Fair Lateness Scheduling: Reducing Maximum Lateness in G-EDF-like Scheduling

Jeremy P. Erickson, James H. Anderson, and Bryan C. Ward

Department of Computer Science, University of North Carolina at Chapel Hill*

July 4, 2013

Abstract

In prior work on soft real-time (SRT) multiprocessor scheduling, tardiness bounds have been derived for a variety of scheduling algorithms, most notably, the global earliest-deadline-first (G-EDF) algorithm. In this paper, we devise G-EDF-like (GEL) schedulers, which have identical implementations to G-EDF and therefore the same overheads, but that provide better tardiness bounds. We discuss how to analyze these schedulers and propose methods to determine scheduler parameters to meet several different tardiness bound criteria. We employ linear programs to adjust such parameters to optimize arbitrary tardiness criteria, and to analyze lateness bounds (lateness is related to tardiness). We also propose a particular scheduling algorithm, namely the global fair lateness (G-FL) algorithm, to minimize maximum absolute lateness bounds. Unlike the other schedulers described in this paper, G-FL only requires linear programming for analysis. We argue that our proposed schedulers, such as G-FL, should replace G-EDF for SRT applications.

1 Introduction

Kenna et al. (2011) demonstrated that analysis-based soft real-time (SRT) schedulers are useful on multiprocessor systems when bounded deadline tardiness is acceptable. Previous work on bounded tardiness (Devi and Anderson, 2008; Erickson et al., 2010b; Leontyev and Anderson, 2010) has provided tardiness bounds for specific schedulers. For many applications, such as the video processing described by Kenna et al. (2011), the output of SRT tasks can be stored in a buffer, and the buffer read at the desired rate to simulate hard real-time (HRT) completion. The sizes of such buffers can be determined from tardiness bounds.

Smaller buffers are sufficient to provide equivalent performance for systems with smaller tardiness bounds. These smaller buffers may reduce the cost of the system, and may enable more applications to run on resource-constrained devices such as smartphones and tablets. In this paper, we discuss schedulers that have an identical implementation to the previously studied (Devi and Anderson, 2008; Erickson et al., 2010b) *global earliest deadline first* (*G-EDF*) scheduler, but that have better tardiness bounds. Specifically, we consider methods for defining the parameters such

*Work supported by NSF grants CNS 1016954, CNS 1115284, and CNS 1239135; ARO grant W911NF-09-1-0535; and AFRL grant FA8750-11-1-0033. The third author was supported by an NSF Graduate Research Fellowship.

schedulers and analyzing tardiness under them. With such methods, it is possible that jobs of some tasks can be guaranteed to complete by times *before* their deadlines. Therefore, instead of evaluating tardiness (which is defined to be zero for such tasks), we evaluate *lateness*, defined as the difference between deadline and completion time. Analyzing lateness rather than tardiness for many applications provides a modest advantage, but the results in this paper are easier to state for lateness than for tardiness, and tardiness can be trivially calculated from lateness.

We propose a specific scheduler — the *global fair lateness* (*G-FL*) scheduler — that minimizes maximum lateness bounds, as well as a general linear programming technique to minimize other criteria related to lateness. In addition, we present the results of experiments conducted to show the improvements available using our methods.

Past Work. Leontyev and Anderson (2010) provided general analysis for SRT scheduling. They observed that scheduling priorities for most algorithms can be modeled by giving each job a *priority point* (*PP*) in time, with the scheduler always selecting for execution the job with the earliest PP (with appropriate tie-breaking). For example, fixed-priority scheduling can be modeled by assigning all jobs of each task a single PP near the beginning of the schedule. G-EDF can be modeled by defining the absolute deadline of a job as its PP. Leontyev and Anderson also defined a class of *window-constrained* scheduling algorithms, which provide bounded tardiness with no utilization loss. Leontyev et al. (2009) analyzed response times for many schedulers, including a class of *G-EDF-like* (*GEL*) schedulers, in which the PP of each job is defined by a per-task constant after the job's release. Although these papers provide analysis for a large class of scheduling algorithms, they do not provide a method to determine the best scheduling algorithm for an application with particular tardiness requirements.

