Reducing Response-Time Bounds via Global Fixed Preemption Point EDF-like Scheduling

Joseph Goh
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, USA
jgoh@cs.unc.edu

James H. Anderson
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, USA
anderson@cs.unc.edu

***Abstract*—The fixed preemption point (FPP) model has been studied as an alternative to fully preemptive and non-preemptive models, as restricting preemptions to specific, predictable locations within a task's execution can simplify overhead analysis without disallowing preemptions entirely. Prior work has produced response-time analyses for global Earliest Deadline First (G-EDF) scheduling under the FPP model. However, scheduling decisions based solely on task deadlines may be too coarse-grained and may not lead to the lowest response times. In this paper, we propose global FPP EDF-like (G-FPP-EL) scheduling, which assigns a priority point in time for each non-preemptive region of a task. We adapt compliant-vector analysis (CVA) to our model and present general response-time bounds for G-FPP-EL schedulers. We then demonstrate that it is possible to design G-FPP-EL schedulers acheiving response-time bounds optimal under CVA and argue that such schedulers should replace global FPP EDF.**

***Index Terms*—Limited preemption models, fixed preemption points, G-EDF-like scheduling, soft real-time, real-time systems**

I. Introduction

Soft real-time (SRT) multiprocessor schedulers, which can guarantee bounded deadline tardiness, have been demonstrated to be useful for systems that do not require strict, hard real-time (HRT) completion of tasks [1]. Prior work has provided tardiness bounds for specific SRT schedulers including the global earliest deadline first (G-EDF) scheduler [2]. In [3], Erickson et al. proposed a linear-programming technique for designing G-EDF-like (GEL) schedulers such that the scheduler's parameters are optimized for reduced tardiness bounds. Using their technique, one may determine the optimal placement of *priority points* (PPs), which take the place of deadlines in scheduling decisions.

However, fully preemptive schedulers, including GEL schedulers, allow for frequent preemption of running tasks, introducing additional overheads that must be accounted for. These overheads can complicate timing analysis, resulting in worst-case execution time (WCET) estimates that are overly pessimistic. Non-preemptive schedulers, on the other hand, incur no such overheads but allow for long periods of priority inversions resulting in increased tardiness bounds.

Work was supported by NSF grants CPS 1837337, CPS 2038855, and CPS 2038960, ARO grant W911NF-20-1-0237, and ONR grant N00014-20-1-2698.

To address these concerns, limited preemptive scheduling techniques have been proposed as a method to balance the benefits and drawbacks of fully preemptive and non-preemptive scheduling [4], [5]. One such approach is the fixed preemption point model, wherein preemptions are limited to specific points in each task's execution. While G-EDF scheduling under this model has been previously explored [6], [7], scheduling based solely on deadlines may not acheive the lowest response-time bounds, and further optimization may be possible by prioritizing each non-preemptive section of a task individually.

In this paper, we propose global fixed preemption point EDF-like (G-FPP-EL) scheduling, which utilizes the fixed preemption-point model while updating PPs of executing tasks at their preemption points. We present general response-time analysis of G-FPP-EL schedulers, which can also model global fixed preemption point EDF (G-FPP-EDF) and global non-preemptive EDF (G-NP-EDF). By adapting the linear-programming technique of [3], we demonstrate it is possible to assign PPs such that we obtain lower response-time bounds than G-FPP-EDF. We argue that such G-FPP-EL schedulers designed in such a way should replace G-FPP-EDF.

A. Related Work

Leontyev and Anderson provide general analysis for SRT scheduling on multiprocessors in [2]. They observe that scheduling priorities for many algorithms, including G-EDF, can be modeled by assigning a *priority point (PP)* in time, with the scheduler choosing to schedule work with the earliest PP. For instance, G-EDF may be modeled by setting each job's PP equal to its deadline.

Using the technique of *compliant-vector analysis (CVA)*, first proposed in [8], Erickson et al. analyze *lateness*, defined as the difference between a job's deadline and completion time, of arbitrary GEL schedulers. They then present the global Fair Lateness (G-FL) scheduler, which provides the smallest lateness bounds of any GEL scheduler under CVA [3].

Erickson et al. further observe that the parameters that define a given GEL scheduler may be used to formulate a linear program which not only calculates response-time bounds but can also select PPs acheiving optimal bounds under CVA. They observe that various properties, such as average lateness and proportional lateness bounds, may also be minimized by selecting scheduler parameters using this technique.

The limited preemptive model was first proposed by Baruah [4], and studies have since designed HRT feasibility and schedulability tests for various limited preemptive models and schedulers [4], [5], [7], [9], [10]. The placement of fixed preemption points within a task's execution has been studied in the context of minimizing overheads while maintaining schedulability on uniprocessors [11], [12].

The splitting of tasks into mutliple, individually prioritized non-preemptive subregions is similar to the concept of job splitting. For instance, in [13], Erickson and Anderson provide overhead-aware analysis of GEL schedulers that evenly split job budgets into equally sized subjobs.

B. Contributions and Outline

In this paper, we present global limited preemptive EDF-like scheduling under a fixed preemption-point model, wherein tasks are modeled as a sequence of subtasks with varying costs and PPs. We provide response-time bounds for such schedulers by adapting CVA for our model, which allows the linear-programming technique of [3] to be used in designing G-FPP-EL schedulers with response times optimal under CVA. Finally, we demonstrate that G-FPP-EL schedulers can have lower response-time bounds under CVA when compared to global non-preemptive EDF.

The paper is organized as follows. Sec. II gives an overview of our fixed preemption-point task model. In Sec. III, we derive response-time bounds by adapting CVA for G-FPP-EL schedulers under the proposed task model. Sec. IV shows via experiments that G-FPP-EL schedulers chosen by applying [3]'s linear-programming technique can reduce maximum and mean response-time bounds under CVA when compared with G-FPP-EDF. Finally, in Sec. V, we discuss our results and how future work could improve or take advantage of our work.

II. Background

A. Task Model

We consider a system $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n arbitrary-deadline sporadic tasks running on $m \geq 2$ identical processors $\{\pi_1, \pi_2, \ldots, \pi_m\}$. Each task τ_i is defined by the parameters, (T_i, C_i, D_i). $T_i > 0$, the *period* of τ_i, denotes the minimum separation time between subsequent releases of jobs of τ_i. $C_i \leq T_i$ denotes the *worst-case execution time (WCET)* of τ_i. D_i is the *relative deadline* of each job of τ_i.

Each task τ_i consists of a sequence of f_i *subtasks* $\{\tau_{i,1}, \tau_{i,2}, \ldots, \tau_{i,f_i}\}$ corresponding to non-preemptive regions of τ_i. τ_i's preemption points define the beginning and end of each subtask, and subtasks are indexed in order of logical execution. Subtask $\tau_{i,j}$ corresponds to the jth non-preemptive region of τ_i, and τ_{i,f_i} denotes the final subtask of τ_i. Each subtask $\tau_{i,j}$ has a WCET $C_{i,j}$. $\tau_{i,j}$'s corresponding region of execution within a job J_i of τ_i is referred to as subjob $J_{i,j}$. Subjob $J_{i,j}$ is considered to be released if and only if J_i is released, but each $J_{i,j}$, excluding $J_{i,1}$, is eligible to execute only if its preceding subjob $J_{i,j-1}$ has finished.

We define the *utilization* of a given task τ_i as

$$U_i = \frac{C_i}{T_i}. \tag{1}$$

Note that, because $C_i \leq T_i$, we have

$$U_i \leq 1. \tag{2}$$

All tasks $\tau_i \in \boldsymbol{\tau}$ satisfy

$$\sum_{\tau_{i,j} \in \tau_i} C_{i,j} = C_i.$$

Each subtask $\tau_{i,j} \in \tau_i$ is assigned a proportional period $\phi_{i,j}$ such that, for all i, j,

$$U_i = \frac{C_{i,j}}{\phi_{i,j}}. \tag{3}$$

Note that, because $U_i \leq 1$, we have

$$C_{i,j} \leq \phi_{i,j}. \tag{4}$$

By (1) and (3), $\phi_{i,j} = C_{i,j}/U_{i,j} = T_i C_{i,j}/C_i$, so τ_i satisfies

$$\sum_{\tau_{i,j} \in \tau_i} \phi_{i,j} = T_i. \tag{5}$$

$\phi_{i,j}$ should not be confused with a subtask's true period, which is identical to T_i. There is no minimum separation time between subjobs of the same job.

