. Bernat and A. Burns, "Combining $\binom{n}{m}$-hard deadlines and dual priority scheduling," in *Proceedings Real-Time Systems Symposium*, 1997, pp. 46–57.

[9] G. Quan and X. Hu, "Enhanced fixed-priority scheduling with (m,k)-firm guarantee," in *Proceedings 21st IEEE Real-Time Systems Symposium*, 2000, pp. 79–88.

[10] R. Blind and F. Allgöwer, "Towards networked control systems with guaranteed stability: Using weakly hard real-time constraints to model the loss process," in *2015 54th IEEE Conference on Decision and Control (CDC)*, 2015, pp. 7510–7515.

[11] G. Bernat and R. Cayssials, "Guaranteed on-line weakly-hard real-time systems," in *Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001) (Cat. No.01PR1420)*, 2001, pp. 25–35.

[12] O. Gettings, S. Quinton, and R. I. Davis, "Mixed criticality systems with weakly-hard constraints," in *Proceedings of the 23rd International Conference on Real Time and Networks Systems*, ser. RTNS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 237–246. [Online]. Available: https://doi.org/10.1145/2834848.2834850

[13] H. Choi, H. Kim, and Q. Zhu, "Job-class-level fixed priority scheduling of weakly-hard real-time systems," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019, pp. 241–253.

[14] J. Lee, S. Y. Shin, L. C. Briand, and S. Nejati, "Probabilistic safe WCET estimation for weakly hard real-time systems at design stages," *ACM Transactions on Software Engineering and Methodology*, vol. 33, no. 2, dec 2023. [Online]. Available: https://doi.org/10.1145/3617176

[15] Y. Sun and M. D. Natale, "Weakly hard schedulability analysis for fixed priority scheduling of periodic real-time tasks," *ACM Transactions on Software Engineering and Methodology*, vol. 16, no. 5s, sep 2017. [Online]. Available: https://doi.org/10.1145/3126497

[16] Z. A. H. Hammadeh, R. Ernst, S. Quinton, R. Henia, and L. Rioux, "Bounding deadline misses in weakly-hard real-time systems with task dependencies," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, 2017, pp. 584–589.

[17] P. Pazzaglia, Y. Sun, and M. D. Natale, "Generalized weakly hard schedulability analysis for real-time periodic tasks," *ACM Transactions on Software Engineering and Methodology*, vol. 20, no. 1, dec 2020. [Online]. Available: https://doi.org/10.1145/3404888

[18] L. Köhler and R. Ernst, "Improving a compositional timing analysis framework for weakly-hard real-time systems," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2019, pp. 228–240.

[19] L. Ahrendts, S. Quinton, T. Boroske, and R. Ernst, "Verifying weakly-hard real-time properties of traffic streams in switched networks," in *30th Euromicro Conference on Real-Time Systems (ECRTS 2018)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), S. Altmeyer, Ed., vol. 106. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018, pp. 15:1–15:22. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECRTS.2018.15

[20] F. Markovic, P. Roux, S. Bozhko, A. V. Papadopoulos, and B. B. Brandenburg, "CTA: A correlation-tolerant analysis of the deadline-failure probability of dependent tasks," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec 2023, pp. 317–330. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/RTSS59052.2023.00035

[21] A. Gujarati, M. Nasri, R. Majumdar, and B. B. Brandenburg, "From iteration to system failure: Characterizing the FITness of periodic weakly-hard systems," in *31st Euromicro Conference on Real-Time Systems (ECRTS 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), S. Quinton, Ed., vol. 133. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2019, pp. 9:1–9:23. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECRTS.2019.9

[22] I. Broster, A. Burns, and G. RodrÍguez-Navas, "Timing analysis of real-time communication under electromagnetic interference," *Real-Time Systems*, vol. 30, pp. 55–81, 2005.

[23] A. Gujarati, "Towards "ultra-reliable" CPS: Reliability analysis of distributed real-time systems," doctoralthesis, Technische Universität Kaiserslautern, 2020. [Online]. Available: https://nbn-resolving.de/urn:nbn:de:hbz:386-kluedo-61431

[24] A. Burns, R. I. Davis, S. Baruah, and I. Bate, "Robust mixed-criticality systems," *IEEE Transactions on Computers*, vol. 67, no. 10, pp. 1478–1491, 2018.