One algorithm that has been widely studied is G-EDF itself. Although G-EDF is known to be suboptimal for HRT scheduling on multiprocessors, it is attractive for several reasons. Optimal algorithms, such as those described in (Anderson and Srinivasan, 2004; Baruah et al., 1996; Megel et al., 2010; Regnier et al., 2011), either cause tasks to experience frequent preemptions and migrations, resulting in prohibitive overheads, or are difficult to implement in practice. In contrast, Bastoni et al. (2010) demonstrated that the overheads caused by G-EDF are reasonable when it is used on a moderate number of processors. Furthermore, unlike optimal algorithms, G-EDF has the desirable property that it is a *job-level static-priority* (*JLSP*) algorithm. The JLSP property is required for most known work on real-time synchronization algorithms (Brandenburg, 2011). Moreover, Srinivasan and Baruah (2002) proved that no HRT-optimal algorithm can be JLSP. Similarly, several promising non-HRT-optimal schedulers such as the earliest-deadline-until-zero-laxity (EDZL) algorithm (Lee, 1994; Baker et al., 2008) are also not JLSP.

In this work, we modify G-EDF to improve its tardiness bounds. In order to do so, we use the technique of *compliant-vector analysis* (*CVA*), first proposed by Erickson et al. (2010a). In Section 3, we provide a detailed descrip-

tion of CVA as applied to arbitrary GEL schedulers. Although the same systems can be analyzed using the method described by Leontyev et al. (2009), CVA can provide tighter bounds.

Our Contribution. In this paper, we demonstrate that CVA is useful for determining an appropriate scheduler within the class of GEL schedulers. All GEL schedulers are JLSP and have the same overheads as G-EDF with arbitrary deadlines. However, we show that we can improve tardiness bounds by selecting a different GEL scheduler rather than G-EDF, such as the G-FL scheduler mentioned earlier.

We provide a technique that employs linear programming to allow a system designer to minimize the linear functions of lateness bounds that matter most to a system designer. For example, the system designer can determine PPs that will minimize the average lateness bound across a task system. With this technique, we can use a linear program (LP) solver to determine PPs for the GEL scheduler that meets the aforementioned constraints. We also show that G-FL minimizes the maximum lateness bound (under CVA) for all tasks. Unlike the other algorithms we propose, G-FL requires the use of an LP solver only for its analysis, not to determine PPs. As implied by its name, G-FL provides the same lateness bound for all tasks. G-FL has several useful properties. If a task system can be proven HRT schedulable by any GEL algorithm under CVA, then it can be proven schedulable with G-FL. If the task system cannot be proven HRT schedulable, then G-FL will still provide a fair allocation of lateness bounds to all tasks. However, our work does not demonstrate that a better GEL scheduler cannot exist given better analysis. It also does not preclude the possiblility that an *individual* task may have a lower CVA lateness bound with a scheduler other than G-FL. In fact, several of the LP-based schedulers we propose often have lower CVA lateness bounds for individual tasks than G-FL, as shown experimentally in Section 7.

Related Work. Because our work involves altering the PPs used by the scheduler, it superficially resembles the work of Lee et al. (2011), in which the deadlines of some tasks are shortened at design time to create "contention-free slots," which allow the priorities of some jobs to be lowered during runtime, increasing system schedulability. However, in their work, the actual deadlines by which jobs must complete are altered, which is not true of our work. Furthermore, their work requires modifying G-EDF in a manner that adds additional runtime overhead and removes the JLSP property, while our work does not.

In an HRT context, Back et al. (2012) examined a class of GEL schedulers that generalizes G-FL, and Chwa et al. (2012) examined general GEL schedulers. In both cases, the schedulers maintain the benefits described above, including the JLSP property. However, there are task systems using just over half of system capacity that cannot meet all their deadlines using any JLSP scheduler (Srinivasan and Baruah, 2002), so some system utilization is necessarily lost in the HRT case. Our work supports all task systems that do not overutilize the system, at the cost of allowing

bounded lateness.