Additionally, the "ideal" relative release time $\rho_{i,j}$ of a subtask is defined as

$$\rho_{i,j} = \sum_{k=1}^{j-1} \phi_{i,k}. \tag{6}$$

Ideally, if $J_{i,1}$ of task τ_i were released at the same time as J_i and each subjob $J_{i,k}$ were released $\phi_{i,k-1}$ time units after the release of the immediately preceding subjob $J_{i,k-1}$, $J_{i,j}$ would release $\rho_{i,j}$ time units after J_i. $\rho_{i,j}$ should not be confused with a $J_{i,j}$'s true release time, which is identical to the release time of J_i.

$\rho_{i,j}$ and $\phi_{i,j}$ will be used to concisely express the processor demand and interference to other tasks induced by $\tau_{i,j}$.

We assume

$$\sum_{\tau_i \in \boldsymbol{\tau}} U_i \leq m, \tag{7}$$

which was shown by Leontyev and Anderson to be a necessary condition for bounded tardiness [2]. We define

$$U^+ = \left\lceil \sum_{\tau_i \in \boldsymbol{\tau}} U_i \right\rceil. \tag{8}$$

We assume that $n > m$. If this is not the case, each task may be assigned its own processor and no job will have a response-time exceeding its WCET.

We use $C_{i,\max}$ to denote the largest WCET of all subtasks of τ_i such that

$$C_{i,\max} = \max_{\tau_{i,j} \in \tau_i} \{C_{i,j}\}. \tag{9}$$

We also use $C_{\max}$ to denote the largest WCET of all subtasks in $\boldsymbol{\tau}$ such that

$$C_{\max} = \max_{\tau_i \in \boldsymbol{\tau}} \{C_{i,\max}\}. \tag{10}$$

For each subtask $\tau_{i,j}$, we use $\Upsilon_{i,j}$ to denote its relative *priority point (PP)*. When a processor becomes available, the G-FPP-EL scheduler schedules the subjob with the earliest PP. $\Upsilon_{i,j}$ must satisfy

$$\Upsilon_{i,j} \geq 0. \tag{11}$$

For all τ_i and $j < k$ we require

$$\Upsilon_{i,j} \leq \Upsilon_{i,k}, \tag{12}$$

reflecting that subjob $J_{i,k}$ cannot have higher priority than a subjob $J_{i,j}$ that precedes it. For succinctness of later analysis, we define $Y_{i,j}$ as the relative PP of $\tau_{i,j}$ relative to its "ideal" release $\rho_{i,j}$ such that

$$\Upsilon_{i,j} = \rho_{i,j} + Y_{i,j}. \tag{13}$$

We define $Y_{\min}$ as the smallest $Y_{i,j}$ across all subtasks $\tau_{i,j}$ such that

$$Y_{\min} = \min_{\tau_{i,j} \in \boldsymbol{\tau}} \{Y_{i,j}\}. \tag{14}$$

for use in later analysis.

If a job has its absolute deadline at time d and completes execution at time t, its *lateness* is defined as $t - d$, and its *tardiness* is $\max\{0, t-d\}$. If the job is released at time r, its *response-time* is $t - r$. Similarly, a subjob with release time r and completion at t' has response-time $t' - r$. We upper-bound the lateness and response time of each τ_i by upper-bounding the response time of each $\tau_{i,j}$.

We denote R_i as the response-time bound of τ_i and L_i as its lateness bound. From the definition of lateness,

$$L_i = R_i - D_i. \tag{15}$$

$R_{i,j}$ denotes the response-time bound of $\tau_{i,j}$. Since a job of τ_i and its final subjob (which corresponds to τ_{i,f_i}) have identical release and completion time, we have

$$R_i = R_{i,f_i} \tag{16}$$

and consequently,

$$L_i = R_{i,f_i} - D_i. \tag{17}$$

For succinctness, we use vector notation for variables subscripted with i, j to refer to the set of all values for the subtasks of τ_i. For example, the values of $Y_{i,j}$ for each subtask of τ_i are written, $\vec{Y_i} = \langle Y_{i,1}, Y_{i,2}, \ldots, Y_{i,f_i} \rangle$. We model all time points and time-based variables such as PPs and WCETs as continuous values.

Notably, under this model, subjobs *cannot* be preempted once they have begun executing. Therefore, even if a subjob $J_{i,j}$ is ready at time t its PP occurs before another subjob $J_{k,l}$ executing at t, $J_{i,j}$ cannot interrupt $J_{k,l}$ and cannot be scheduled if there is no idle processor available at t. The $J_{i,j}$ experiences a *priority inversion* and is referred to as the *blocked subjob*, whereas $J_{k,l}$ is referred to as the *blocking subjob*.

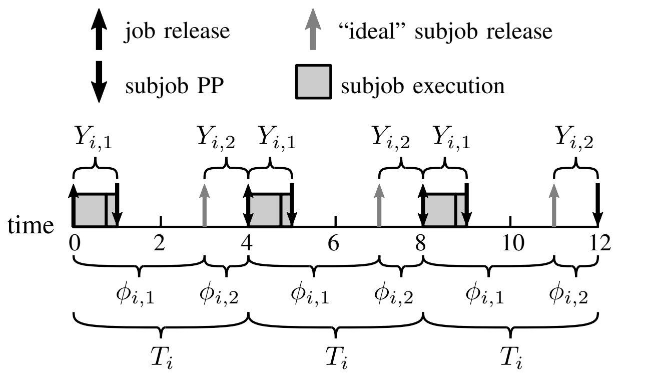


Fig. 1. Illustration of task parameters on timeline of τ_i releasing jobs as fast as possible where $T_i = 4$, $C_i = 1$, $\vec{C_i} = \{0.75, 0.25\}$, $\vec{\rho_i} = \{0, 3\}$, $\vec{\phi_i} = \{3, 1\}$, $\vec{\Upsilon_i} = \{1, 4\}$, and $\vec{Y_i} = \{1, 1\}$. All subjobs of a given job release simultaneously, regardless of their "ideal" release times and proportional periods.

B. Derivation of Subtasks from Preemption Points

Consider a conventionally defined sporadic task set $\boldsymbol{\tau} = \{\tau_1, \tau_2, \ldots, \tau_n\}$ with $\tau_i = (T_i, D_i, C_i)$, where each WCET C_i is measured from sequential, uninterrupted execution and does not include any preemption-related overheads. Assuming that a list of fixed preemption points are pre-provided, we can use the following procedure to define a subtask sequence τ'_i compatible with our model for each τ_i.

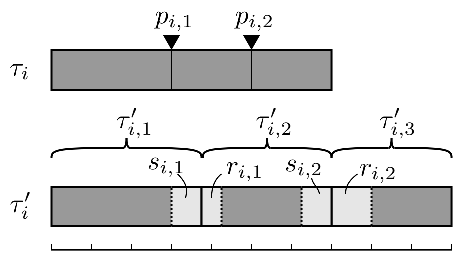


Fig. 2. Conversion of conventional task τ_i with preemption points to subtask sequence $\tau'_i = \{\tau'_{i,1}, \tau'_{i,2}, \tau'_{i,3}\}$. Preemption-related overheads $s_{i,1}, r_{i,1}, s_{i,2}, r_{i,2}$ are illustrated as delays added to τ_i's execution, but may encompass increased execution time due to task slowdown.