[25] A. Masrur, "A probabilistic scheduling framework for mixed-criticality systems," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016, pp. 1–6.

[26] S. Draskovic, R. Ahmed, P. Huang, and L. Thiele, "Schedulability of probabilistic mixed-criticality systems," *Real-Time Systems*, vol. 57, p. 397–442, 2021. [Online]. Available: https://doi.org/10.1007/s11241-021-09365-4

[27] B. Hu, K. Huang, P. Huang, L. Thiele, and A. Knoll, "On-the-fly fast overrun budgeting for mixed-criticality systems," in *2016 International Conference on Embedded Software (EMSOFT)*, 2016, pp. 1–10.

[28] A. F. Mills and J. H. Anderson, "A stochastic framework for multiprocessor soft real-time scheduling," in *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010, pp. 311–320.

[29] ——, "A multiprocessor server-based scheduler for soft real-time tasks with stochastic execution demand," in *2011 IEEE 17th International Conference on Embedded and Real-Time Computing Systems and Applications*, vol. 1, 2011, pp. 207–217.

[30] L. Abeni and G. Buttazzo, "Qos guarantee using probabilistic deadlines," in *Proceedings of 11th Euromicro Conference on Real-Time Systems. Euromicro RTS'99*, 1999, pp. 242–249.

[31] Z. Tong, S. Ahmed, and J. H. Anderson, "Holistically budgeting processing graphs," in *2023 IEEE Real-Time Systems Symposium (RTSS)*, 2023, pp. 27–39.

[32] S. Bozhko, F. Markovic, G. von der Bruggen, and B. B. Brandenburg, "What really is pWCET? a rigorous axiomatic proposal," in *2023 IEEE Real-Time Systems Symposium (RTSS)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec 2023, pp. 13–26. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/RTSS59052.2023.00012

[33] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic schedulability analysis techniques for real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 6, no. 1, pp. 04:1–04:53, 2019. [Online]. Available: https://drops-dev.dagstuhl.de/entities/document/10.4230/LITES-v006-i001-a004

[34] ——, "A survey of probabilistic timing analysis techniques for real-time systems," *Leibniz Transactions on Embedded Systems*, vol. 6, no. 1, pp. 03:1–03:60, 2019. [Online]. Available: https://drops-dev.dagstuhl.de/entities/document/10.4230/LITES-v006-i001-a003

[35] A. Cervin, "Analysis of overrun strategies in periodic control tasks," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 219–224, 2005, 16th IFAC World Congress. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474667016370884

[36] M. Maggio, A. Hamann, E. Mayer-John, and D. Ziegenbein, "Control-system stability under consecutive deadline misses constraints," in *32nd Euromicro Conference on Real-Time Systems (ECRTS 2020)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), M. Völp, Ed., vol. 165. Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020, pp. 21:1–21:24. [Online]. Available: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECRTS.2020.21

[37] F. P. Cantelli, "Sui confini della probabilità," *Atti del Congresso internazionale dei matematici*, vol. 6, pp. 47–59, 1928.

[38] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, p. 46–61, jan 1973. [Online]. Available: https://doi.org/10.1145/321738.321743

[39] R. Stafford, "Random vectors with fixed sum," 2006. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/9700-random-vectors-with-fixed-sum

[40] P. Emberson, R. Stafford, and R. Davis, "Techniques for the synthesis of multiprocessor tasksets," in *1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)*, July 2010, pp. 6–11.

[41] T. Oliphant, P. Peterson, and E. Jones, "SciPy," 2008. [Online]. Available: https://scipy.org/

[42] M. Lalee, J. Nocedal, and T. Plantenga, "On the implementation of an algorithm for large-scale equality constrained optimization," *SIAM Journal on Optimization*, vol. 8, no. 3, pp. 682–706, 1998. [Online]. Available: https://doi.org/10.1137/S1052623493262993

[43] J. Goh and J. H. Anderson, "Towards Principled Budget Enforcement in Real-Time Systems," in *2024 IEEE Real-Time Systems Symposium (RTSS)*. IEEE Computer Society, Dec 2024, pp. 256–266.