Organization. In Section 2, we describe the task model under consideration and define basic terms. In Section 3, we present CVA as applied to our work. In Section 4, we present the basic linear programming technique for working with CVA, and we use it to prove several properties about CVA. In Section 5, we present G-FL and prove that it minimizes the maximum lateness bound over all tasks. Afterwards, in Section 6, we describe the use of linear programming to define GEL schedulers to optimize values of other functions of lateness. In Section 7, we present experiments comparing tardiness bounds and tardiness in computed schedules for G-FL, G-EDF, and other LP-derived GEL schedulers. These experiments show that G-FL and LP-derived GEL schedulers can provide significantly lower tardiness bounds than G-EDF.

2 Task Model

For the reader's convenience, a table of notation used in this paper is provided in Appendix A. We consider a system $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of *n arbitrary-deadline* sporadic tasks $\tau_i = (T_i, C_i, D_i)$ running on $m \geq 2$ processors, where T_i is the minimum separation time between subsequent releases of jobs of τ_i, $C_i \leq T_i$ is the worst-case execution time of any job of τ_i, and $D_i \geq 0$ is the relative deadline of each job of τ_i. We let $C_{\max}$ denote the largest worst-case execution time in the system, i.e.,

$$C_{\max} = \max_{\tau_i \in \tau} \{C_i\}. \tag{1}$$

We use

$$U_i = \frac{C_i}{T_i} \tag{2}$$

to denote the *utilization* of τ_i. Because $C_i \leq T_i$,

$$U_i \leq 1. \tag{3}$$

All quantities are real-valued. We assume that

$$\sum_{\tau_i \in \tau} U_i \leq m, \tag{4}$$

which was demonstrated in Leontyev and Anderson (2010) to be a necessary condition for bounded tardiness. We define

$$U^+ = \left\lceil \sum_{\tau_i \in \tau} U_i \right\rceil. \tag{5}$$

We assume that $n > m$. If this is not the case, then each task can be assigned its own processor, and no job of each τ_i will have a response time exceeding C_i.

If a job has an absolute deadline d and completes execution at time t, then its *lateness* is $t - d$, and its *tardiness* is $\max\{0, t - d\}$. Its *proportional lateness* (*tardiness*) is simply its lateness (tardiness) divided by its relative deadline. If such a job is released at time r, then its *response time* is $t - r$. We bound these quantities on a per-task basis, i.e., for each τ_i, we consider upper bounds on these quantities that apply to all jobs of τ_i.

We use for each τ_i the notation Y_i to refer to its relative PP, R_i to refer to its response time bound, L_i to refer to its lateness bound, and I_i to refer to its proportional lateness bound. From the definition of lateness,

$$L_i = R_i - D_i. \tag{6}$$

From the definition of proportional lateness,

$$I_i = L_i / D_i. \tag{7}$$

We assume

$$\forall i, Y_i \geq 0. \tag{8}$$

For all variables subscripted with an i, we also use vector notation to refer to the set of all values for the task system. For example, $\vec{T} = \langle T_1, T_2, \ldots T_n \rangle$.

3 Basic Compliant-Vector Analysis

In this section, we present the CVA necessary to analyze arbitrary GEL schedulers, including G-FL. We first describe the lateness bounds for an arbitrary GEL scheduler, providing a general condition that results in correct lateness bounds, and then prove it correct.

Erickson et al. (2010b) presented CVA for G-EDF with arbitrary deadlines. By using $\vec{Y}$, the relative PPs, in place of $\vec{D}$, the relative deadlines, we can use the same existing analysis to analyze arbitrary GEL schedulers. Much of the analysis in this section therefore closely follows the analysis by Erickson et al. (2010b).

We will first provide definitions needed to specify our lateness bounds, accompanied by some basic intuition. We will then prove our bounds correct.