Suppose τ_i has $f_i - 1$ total preemption points. Let the jth preemption point be denoted $P_{i,j}$ and $p_{i,j} \in (0, C_i)$ represent the WCET of work in τ_i preceding $P_{i,j}$. Preemption points are ordered such that $p_{i,j} < p_{i,j+1}$ for all i and $j < f_i - 1$. The worst-case delays occurring due to $P_{i,j}$ are denoted $s_{i,j}$ and $r_{i,j}$, representing additional processing time incurred due to preempting or resuming execution of τ_i at $P_{i,j}$, respectively. For convenience, we also define $p_{i,0} = 0$ and $r_{i,0} = 0$ to mark the beginning of a task and $s_{i,f_i} = 0$ to mark the end.

C'_i, the total WCET of τ'_i, is defined as the sum of the WCET of τ_i and all its preemption overheads.

$$C'_i = C_i + \sum_{j=1}^{f_i - 1} (s_{i,j} + r_{i,j}).$$

Note that $C'_i \leq T_i$ is needed for bounded response-times as stated in (2).

From a sequence of $f_i - 1$ preemption points of τ_i, we can derive f_i subtasks of τ_i. For each integer j with $1 \leq j \leq f_i$, we define subtask $\tau_{i,j}$ to be the subtask whose execution ends at $P_{i,j}$, with corresponding WCET

$$C'_{i,j} = p_{i,j} - p_{i,j-1} + s_{i,j} + r_{i,j-1}.$$

For the remainder of this paper, we analyze task sets assumed to have been converted using the above procedure.

III. Compliant-Vector Analysis

In this section, we present CVA for G-FPP-EL schedulers under our task model, which will allow us to derive response-time bounds. While work in this section uses techniques similar to those used by Erickson et al. for GEL schedulers [3], additional considerations are needed to accommodate the non-preemptivity, varying sizes, and separate PPs of each subtask.

A. Linear Bound on Demand

When analyzing a task system, we must quantify the total processor demand that each task can require over given intervals, accounting for jobs with both releases and PPs within that interval.

If modeling an implicit-deadline sporadic system with PPs set equal to task deadlines such that, for all $\tau_{i,j} \in \tau_i$, $\Upsilon_{i,j} = D_i = T_i$, we can bound the demand that can be created by τ_i by simply multiplying U_i by the length of the interval. However, when a subtask's PP occurs before the succeeding subtask's "ideal" release time, i.e., $\Upsilon_{i,j} < \rho_{i,j}$ and $Y_{i,j} < \phi_{i,j}$, such a bound may underestimate the demand from τ_i.

An example is depicted in Fig. 3 where $U_i = 0.25$, but 0.75 units of demand are generated in fewer than $\phi_{i,1} = 3$ time units. The slope of the linear bound may be set equal to U_i, which may be interpreted as the average rate at which τ_i generates demand. However, in order avoid underestimating the demand at any time point, the line must be shifted upward by some fixed amount.

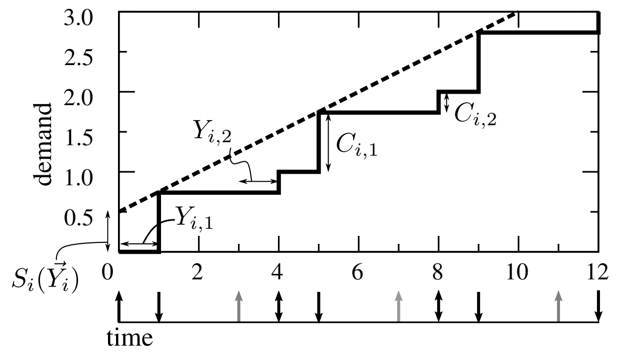


Fig. 3. Illustration of linear bound (dotted line) on demand generated (solid line) by task τ_i defined in Fig. 1. Timeline is identical to Fig. 1. Note that demand rises at subjob PPs.

Thus, we use the following term to account for the extra demand.

$$S_i(\vec{Y_i}) = \max_{\tau_{i,j} \in \tau_i} \left\{ C_{i,j} \cdot \max\left\{0, 1 - \frac{Y_{i,j}}{\phi_{i,j}}\right\}\right\} \tag{18}$$

This term is visually represented in the vertical axis in Fig. 3, showing the smallest amount the smallest amount the linear demand bound must be "raised" by in order not to underestimate, i.e., never fall below, the demand from $\tau_{i,j}$.

This notion is more formally stated as Lemma 3.2 and Lemma 3.3. We define $\tau_{i,*}$ as any subtask of τ_i satisfying

$$C_{i,*} \cdot \left(1 - \frac{Y_{i,*}}{\phi_{i,*}}\right) = \max_{\tau_{i,j} \in \tau_i} \left\{ C_{i,j} \cdot \left(1 - \frac{Y_{i,j}}{\phi_{i,j}}\right)\right\}. \tag{19}$$

$\tau_{i,*}$ can be described as the subtask of τ_i requiring the highest amount of potential extra demand to be accounted for, and thus is the subtask determining the value of $S_i(\vec{Y_i})$.

For simplicity, we use

$$S(\vec{Y}) = \sum_{\tau_i \in \boldsymbol{\tau}} S_i(\vec{Y_i}) \tag{20}$$

to represent the total such demand in $\boldsymbol{\tau}$.

The following lemma, proven in the appendix [14], bounds any $Y_{i,j}$ with respect to $Y_{i,*}$, $\phi_{i,*}$, and $\phi_{i,j}$. Lemma 3.1 enables us to derive the linear bounds on demand in the succeeding two lemmas.

Lemma 3.1. *For all $\tau_{i,j} \in \tau_i$ and any $\tau_{i,*}$ satisfying* (19)*,*

$$Y_{i,j} \geq Y_{i,*} - \phi_{i,*} + \phi_{i,j}.$$

We define DBF$(\tau_i, \vec{Y_i}, \ell)$, the demand bound function, as the total demand that a task τ_i can produce within an interval of length ℓ, as depicted by the solid line in Fig. 3. We use the next two lemmas, proven in the appendix [14], to provide linear bounds on the DBF. This will later allow us to bound response-times as a linear expression.

Lemma 3.2. *For any τ_i, if $\ell \geq Y_{i,*} - \phi_{i,*}$, then*

$$\mathit{DBF}(\tau_i, \vec{Y_i}, \ell) \leq U_i\ell + C_{i,*}\left(1 - \frac{Y_{i,*}}{\phi_{i,*}}\right). \tag{21}$$

Lemma 3.3. $\forall \ell \geq 0$*,*

$$\mathit{DBF}(\tau_i, \vec{Y_i}, \ell) \leq U_i\ell + S_i(\vec{Y_i}). \tag{22}$$

B. Compliant-Vector Analysis

We define a real value $x_{i,j} \geq 0$ for each $\tau_{i,j}$ such that each subjob of $\tau_{i,j}$ has a response time of at most

$$R_{i,j} = \rho_{i,j} + Y_{i,j} + x_{i,j} + C_{i,j}. \tag{23}$$

By (17), we can express the lateness bound of task τ_i as

$$L_i = \rho_{i,f_i} + Y_{i,f_i} + x_{i,f_i} + C_{i,f_i} - D_i. \tag{24}$$

We define

$$V_i(\vec{x_i}, \vec{Y_i}) = U_i \max_{\tau_{i,j} \in \tau_i} \{Y_{i,j} + x_{i,j}\} + C_{i,*}\left(1 - \frac{Y_{i,*}}{\phi_{i,*}}\right) - U_i C_{\max} - S_i(\vec{Y_i}). \tag{25}$$

and

$$G(\vec{x}, \vec{Y}) = \sum_{\tau_i \in \boldsymbol{\tau}} U_i C_{\max} + \sum_{\tau_i \in M_G} \max\left\{0, V_i(\vec{x_i}, \vec{Y_i})\right\}. \tag{26}$$

where M_G is the set of $U^+ - 1$ tasks that maximizes $G(\vec{x}, \vec{Y})$. $G(\vec{x}, \vec{Y})$ bounds the demand from certain critical tasks that can contribute to lateness of the system. We also define

$$H_{i,j}(Y_{i,j}) = \sum_{\tau_k \in M_i} \max\{0, C_{k,\max} - (Y_{i,j} + \rho_{i,j})\} \tag{27}$$

where M_i is the set of $m - U^+$ tasks in $\boldsymbol{\tau}$ excluding τ_i that maximizes $H_{i,j}(Y_{i,j})$. $H_{i,j}(Y_{i,j})$ bounds the demand from priority inversions that can contribute to lateness of $\tau_{i,j}$.