## APPENDIX A
## ADDITIONAL EXPERIMENTS

In this section, we present additional experimental results obtained by repeating the experiments of Sec. V with a higher number of samples and added configurations. Particularly, we generated 1,000 task sets per configuration instead of 200, and the possible utilization sums per processor were $\{0.40, 0.45, \ldots, 0.95\}$. All other task-set configuration and implementation details remained identical. It should be noted that these experiments were performed and recorded following the camera-ready submission of this paper. Thus, the published conference version of this paper [43] does not reference the results presented in this section.

All mean FIT bounds obtained by ConvexFIT and Fudge for each configuration are graphed in Fig. 5, whereas Fig. 6 shows the mean base-10 log difference in FIT bounds obtained by ConvexFIT and Fudge. The results of this additional set of experiments replicate the observations discussed in Sec. V.

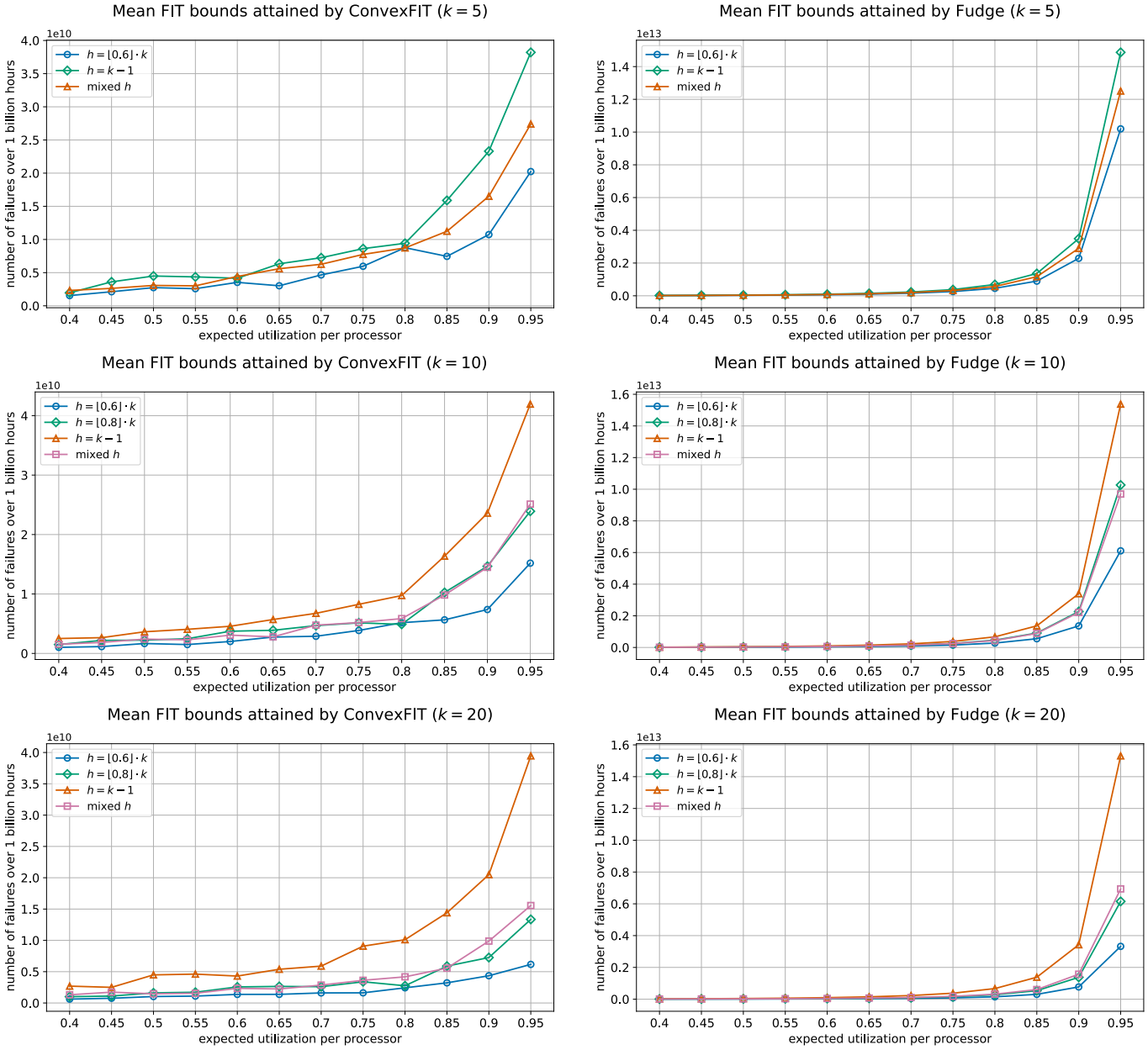


Fig. 5. Mean FIT bounds obtained from ConvexFIT and Fudge for task sets with $k = 5$ and 1,000 task sets per configuration. Note that $h = \lfloor 0.8 \rfloor \cdot k$ when $k = 5$ is identical to $h = k - 1$, and is thus omitted.

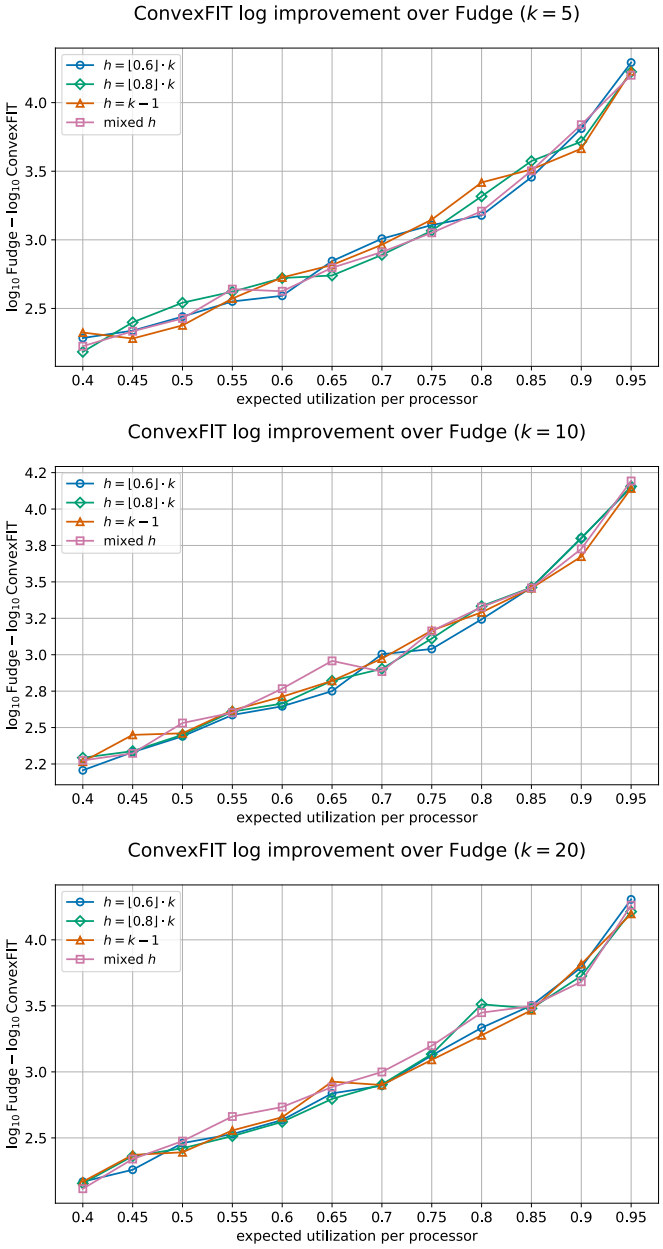


Fig. 6. Mean log (base 10) difference of mean FIT bounds obtained by ConvexFIT versus Fudge for task sets with $k \in \{5, 10, 20\}$ and 1,000 task sets per configuration.