While analyzing a task system, we need to account for the total processor demand that each task can require over certain intervals, accounting only for jobs that have both releases and PPs within that interval. We will use a linear upper bound for this quantity. In the context of implicit-deadline systems (where $D_i = T_i$), in order to analyze τ_i,

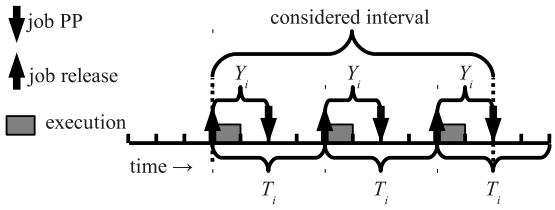


Figure 1: Illustration of the worst-case arrival pattern for analyzing demand within an interval.

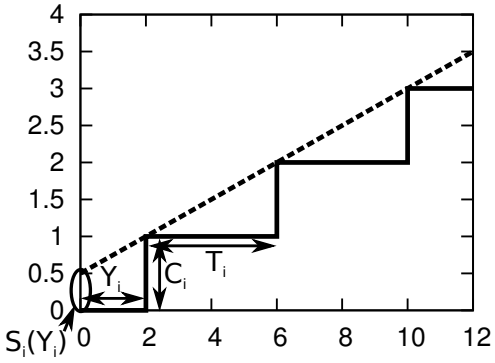


Figure 2: Illustration of the technique for bounding the demand for the task in Figure 1.

Erickson et al. (2010a) simply multiplied the length of such an interval by the utilization of the task being analyzed. When $Y_i < T_i$, this technique may underestimate the demand, as depicted in Figure 1, where $3C_i$ units of demand are present in an interval shorter than $3T_i$ units. We will use the term

$$S_i(Y_i) = C_i \cdot \max\left\{0, 1 - \frac{Y_i}{T_i}\right\}, \tag{9}$$

illustrated in Figure 2, to account for this extra demand. We will also use the total of such demand across the whole system,

$$S(\vec{Y}) = \sum_{\tau_i \in \tau} S_i(Y_i). \tag{10}$$

We define our lateness bounds recursively, defining for each τ_i a real value x_i such that each job of τ_i has a response time of at most

$$R_i = Y_i + x_i + C_i \tag{11}$$

and thus an lateness bound of

$$L_i = Y_i + x_i + C_i - D_i. \tag{12}$$

As part of our lateness expression, we define the term

$$G(\vec{x},\vec{Y}) = \sum_{(U^+-1) \text{ largest}} (x_i U_i + C_i - S_i(Y_i)), \tag{13}$$

which accounts for demand from certain critical tasks that can contribute to the lateness in the system. $G(\vec{x})$ accounts for demand over the same intervals as $S(\vec{Y})$, and like $S(\vec{Y})$, $G(\vec{x})$ actually accounts for demand in excess of what would be predicted simply from using utilization multiplied by the interval length. Furthermore, as shown in the analysis below, a task that contributes to $G(\vec{x})$ does not actually need to contribute to $S(\vec{Y})$. In order to simplify the expression for the bounds, we therefore include the negative $S_i(Y_i)$ term in (13) so that $S(\vec{Y})$ remains a sum across all tasks.

We will show at the end of this section, as Theorem 1, that if $\vec{x}$ is *compliant* (defined below) then (11) is a correct response time bound for each τ_i. The analysis is most interesting for the common case when $U^+ > 1$, but our definition also accounts for the simpler degenerate case that $U^+ = 1$.

Definition 1. *$\vec{x}$ is* near-compliant *iff*

$$\forall i, x_i \geq \frac{G(\vec{x},\vec{Y}) + S(\vec{Y}) - C_i}{m}. \tag{14}$$

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

For the remainder of this section, we consider an arbitrary but fixed schedule. The eventual proof of Theorem 1 will proceed by induction over the jobs of that schedule considered in priority order. We will bound the response time of a job J_i of task τ_i, based on the inductive assumption that jobs with higher priority than J_i have response times no greater than specified by (11).

We define H as the set of all jobs with priority at least that of J_i. By the definition of priority, we do not need to consider any work due to jobs not in H when analyzing the response time of J_i. We define $W_j(t)$ as the total amount of work remaining at time t for jobs of τ_j in H. We define $W(t)$ as the total amount of work remaining at time t for all jobs in H, i.e., $W(t) = \sum_{\tau_j \in \tau} W_j(t)$.