We will show at the end of this section, as Theorem III.1, that if $\vec{x}$ is *compliant*, $\forall \tau_i \in \boldsymbol{\tau}$, (23) is a correct bound.

Definition III.1. *(adapted from [3])* $\vec{x}$ *is* near-compliant *iff* $\forall i, j$,

$$x_{i,j} \geq \frac{S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j}) - C_{i,j}}{m}, \tag{28}$$

$$Y_{i,j} + \rho_{i,j} + x_{i,j} + C_{i,j} \leq Y_{i,j+1} + \rho_{i,j+1} + x_{i,j+1}, \tag{29}$$

and

$$Y_{i,f_i} + \rho_{i,f_i} + x_{i,f_i} + C_{i,f_i} - T_i \leq Y_{i,1} + x_{i,1}. \tag{30}$$

A *near-compliant vector* is compliant *iff* $\forall i, j$, $x_{i,j} \geq 0$ or $U^+ = 1$.

For the remainder of this section, we consider an arbitrary but fixed schedule. We bound the response-time of a subjob of interest, subjob $J_{i,j}$ of task τ_i, based on the inductive assumption that subjobs with higher priority than $J_{i,j}$ have response-times no greater than specified by (23).

We define *HP* as the set of all subjobs with priority at least that of $J_{i,j}$, i.e., all jobs in *HP* have PPs no later than that of $J_{i,j}$. Because we do not allow preemptions of subtasks, $J_{i,j}$ can experience *blocking*, where $J_{i,j}$ has been released and fewer than m jobs in *HP* are executing, but an already executing job of lower priority prevents $J_{i,j}$ from being scheduled. Furthermore, lower-priority work can block and delay work in *HP*. Thus, we also define *LP* as the set of all subjobs $J_{i,j}$ with lower priority than $J_{i,j}$. Jobs not in *HP* $\cup$ *LP* cannot be scheduled before $J_{i,j}$, and thus cannot prevent $J_{i,j}$ from executing.

We define $W_k(t)$ as the total amount of work in *HP* remaining at time t for jobs of τ_k. We define $W(t)$ as the total work remaining at time t for all jobs in *HP* such that

$$W(t) = \sum_{\tau_k \in \tau} W_k(t).$$

We denote $B_k(t)$ as the amount of work remaining at time t for jobs of τ_k in *LP* capable of blocking work in *HP* and $B(t)$ as the total such work across all tasks such that

$$B(t) = \sum_{\tau_k \in \tau} B_k(t).$$

For our analysis, we define the following time points. t_y denotes the absolute PP of $J_{i,j}$. t_r denotes the release time of $J_{i,j}$. t_b denotes the earliest time point such that, during $[t_b, t_y)$, at least U^+ processors are busy executing higher-priority work. I.e., at every time in $[t_b, t_y)$, at least U^+ processors are executing subjobs in *HP*. t_i denotes the earliest time such that at every time in $[t_i, t_b)$, fewer than U^+ processors are executing jobs from *HP*. These time points are illustrated in an example schedule in Sec. 4.

For each time point t, a superscripted '$-$' is used to denote any instant immediately before that time point such that no scheduling changes occur during the interval $[t^-, t)$.

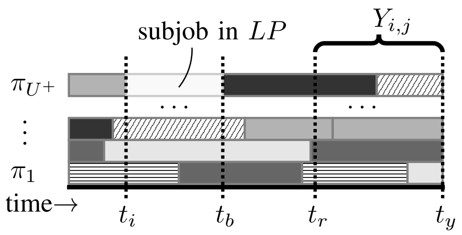


Fig. 4. Illustration of example schedule with time points t_i, t_b, t_r, and t_y.

We begin by bounding $W(t_L) + B(t_L)$, the amount of work in *HP* $\cup$ *LP* remaining at t_L that is capable of delaying $J_{i,j}$.

The following lemma bounds work in *HP* remaining at time t_b, if it exists.

Lemma 3.4. *If t_b exists, $\vec{x}$ is compliant, and every subjob $J_{k,l}$ of $\tau_{k,l}$ with higher priority than $J_{i,j}$ completes with response-time no greater than $\rho_{k,l} + Y_{k,l} + x_{k,l} + C_{k,l}$, then*

$$W(t_b) \leq \sum_{\tau_k \in \boldsymbol{\tau}} U_k(t_y - t_b) + S(\vec{Y}) + G(\vec{x}, \vec{Y}). \tag{31}$$

Proof. We examine the execution state of each task $\tau_k \in \boldsymbol{\tau}$ immediately before time t_b and derive and upper bound for $W_k(t_b)$, the amount of uncompleted work in *HP* remaining for τ_k. We can then bound $W(t_b)$, the total amount of work in *HP* remaining at t_b.

Case 1. **τ_k has no subjob in *HP* executing at time t_b^-.**

If at least one processor is idle at t_b^-, then the τ_k must not have any unfinished released work at t_b^-. Otherwise, since a processor is available, such work would have been scheduled before t_b.

If all processors are busy at t_b^-, by definition of t_b and because $U^+ \leq m$ by (7), at least one processor must be executing a subjob in *LP*. Each such subjob must have begun executing no later than $t_b - C_{\max}$ as they have not finished by t_b^-.

If τ_k has any unfinished released work in *HP* at t_b^-, such work cannot have been released at or before $t_b - C_{\max}$. Otherwise, that work would have been scheduled in place of one of the subjobs in *LP* executing at t_b^-.

Therefore, $W_k(t_b)$ must consist only of work from jobs released by τ_k after $t_b - C_{\max}$ and with PPs before t_y, giving us

$$\begin{aligned} W_k(t_b) &\leq \text{DBF}\left(\tau_k, \vec{Y_k}, t_y - (t_b - C_{\max})\right) \\ &\leq \{\text{By Lemma 3.3}\} \\ &\quad U_k\left(t_y - (t_b - C_{\max})\right) + S_k(\vec{Y_k}) \\ &= U_k(t_y - t_b) + U_k C_{\max} + S_k(\vec{Y_k}). \end{aligned} \tag{32}$$

Case 2. **τ_k has a subjob in *HP* executing at time t_b^- but there exists a time in $[t_i, t_b)$ when no job of τ_k is executing.**

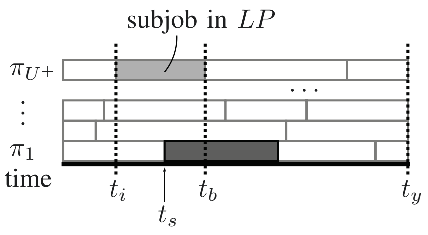


Fig. 5. Illustration of task τ_k and time point t_s in Case 2. If $J_{k,l}$ were released before the subjob in *LP*, it would have been scheduled at t_i instead.

In this case, denote t_s as the earliest time such that τ_k is executing continuously in $[t_s, t_b)$.

If at least one processor is idle at t_s^-, τ_k must not have any unfinished released work at t_s^-, as it otherwise would have began executing earlier than t_s.

If all processors are busy at t_s^-, because at most $U^+ - 1$ processors are executing work in *HP*, at least one processor is executing a subjob in *LP*. Such a subjob could have been released no later than $t_s - C_{\max}$. Any work of τ_k that is released and unfinished but not executing at t_s^- could only have been released after $t_s - C_{\max}$, as it would have otherwise been scheduled in place of a subjob in *LP* executing at t_s^-.

Therefore, $W_k(t_s)$ must consist only of work from jobs released by τ_k after $t_s - C_{\max}$ and with PPs before t_y, giving us

$$\begin{aligned} W_k(t_s) &\leq \text{DBF}\left(\tau_k, \vec{Y_k}, t_y - (t_s - C_{\max})\right) \\ &\leq \{\text{By Lemma 3.3}\} \\ &\quad U_k\left(t_y - (t_s - C_{\max})\right) + S_k(\vec{Y_k}) \\ &= U_k(t_y - t_s) + U_k C_{\max} + S_k(\vec{Y_k}). \end{aligned} \tag{33}$$

Because τ_k is executing continuously in $[t_s, t_b)$, and because the subjob of τ_k executing at t_b^- and any preceding subjobs are in *HP*,

$$\begin{aligned} W_k(t_b) &= W_k(t_s) - (t_b - t_s) \\ &\leq \{\text{By (33)}\} \\ &\quad U_k(t_y - t_s) + U_k C_{\max} + S_k(\vec{Y_k}) - (t_b - t_s) \\ &\leq \{\text{Because } U_k \leq 1\} \\ &\quad U_k(t_y - t_s) + U_k C_{\max} + S_k(\vec{Y_k}) - U_k(t_b - t_s) \\ &= U_k(t_y - t_b) + U_k C_{\max} + S_k(\vec{Y_k}). \end{aligned} \tag{34}$$

Case 3. **τ_k executes work in *HP* continuously in $[t_i, t_b)$ and $J_{k,l}$, the subjob of τ_k executing at t_b, has its PP at or after t_b.**

The release time of $J_{k,l}$ must be no earlier than $t_b - Y_{k,l} - \rho_{k,l}$ or its PP would be before t_b and Case 4 below would instead apply.

Thus, $W_k(t_s)$ must consist only of work from jobs released by τ_k at or after $t_b - (\rho_{k,l} + Y_{k,l})$ and with PPs before t_y. We also exclude work from subjobs of the same job preceding

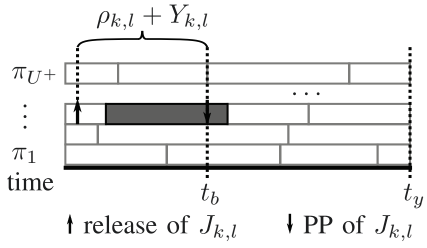


Fig. 6. Illustration of task τ_k in Case 3 with $J_{k,l}$ highlighted.

$J_{k,l}$, as they must have finished before t_s. Note that the length of the interval $[t_b - (\rho_{k,l} + Y_{k,l}), t_y]$ satisfies

$$\begin{aligned} t_y - (t_b - (\rho_{k,l} + Y_{k,l})) &= t_y - t_b + \rho_{k,l} + Y_{k,l} \\ &\geq \{\text{Because } t_b < t_y \text{ by definition} \\ &\qquad \text{and because } \rho_{k,l} \geq 0\} \\ &\quad Y_{k,l} \\ &\geq \{\text{By Lemma 3.1}\} \\ &\quad Y_{k,*} - \phi_{k,*} + \phi_{k,l} \\ &\geq Y_{k,*} - \phi_{k,*}, \end{aligned} \tag{35}$$

so Lemma 3.2 may be used to bound the demand created in the interval. Therefore we have

$$\begin{aligned} &W_k(t_b) \\ &\leq \text{DBF}\left(\tau_k, \vec{Y_k}, t_y - (t_b - (\rho_{k,l} + Y_{k,l}))\right) - \sum_{\alpha=1}^{l-1} C_{k,\alpha} \\ &\leq \{\text{By Lemma 3.2 and (35) and because} \\ &\qquad \textstyle\sum_{\alpha=1}^{l-1} C_{k,\alpha} \leq \sum_{\alpha=1}^{l-1} \phi_{k,\alpha} = \rho_{k,l} \text{ by (4) and (6)}\} \\ &\quad U_k\left(t_y - (t_b - (\rho_{k,l} + Y_{k,l}))\right) + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right) - \rho_{k,l} \\ &= U_k(t_y - t_b) + U_k\rho_{k,l} + U_k Y_{k,l} + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right) - \rho_{k,l} \\ &\leq \{\text{Because } U_k \leq 1 \text{ and so } U_k\rho_{k,l} - \rho_{k,l} \leq 0\} \\ &\quad U_k(t_y - t_b) + U_k Y_{k,l} + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right). \end{aligned} \tag{36}$$

Case 4. **τ_k executes work in *HP* continuously in $[t_i, t_b)$ and $J_{k,l}$, the subjob of τ_k executing at t_b, has its PP before t_b.**

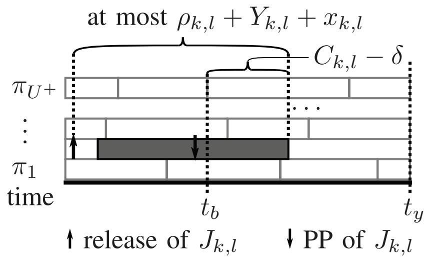


Fig. 7. Illustration of task τ_k in Case 4 with $J_{k,l}$ highlighted.

Since $J_{k,l}$'s PP is before t_b and $t_b < t_y$, $J_{k,l}$ is not $J_{i,j}$. Thus, by precondition of the lemma, $J_{k,l}$'s response time must be at most $\rho_{k,l} + Y_{k,l} + x_{k,l} + C_{k,l}$.

We define δ such that the remaining execution of $J_{k,l}$ at t_b is $C_{k,l} - \delta$. If $J_{k,l}$ runs for its full WCET, then δ is equal to $J_{k,l}$'s execution before t_b. Otherwise, δ may be greater.

$J_{k,l}$ finishes at $t_b + C_{k,l} - \delta$ and has a response time of at most $\rho_{k,l} + Y_{k,l} + x_{k,l} + C_{k,l}$, so $J_{k,l}$ must have been released no later than

$$\begin{aligned} &(t_b + C_{k,l} - \delta) - (\rho_{k,l} + Y_{k,l} + x_{k,l} + C_{k,l}) \\ &= t_b - \delta - \rho_{k,l} - Y_{k,l} - x_{k,l}. \end{aligned} \tag{37}$$

Thus, $W_k(t_b)$ must consist only of work from jobs released after that time and with PPs at or before t_y. We may exclude work from subjobs of the same job preceding $J_{k,l}$, as they must have finished before $J_{k,l}$ began executing. We also exclude the δ units of $J_{k,l}$'s work that are not part of its remaining execution.

Note that the length of the interval $[t_b - \delta - \rho_{k,l} - Y_{k,l} - x_{k,l}, t_y]$ satisfies

$$\begin{aligned} &t_y - (t_b - \delta - \rho_{k,l} - Y_{k,l} - x_{k,l}) \\ &= (t_y - t_b) + \delta + \rho_{k,l} + Y_{k,l} + x_{k,l} \\ &\geq \{\text{Because } t_b < t_y,\ \rho_{k,l} \geq 0, \text{ and } \delta \geq 0 \text{ by definition} \\ &\qquad \text{and } x_{k,l} \geq 0 \text{ by precondition of lemma and Def. III.1}\} \\ &\quad Y_{k,l} \\ &\geq \{\text{By Lemma 3.1}\} \\ &\quad Y_{k,*} - \phi_{k,*} + \phi_{k,l} \\ &\geq Y_{k,*} - \phi_{k,*}, \end{aligned} \tag{38}$$

so Lemma 3.2 may be used to bound the demand created in the interval. Therefore we have

$$\begin{aligned} &W_k(t_b) \\ &\leq \text{DBF}\left(\tau_k, \vec{Y_k}, t_y - (t_b - \delta - \rho_{k,l} - Y_{k,l} - x_{k,l})\right) \\ &\qquad - \sum_{\alpha=1}^{l-1} C_{k,\alpha} - \delta \\ &\leq \{\text{By Lemma 3.2 and (38) and because} \\ &\qquad \textstyle\sum_{\alpha=1}^{l-1} C_{k,\alpha} \leq \sum_{\alpha=1}^{l-1} \phi_{k,\alpha} = \rho_{k,l} \text{ by (4) and (6)}\} \\ &\quad U_k\left(t_y - (t_b - \delta - \rho_{k,l} - Y_{k,l} - x_{k,l})\right) \\ &\qquad + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right) - \rho_{k,l} - \delta \\ &= U_k(t_y - t_b) + U_k(\rho_{k,l} + \delta) + U_k Y_{k,l} + U_k x_{k,l} \\ &\qquad + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right) - (\rho_{k,l} + \delta) \\ &\leq \{\text{Because } U_k \leq 1 \text{ so } U_k(\rho_{k,l} + \delta) \leq \rho_{k,l} + \delta\} \\ &\quad U_k(t_y - t_b) + U_k(Y_{k,l} + x_{k,l}) + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right) \end{aligned} \tag{39}$$