Several important time instants are depicted in Figure 3. We denote as y_i the PP of J_i. We denote as t_b ("busy") the earliest time instant such that during every time in $[t_b, y_i)$, at least U^+ processors are executing jobs from H. We denote as t_d ("idle") the earliest time instant such that during every time in $[t_d, t_b)$, fewer than U^+ processors are executing jobs from H.

Throughout our proofs, we will refer to the total demand that a task τ_i can produce within an interval of length ℓ, accounting for jobs that have both release times and PPs within the interval. This quantity is denoted $\text{DBF}(\tau_i, Y_i, \ell)$.

The next two lemmas, both proved in the appendix, provide slightly pessimistic bounds on $\text{DBF}(\tau_i, Y_i, \ell)$. Lemma 1

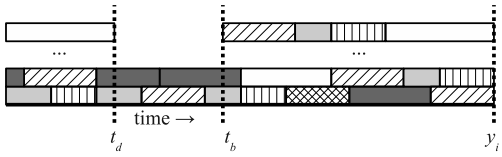
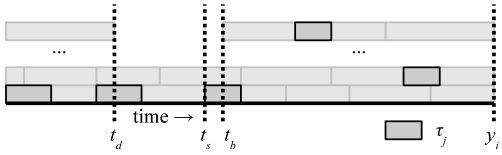
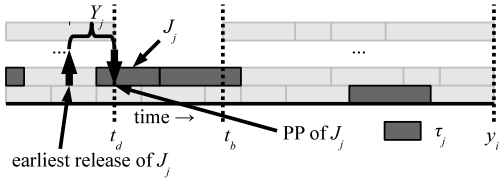


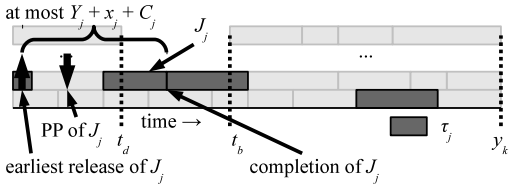
Figure 3: Example schedule depicting t_d, t_b, and y_i. Different lines represent different processors, and different hatchings or shadings represent different tasks.



(a) τ_j not running for entire idle interval.



(b) τ_j running for entire idle interval, J_j not past PP at t_d.



(c) τ_j running for entire idle interval, J_j past PP at t_d.

Figure 4: Proof details for Thm. 1 and supporting lemmas. Different lines in each figure represent different processors. In each figure, the jobs of the task under consideration are outlined in black, while any other jobs are outlined in gray.

provides a less pessimistic bound than Lemma 2 when $Y_i > T_i$, but it only applies for intervals of length greater than Y_i.

Lemma 1. *If $\ell \geq Y_i$, then*

$$DBF(\tau_i, Y_i, \ell) \leq U_i \ell + C_i \left(1 - \frac{Y_i}{T_i}\right).$$

Lemma 2. $\forall \ell \geq 0, DBF(\tau_i, Y_i, \ell) \leq U_i \ell + S_i(Y_i)$.

Because at least U^+ processors are assumed to be busy in $[t_b, y_i)$, we know that at least $U^+(y_i - t_b)$ units of work will be completed during that interval. Therefore, in Lemma 3, we determine an upper bound on the remaining work at t_b. In order to do so, we will compute an upper bound on the remaining work at t_b for each task in the system. Lemma 3 is the core of the argument behind Theorem 1, so we prove it in full here.

Lemma 3. *If $\vec{x}$ is compliant and all jobs of τ_j with higher priority than J_i complete with response time no greater than $Y_j + x_j + C_j$, then $W(t_b) \leq G(\vec{x}, \vec{Y}) + S(\vec{Y}) + U^+(y_i - t_b)$.*

Proof. We consider each task τ_j individually. We account for the remaining work for τ_j at time t_b, considering all possible cases, described next.