Total remaining work at t_b: By definition of t_b, at most $U^+ - 1$ tasks can be in Cases 2-4. All other tasks must be in Case 1. By (36) and (39), Case 3 cannot cause $W_k(t_b)$ greater than Case 4. Since Cases 1 and 2 reach the same bound, $W(t_b)$ can be upper-bounded by selecting the set M of $U^+ - 1$ tasks maximizing

$$\begin{aligned} &\sum_{\tau_k \in M}\Bigg(U_k(t_y - t_b) + \max_{\tau_k \in \boldsymbol{\tau}}\Bigg\{U_k C_{\max} + S_k(\vec{Y_k}), \\ &\qquad U_k \max_{\tau_{k,l} \in \tau_k}\{Y_{k,l} + x_{k,l}\} + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right)\Bigg\}\Bigg) \\ &\qquad + \sum_{\tau_k \in \boldsymbol{\tau} \setminus M}\left(U_k(t_y - t_b) + U_k C_{\max} + S_k(\vec{Y_k})\right) \\ &= \sum_{\tau_k \in \boldsymbol{\tau}} U_k(t_y - t_b) + \sum_{\tau_k \in \boldsymbol{\tau}} S_k(\vec{Y_k}) + \sum_{\tau_k \in \boldsymbol{\tau}} U_k C_{\max} \\ &\qquad + \sum_{\tau_k \in M} \max\Bigg\{0, U_k \max_{\tau_{k,l} \in \tau_k}\{Y_{k,l} + x_{k,l}\} \\ &\qquad\qquad + C_{k,*}\left(1 - \frac{Y_{k,*}}{\phi_{k,*}}\right) - U_k C_{\max} - S_k(\vec{Y_k})\Bigg\} \\ &= \{\text{By the definitions of } V_i(\vec{x_i}, \vec{Y_i}) \text{ and } G(\vec{x}, \vec{Y}) \\ &\qquad \text{in (25) and (26)}\} \\ &\quad \sum_{\tau_k \in \boldsymbol{\tau}} U_k(t_y - t_b) + S(\vec{Y}) + G(\vec{x}, \vec{Y}) \end{aligned}$$

Thus, the lemma holds. □

In order to bound $B(t_y)$, the amount of lower-priority work remaining at t_y that can delay $J_{i,j}$ from being scheduled, we use the following lemma.

Lemma 3.5. *For all τ_k,*

$$B_k(t_y) \leq \max\{0, C_{k,l} - (Y_{i,j} + \rho_{i,j})\}. \tag{40}$$

Proof. Let subjob $J_{k,l}$ of subtask $\tau_{k,l}$ be a subjob in *LP* executing at time t_r. By definition of priority, a subjob in *LP* cannot begin execution at or after t_r unless $J_{i,j}$ has already been scheduled. Thus, τ_k cannot have any subjobs after $J_{k,l}$ that delay $J_{i,j}$ from being scheduled, so $J_{k,l}$ must be the only contributor to $B_k(t_y)$ Since $J_{k,l}$ was scheduled before t_r and continues executing until it finishes, its remaining work at t_y is at most

$$\begin{aligned} B_k(t_y) &\leq \max\{0, C_{k,l} - (t_y - t_r)\} \\ &= \max\{0, C_{k,l} - (Y_{i,j} + \rho_{i,j})\}. \end{aligned}$$

Thus, the lemma holds. □

Using Lemmas 3.4 and 3.5, for the case that t_b exists, we may bound $W(t_y) + B(t_y)$, the total remaining work at time t_y that can prevent $J_{i,j}$ from being scheduled.

Lemma 3.6. *If t_b exists, $\vec{x}$ is compliant, and every subjob $J_{k,l}$ of $\tau_{k,l}$ with higher priority than $J_{i,j}$ complete with response-time no greater than $\rho_{k,l} + Y_{k,l} + x_{k,l} + C_{k,l}$, then*

$$W(t_y) + B(t_y) \leq S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j}). \tag{41}$$

Proof. By the definition of t_b, at least U^+ processors are executing work in *HP* during the interval $[t_b, t_y]$. Thus, at least $U^+(t_y - t_b)$ units of work in *HP* must complete during $[t_b, t_y]$, so

$$\begin{aligned} W(t_y) &\leq W(t_b) - U^+(t_y - t_b) \\ &\leq \{\text{By Lemma 3.4}\} \\ &\quad \sum_{\tau_k \in \boldsymbol{\tau}} U_k(t_y - t_b) + S(\vec{Y}) + G(\vec{x}, \vec{Y}) \\ &\qquad - U^+(t_y - t_b) \\ &\leq \left\{\text{Because } \textstyle\sum_{\tau_k \in \boldsymbol{\tau}} U_k \leq U^+ \text{ by (8)}\right\} \\ &\quad S(\vec{Y}) + G(\vec{x}, \vec{Y}). \end{aligned} \tag{42}$$

Since at least U^+ processors are occupied with work in *HP* at time t_y, at most $m - U^+$ tasks can have a subjob in *LP* executing at t_y and contributing to $B(t_y)$. Thus, $B(t_y)$ can be bounded by selecting the set M_i of $m - U^+$ tasks other than τ_i that maximizes

$$\begin{aligned} B(t_y) &\leq \sum_{\tau_k \in M_i} B_k(t_y) \\ &\leq \{\text{By Lemma 3.5}\} \\ &\quad \sum_{\tau_k \in M_i} \max\{0, C_{k,l} - (Y_{i,j} + \rho_{i,j})\} \\ &= \{\text{By the definition of } H_{i,j}(Y_{i,j}) \text{ in (27)}\} \\ &\quad H_{i,j}(Y_{i,j}). \end{aligned} \tag{43}$$

Therefore, by (42) and (43),

$$W(t_y) + B(t_y) \leq S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j})$$

and the lemma holds. □

The following lemma bounds $W(t_y) + B(t_y)$ for the case that t_b does not exist.

Lemma 3.7. *If t_b does not exist, $\vec{x}$ is compliant, and every subjob $J_{k,l}$ of $\tau_{k,l}$ with higher priority than $J_{i,j}$ completes with response time no greater than $\rho_{k,l} + Y_{k,l} + x_{k,l} + C_{k,l}$, then*

$$W(t_y) + B(t_y) \leq S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j}). \tag{44}$$

Due to space constraints, the full proof of Lemma 3.7 is deferred to the appendix [14]. A proof sketch follows.

Proof sketch of Lemma 3.7: For each task τ_k not executing at t_r^-, $W_k(t_r)$ is bounded using derivation identical to Case 1 in the proof of Lemma 3.4 with t_r supplanting t_b. Similarly, $W_k(t_r)$ for each τ_k executing a subjob in *HP* at t_r^- is bounded using derivation identical to Cases 3 and 4 in the proof of Lemma 3.4 with t_y supplanting t_b. $B_k(t_r)$ for each τ_k executing a subjob in *LP* at t_r^- is most the cost of that subjob, as work in *LP* cannot be scheduled before $J_{i,j}$ after t_r. This also implies no task can have work remaining in both $W_k(t_r)$ and $B_k(t_r)$.

We show that if τ_k has a subjob $J_{k,l}$ executing at t_y^-, $W_k(t_r)$ in the case that $J_{k,l} \in$ *HP* is always greater than $B_k(t_r)$ when $J_{k,l} \in$ *LP*. Additionally, since t_b does not exist, at most $U^+ - 1$ processors are executing work in *HP* at t_r^-. By combining these results, we demonstrate that $W(t_y) + B(t_y)$ cannot exceed the bound given by the lemma.

Our next lemma bounds the response time of all subjobs.