Case 1 (Figure 4(a)). It may be the case that τ_j is executing immediately before t_b, but either is not executing at some time in $[t_d, t_b)$, or is not executing before t_b because $t_b = 0$. In this case, denote as t_s the earliest time such that τ_j executes continuously in $[t_s, t_b)$. τ_j must not have any unfinished work just before t_s, because at least one processor is available then. Therefore, any work that contributes to $W_j(t_s)$ must consist only of jobs that are released at or after t_s and that have PPs at or before y_i. Therefore,

$$\begin{aligned} W_j(t_s) &\leq \text{DBF}(\tau_j, Y_j, y_i - t_s) \\ &\leq \{\text{By Lemma 2}\} \\ &\quad U_j(y_i - t_s) + S_j(Y_j). \end{aligned} \tag{15}$$

Furthermore, because τ_j executes continuously in $[t_s, t_b)$,

$$\begin{aligned} W_j(t_b) &= W_j(t_s) - (t_b - t_s) \\ &\leq \{\text{By (15)}\} \\ &\quad U_j(y_i - t_s) + S_j(Y_j) - (t_b - t_s) \\ &\leq \{\text{Because } U_j \leq 1)\} \\ &\quad U_j(y_i - t_s) + S_j(Y_j) - U_j(t_b - t_s) \\ &= \{\text{Rearranging}\} \\ &\quad U_j(y_i - t_b) + S_j(Y_j). \end{aligned} \tag{16}$$

Case 2 (Figure 4(b)). It may be the case that τ_j is executing continuously in $[t_d, t_b)$, and its job executing at t_d has a PP at or after t_d. In this case, we call that job J_j. The release time of J_j must be no earlier than $t_d - Y_j$, or Case 3 below would instead apply. Thus, the work for τ_j remaining at t_d must consist only of jobs that have releases at or after $t_d - Y_j$ and PPs at or before y_i. Therefore,

$$W_j(t_d) \leq \text{DBF}(\tau_j, Y_j, y_i - (t_d - Y_j))$$

$$
\begin{aligned}
&\leq \{\text{By Lemma 1; note that } y_i - (t_d - Y_j) = Y_j + (y_i - t_d) \geq Y_i\} \\
&\quad U_j(y_i - (t_d - Y_j)) + C_j \left(1 - \frac{Y_j}{T_j}\right) \\
&= \{\text{Rearranging and using } U_j = C_j/T_j\} \\
&\quad U_j(y_i - t_d) + \frac{C_j Y_j}{T_j} + C_j - \frac{C_j Y_j}{T_j} \\
&= \{\text{Cancelling}\} \\
&\quad U_j(y_j - t_d) + C_j. \qquad (17)
\end{aligned}
$$

Because τ_j runs continuously in $[t_d, t_b)$,

$$
\begin{aligned}
W_j(t_b) &= W_j(t_d) - (t_b - t_d) \\
&\leq \{\text{By (17)}\} \\
&\quad U_j(y_i - t_d) + C_j - (t_b - t_d) \\
&\leq \{\text{Because } U_j \leq 1)\} \\
&\quad U_j(y_i - t_d) + C_j - U_j(t_b - t_d) \\
&= \{\text{Rearranging}\} \\
&\quad U_j(y_i - t_b) + C_j. \qquad (18)
\end{aligned}
$$

Case 3 (Figure 4(c)). It may be the case that τ_j is executing continuously in $[t_d, t_b)$, and its job executing at t_d has a PP earlier than t_d. We call that job J_j. We refer to the PP of J_j as y_j. Because $y_j < t_d \leq t_b \leq y_i$, J_j must have priority higher than J_i, i.e., J_j is not J_i. Therefore, by the precondition of the lemma, J_j's response time must be no greater than

$$
Y_j + x_j + C_j. \qquad (19)
$$

We define δ such that the remaining execution of J_j at t_d is $C_j - \delta$. If J_j runs for its full worst-case execution time, then δ is simply its execution before t_d, but δ may be larger if J_j completes early. J_j must finish no earlier than $t_d + C_j - \delta$. By the definition of response time from Section 2 and (19), τ_j must be released no earlier than $t_d + C_j - \delta - (Y_j + x_j + C_j)$. Thus, the work for τ_j remaining at t_d must consist only of jobs that have releases at or after $t_d + C_j - \delta - (Y_j + x_j + C_j)$ and PPs at or before y_i. Below we use Lemma 1 to bound the work from these jobs, so we need to establish that the length of this interval is at least Y_j. We first observe that, resulting from the definition of t_b

and the fact that τ_j is running just before t_b, $U^+ > 1$. Therefore, by the precondition of the lemma and Definition 1,

$$x_j \geq 0. \tag{20}$$

Thus,

$$\begin{aligned} y_i - (t_d + C_j - \delta - (Y_j + x_j + C_j)) &= \{\text{Rearranging}\} \\ &\quad (y_i - t_d) + x_j + \delta + Y_j \\ &\geq \{\text{Because } y_i \geq t_d,\ \delta \geq 0, \text{ and by (20)}\} \\ &\quad Y_j. \end{aligned} \tag{21}$$

Recall that only $C_j - \delta$ units of work remain for J_j at time t_d, but $\text{DBF}(\tau_j, Y_j, \ell)$ assumes a demand of C_j for every job within the considered interval. Therefore,

$$\begin{aligned} W_j(t_d) &\leq \text{DBF}(\tau_j, Y_j, y_i - (t_d + C_j - \delta - (Y_j + x_j + C_j))) - \delta \\ &\leq \{\text{By Lemma 1, and by (21) }\} \\ &\quad U_j(y_i - (t_d + C_j - \delta - (Y_j + x_j + C_j))) + C_j \left(1 - \frac{Y_j}{T_j}\right) - \delta \\ &\leq \{\text{Rearranging, using } U_j = C_j/T_j, \text{ and because } U_j \leq 1)\} \\ &\quad U_j(y_i - t_d) - U_j C_j + U_j \delta + \frac{C_j Y_j}{T_j} + U_j x_j + U_j C_j + C_j - \frac{C_j Y_j}{T_j} - U_j \delta \\ &= \{\text{Cancelling}\} \\ &\quad U_j(y_i - t_d) + U_j x_j + C_j \end{aligned} \tag{22}$$

Because τ_j runs continuously in $[t_d, t_b)$,

$$\begin{aligned} W_j(t_b) &= W_j(t_d) - (t_b - t_d) \\ &\leq \{\text{By (22)}\} \\ &\quad U_j(y_i - t_d) + U_j x_j + C_j - (t_b - t_d) \\ &\leq \{\text{Because } U_j \leq 1)\} \\ &\quad U_j(y_i - t_d) + U_j x_j + C_j - U_j(t_b - t_d) \end{aligned}$$

$$= \{\text{Rearranging}\}$$

$$U_j(y_i - t_b) + U_j x_j + C_j \tag{23}$$

Total Remaining Work at t_b. We consider two possible cases, depending on the value of U^+.

If $U^+ = 1$, then by the definition of t_b, all tasks must be in Case 1. In addition, by the definition of $G(\vec{x}, \vec{Y})$ in (13),

$$G(\vec{x}, \vec{Y}) = 0. \tag{24}$$

Therefore, the total work is at most

$$\begin{aligned}
\sum_{\tau_j \in \tau} (U_j(y_i - t_b) + S_j(Y_j)) &= \{\text{Rearranging}\} \\
&\left(\sum_{\tau_j \in \tau} U_j \right) (y_i - t_b) + \sum_{\tau_j \in \tau} S_j(Y_j) \\
&\leq \{\text{By the definition of } U^+ \text{ in (5), and the definition} \\
&\quad \text{of } S(\vec{Y}) \text{ in (10), and (24)}\} \\
&\quad G(\vec{x}, \vec{Y}) + S(\vec{Y}) + U^+(y_i - t_b),
\end{aligned}$$

and the lemma holds.