Lemma 3.8. *If $\vec{x}$ is compliant and every subjob $J_{k,l}$ of $\tau_{k,l}$ with priority higher than $J_{i,j}$ completes with response time no greater than $\rho_{k,l} + Y_{k,l} + x_{k,l} + C_{k,l}$, then $J_{i,j}$ finishes with response time no greater than $\rho_{i,j} + Y_{i,j} + x_{i,j} + C_{i,j}$.*

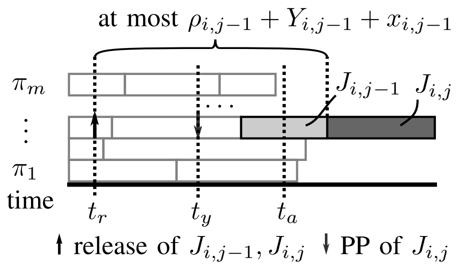
Proof. By Lemmas 3.6 and 3.7, the remaining work in *HP*, including $J_{i,j}$, and potentially interering work in *LP* at time t_y is at most $S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j})$. Therefore, there can be at most $S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j}) - C_{i,j}$ units of competing work scheduled before $J_{i,j}$.

All m processors may be occupied with completing such work before a processor becomes available for $J_{i,j}$ to be scheduled on. Thus, the latest time that a processor becomes available for $J_{i,j}$ is given by

$$t_a = t_y + \frac{S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j}) - C_{i,j}}{m}.$$

We consider the following cases.

Case 1. **At time t_a, $J_{i,j-1}$, the subjob immediately preceding $J_{i,j}$ in its subjob sequence, exists and has not completed.**



at most $\rho_{i,j-1} + Y_{i,j-1} + x_{i,j-1}$
π_m
$\vdots$
π_1
time
$J_{i,j-1}$
$J_{i,j}$
t_r
t_y
t_a
↑ release of $J_{i,j-1}, J_{i,j}$
↓ PP of $J_{i,j}$

Fig. 8. Illustration of subjob $J_{i,j}$ in Case 1. $J_{i,j}$ cannot be scheduled immediately at t_a because $J_{i,j-1}$ has not yet finished.

By the response-time bound stated in the precondition of the lemma and because $J_{i,j-1}$ has higher priority than $J_{i,j}$, $J_{i,j-1}$ finishes at most $Y_{i,j-1} + \rho_{i,j-1} + x_{i,j-1} + C_{i,j-1}$ time units after its release, which occurs simultaneously with $J_{i,j}$'s release. Since at least one processor is available for $J_{i,j}$ at t_a, $J_{i,j}$ begins execution immediately upon completion of $J_{i,j-1}$. Thus, the response time of $J_{i,j}$ is at most

$$\begin{aligned} &Y_{i,j-1} + \rho_{i,j-1} + x_{i,j-1} + C_{i,j-1} + C_{i,j} \\ &\leq \{\text{By (29) from Def. III.1}\} \\ &\quad Y_{i,j} + \rho_{i,j} + x_{i,j} + C_{i,j} \end{aligned}$$

as desired.

Case 2. **$J_{i,j}$ is the first subjob of its subjob sequence and at time t_a, J'_{i,f_i}, the preceding job's final subjob, has not completed.**

By the response-time bound stated in the precondition of the lemma and because J'_{i,f_i} was released at least T_i time units before $J_{i,j}$, J'_{i,f_i} finishes at most $Y_{i,f_i} + \rho_{i,f_i} + x_{i,f_i} + C_{i,f_i} - T_i$ time units after the release of $J_{i,j}$.

Since at least one processor is available for $J_{i,j}$ at t_a, $J_{i,j}$ finishes execution immediately upon completion of J'_{i,f_i}. Thus, the response time of $J_{i,j}$ is at most

$$\begin{aligned} & Y_{i,f_i} + \rho_{i,f_i} + x_{i,f_i} + C_{i,f_i} - T_i + C_{i,j} \\ \leq & \{\text{By (30) from Def. III.1 and because } \rho_{i,j} = 0\} \\ & Y_{i,j} + \rho_{i,j} + x_{i,j} + C_{i,j} \end{aligned}$$

as desired.

***Case 3.* All subjobs of τ_i preceding $J_{i,j}$ complete by t_a.**

In this case, $J_{i,j}$ may be scheduled immediately at t_a and finishes no later than

$$\begin{aligned} t_a + C_{i,j} = & \; t_y + \frac{S(\vec{Y}) + G(\vec{x}, \vec{Y}) + H_{i,j}(Y_{i,j}) - C_{i,j}}{m} + C_{i,j} \\ \leq & \{\text{By (28) from Def. III.1}\} \\ & t_y + x_{i,j} + C_{i,j}. \end{aligned}$$

Since t_y is the absolute priority point of $J_{i,j}$, $J_{i,j}$ finishes no more than $Y_{i,j} + \rho_{i,j} + x_{i,j} + C_{i,j}$ time units after its release.

Thus, in all cases, the response-time bound of the lemma holds for $J_{i,j}$. □

Using these lemmas, we can finally state and prove our response-time bound.

Theorem III.1. *If $\vec{x}$ is compliant, then no subjob of any subtask $\tau_{i,j}$ will have a response time exceeding $\rho_{i,j} + Y_{i,j} + x_{i,j} + C_{i,j}$.*

Proof. The theorem follows immediately from Lemma 3.8 by induction over all subjobs in the system, considered in order of decreasing priority. In the base case(s) of the absolute first subjob(s) to be scheduled, the precondition for Lemma 3.8 holds vacuously. □

C. Linear Optimization of Response-Time Bounds

The response-time bound from Thm. III.1 is based on a compliant vector $\vec{x}$. The condition for compliance, given in Def. III.1, is a linear combination of $\vec{x}$ and vector of PP assignments $\vec{Y}$. Consequently, we can adapt the linear-programming technique of [3] to find $\vec{Y}$ and compliant $\vec{x}$ resulting in response-time bounds that are optimal under CVA. Exact linear constraints and objective functions to calculate optimal values for $\vec{Y}$ and $\vec{x}$ are presented in the appendix [14] and are used in the following section.

IV. Experiments

In this section, we present experiments examining lateness bounds under CVA of G-FPP-EL schedulers. The schedulers we evaluate have PPs chosen by applying the linear-programming technique of [3]. We demonstrate that the resulting G-FPP-EL schedulers achieve improved lateness bounds over variations of G-FPP-EDF.

A. Experimental Design

a) Task Set Generation: We generated implicit-deadline task sets with per-task utilizations distributed uniformly or bimodally. Uniformly distributed utilizations were chosen to be *light*, *moderate*, or *heavy*, corresponding to samples from ranges $[0.001, 0.1]$, $[0.1, 0.4]$, or $[0.5, 0.9]$, respectively. Bimodally distributed utilizations were sampled uniformly from either $[0.001, 0.5]$ or $[0.5, 0.9]$ with respective probabilities of $8/9$ and $1/9$, $6/9$ and $3/9$, or $4/9$ and $5/9$.

Task periods were chosen to be *short*, *medium*, or *long*, corresponding to uniform integral samples from $[3, 33]$, $[10, 100]$, $[50, 250]$, respectively. Fixed preemption points were generated for each task, with their quantity per task being *none*, *low*, or *high*, corresponding to uniform integral samples from $[0, 0]$, $[0, 2]$, or $[2, 5]$, respectively. Preemption point locations were selected uniformly at random from the continous range of positions within a task's worst-case execution.

We considered a system with $m = 8$ processors, as clustered scheduling is typically preferred over global scheduling in systems with a high number of processors [15]. For each per-task utilization and preemption frequency combination, 250 task sets were generated for each total system utilization value in $\{1.25, 1.50, \cdots, 8.0\}$. We did not consider task systems with total utilization of at most one, as they are schedulable on a single processor. The maximum and mean lateness bounds were recorded for each task set and averaged.

b) G-FPP-EDF Approaches Used: We tested two approaches to applying G-EDF to our FPP task model. The first (EDF-1) is to select subtask PPs such that they are equal to the deadline of the task. That is, we set $\Upsilon_{i,j} = D_i$ and $Y_{i,j} = D_i - \rho_{i,j}$ for each $\tau_{i,j}$. The second (EDF-2) is to make all subtask PPs 'implicit' with respect to their "ideal" relative release times and proportional periods. That is, we set $\Upsilon_{i,j} = \rho_{i,j} + \phi_{i,j}$, i.e., $Y_{i,j} = \phi_{i,j}$.

For both approaches, we allow all PPs to be increased or decreased by a single constant. Such a change would not affect the ordering of subjob priorities and thus result in an identical schedule while potentially allowing for better bounds under CVA.

c) Optimization Criteria Used: We selected PPs for each task set using the linear-programming technique and optimization criteria described by [3]. The criteria tested are as follows:

- ML: Minimizing the maximum lateness across all tasks.
- AL: Minimizing average lateness across all tasks.
- ML-AL: Minimizing average lateness across all tasks *without exceeding the maximum lateness bound derived from ML.*

B. Observations

***Observation 1.* EDF-2 often outperformed EDF-1 when system utilization was high, with exceptions.**

Most configurations had a system utilization threshold above which EDF-2 acheived lower bounds under CVA than EDF-1, regardless of metric (maximum or mean). The opposite

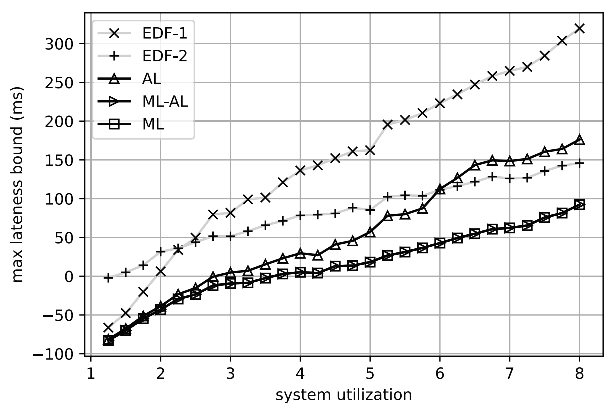


Fig. 9. Maximum lateness bounds for bimodal, moderate utilization, medium period, high preemption-point frequency task sets, averaged over 250 trials.

held true below this threshold. Interestingly, this pattern did not hold for some task sets with uniform, heavy per-task utilizations, where EDF-2 consistently fared worse.

Observation 2. **All optimized schedulers consistently obtained the lowest bounds for their respective optimization criterion.**

This observation shows empirically that G-FPP-EL scheduling optimized by the linear-programming technique of [3] can achieve lower bounds under CVA compared to G-FPP-EDF.

Observation 3. **Optimizing for AL always obtained lower mean lateness bounds than optimizing for ML-AL.**

This observation mirrors one made in [3], and implies that we cannot achieve the lowest possible maximum lateness bounds and the lowest mean lateness bounds simultaneously. I.e., past a certain point, a decrease in maximum lateness requires an increase to the mean lateness and vice versa.

These observations are exemplified by Fig. 9 and further supported by graphs included in the appendix [14].

V. Discussion

While we have shown that our analysis can be leveraged to design G-FPP-EL schedulers with reduced response-time bounds when compared with G-FPP-EDF, we do not address the issue of selecting and enabling preemption points based on their positions and overheads.

Because of the increase in total WCET and utilization caused by each preemption point, it may be the case that enabling a higher number of preemption points results in worse response-time bounds despite increased granularity in PP selection. Additionally, some preemption points, such as those in low-cost tasks or close to other preemption points, may be of relatively little value while still inflating WCETs.

Future research addressing this issue could help guide the selection of effective preemption points and quantify the benefits and trade-offs of using G-FPP-EL scheduling over non-preemptive GEL scheduling.

VI. Conclusion

We have presented response-time bounds for G-FPP-EL schedulers under a fixed preemption-point model using CVA. Our analysis can be inclusive of preemption-related overheads and simplifies overhead analysis significantly when compared with fully preemptive schedulers. We argue that G-FPP-EL schedulers can replace G-EDF and GEL schedulers, especially in systems where fully preemptive scheduling can require unacceptably high overheads and WCET inflation. We have demonstrated that G-FPP-EL schedulers whose parameters are chosen using a linear-programming technique can obtain lower response-time bounds than G-FPP-EDF. In scenarios where an offline computation step is practical, we believe such G-FPP-EL schedulers should be used in place of G-FPP-EDF.

References

[1] C. J. Kenna, J. L. Herman, B. B. Brandenburg, A. F. Mills, and J. H. Anderson, "Soft real-time on multiprocessors: Are analysis-based schedulers really worth it?" in *2011 IEEE 32nd Real-Time Systems Symposium*, 2011, pp. 93–103.

[2] H. Leontyev and J. H. Anderson, "A unified hard/soft real-time schedulability test for global edf multiprocessor scheduling," in *2008 Real-Time Systems Symposium*, Nov 2008, pp. 375–384.

[3] J. P. Erickson, J. H. Anderson, and B. C. Ward, "Fair lateness scheduling: Reducing maximum lateness in g-edf-like scheduling," *Real-Time Syst.*, vol. 50, no. 1, p. 5–47, jan 2014.

[4] S. Baruah, "The limited-preemption uniprocessor scheduling of sporadic task systems," in *17th Euromicro Conference on Real-Time Systems*, 2005, pp. 137–144.

[5] B. Chattopadhyay and S. Baruah, "Limited-preemption scheduling on multiprocessors," in *22nd International Conference on Real-Time Networks and Systems*. New York, NY, USA: Association for Computing Machinery, 2014, p. 225–234.

[6] Q. Zhou, G. Li, J. Li, C. Deng, and L. Yuan, "Response time analysis for tasks with fixed preemption points under global scheduling," *ACM Transactions on Embedded Computing Systems*, vol. 18, no. 5, oct 2019.

[7] A. Thekkilakattil, S. Baruah, R. Dobrin, and S. Punnekkat, "The global limited preemptive earliest deadline first feasibility of sporadic real-time tasks," in *2014 26th Euromicro Conference on Real-Time Systems*, 2014, pp. 301–310.

[8] J. Erickson, U. Devi, and S. Baruah, "Improved tardiness bounds for global edf," in *2010 22nd Euromicro Conference on Real-Time Systems*, 2010, pp. 14–23.

[9] M. Bertogna and S. Baruah, "Limited preemption edf scheduling of sporadic task systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 579–591, 2010.

[10] A. Thekkilakattil, R. I. Davis, R. Dobrin, S. Punnekkat, and M. Bertogna, "Multiprocessor fixed priority scheduling with limited preemptions," in *23rd International Conference on Real Time and Networks Systems*. New York, NY, USA: Association for Computing Machinery, 2015, p. 13–22.

[11] M. Bertogna, G. Buttazzo, M. Marinoni, G. Yao, F. Esposito, and M. Caccamo, "Preemption points placement for sporadic task sets," in *2010 22nd Euromicro Conference on Real-Time Systems*, 2010, pp. 251–260.

[12] M. Bertogna, O. Xhani, M. Marinoni, F. Esposito, and G. Buttazzo, "Optimal selection of preemption points to minimize preemption overhead," in *2011 23rd Euromicro Conference on Real-Time Systems*, 2011, pp. 217–227.

[13] J. P. Erickson and J. H. Anderson, "Reducing tardiness under global scheduling by splitting jobs," in *2013 25th Euromicro Conference on Real-Time Systems*, 2013, pp. 14–24.

[14] J. Goh and J. H. Anderson, "Reducing response-time bounds via global fixed preemption point edf-like scheduling," 2023, full version with appendix. [Online]. Available: https://jamesanderson.web.unc.edu/papers

[15] B. B. Brandenburg, "Scheduling and locking in multiprocessor real-time operating systems," Ph.D. dissertation, USA, 2011.

[16] W. Ogryczak and A. Tamir, "Minimizing the sum of the k largest functions in linear time," *Information Processing Letters*, vol. 85, no. 3, pp. 117–122, 2003.