On the other hand, if $U^+ > 1$, then by Definition 1,

$$\forall j, x_j \geq 0. \tag{25}$$

Because we assume that $\forall i, Y_i \geq 0$, and by the definition of $S_i(Y_i)$ in (9),

$$\forall j, S_j(Y_j) \leq C_j. \tag{26}$$

By (25) and (26), the worst case for each τ_j is Case 3. Any task can be in Case 1, but due to the definition of t_b, at most $U^+ - 1$ tasks can be in Case 2 or 3 due to the definition of t_b. Therefore, the total demand due to all tasks can be upper bounded by selecting a set M of $U^+ - 1$ tasks in τ such that M maximizes

$$\sum_{\tau_j \in M} (U_j(y_i - t_b) + U_j x_j + C_j) + \sum_{\tau_j \in (\tau \setminus M)} (U_j(y_i - t_b) + S_j(Y_j))$$

$$
\begin{aligned}
&= \{\text{Rearranging}\} \\
&\quad \left(\sum_{\tau_j \in \tau} U_j\right)(y_i - t_b) + \sum_{\tau_j \in M} (x_j U_j + C_j) + \sum_{\tau_j \in (\tau \setminus M)} S_j(Y_j) \\
&= \{\text{Adding } \textstyle\sum_{\tau_j \in M} S_j(Y_j) - \sum_{\tau_j \in M} S_j(Y_j) = 0\} \\
&\quad \left(\sum_{\tau_j \in \tau} U_j\right)(y_i - t_b) + \sum_{\tau_j \in M} (x_j U_j + C_j - S_j(Y_j)) + \sum_{\tau_j \in \tau} S_j(Y_j) \\
&\leq \{\text{By the definition of } U^+ \text{ in (5), the definition of } G(\vec{x}, \vec{Y}) \text{ in (13),} \\
&\quad \text{the definition of } S(\vec{Y}) \text{ in (10), and the definition of } M \text{ above}\} \\
&\quad U^+(y_i - t_b) + G(\vec{x}, \vec{Y}) + S(\vec{Y}).
\end{aligned}
$$

Therefore, the lemma holds. □

The next lemma simply bounds $W(y_i)$, given the bound in Lemma 3.

Lemma 4. *If $\vec{x}$ is compliant and all jobs of τ_j with higher priority than J_i complete with response time no greater than $Y_j + x_j + C_j$, then $W(y_i) \leq G(\vec{x}, \vec{Y}) + S(\vec{Y})$.*

Proof. By the definition of t_b, at least U^+ processors are completing work from jobs in H throughout the interval $[t_b, y_i)$. Therefore,

$$
\begin{aligned}
W(y_i) &\leq W(t_b) - U^+(y_i - t_b) \\
&\leq \{\text{By Lemma 3}\} \\
&\quad G(\vec{x}, \vec{Y}) + S(\vec{Y}).
\end{aligned}
$$

□

Our final lemma, proved in the appendix, bounds the response time of J_i.

Lemma 5. *If $\vec{x}$ is compliant and each job of τ_j with higher priority than J_i finishes with response time no greater than $Y_j + x_j + C_j$, then J_i finishes with response time no greater than $Y_i + x_i + C_i$.*

With these lemmas in place, we can now state and prove the primary result of this section.

Theorem 1. *If $\vec{x}$ is compliant, then no job of any task τ_i will have a response time exceeding $Y_i + x_i + C_i$.*

Proof. This result follows immediately from Lemma 5 by induction over all jobs in the system, considered in order of decreasing priority. As the base case, the precondition for Lemma 5 holds vacuously when considering the first job. □

4 Minimum Compliant Vector

In this section, we will demonstrate that there is a unique *minimum* compliant vector for a given selection of PPs. We begin by narrowing our consideration of compliant vectors based on the idea that if we can reduce the bound for one task without increasing any other bound in the system, we clearly do not have the best compliant vector. Definition 2 below defines a notion similar to the minimal compliant vector described by Erickson et al. (2010a). (We prove in Lemma 7 below that the minimum near-compliant vector is unique.)

Definition 2. *A* minimum near-compliant vector *is a near-compliant vector such that, if any component x_i is reduced by any $\delta > 0$, the vector ceases to be near-compliant.*

Like Erickson et al. (2010b), we next precisely characterize any minimum near-compliant vector. Lemma 6 is proved in the appendix.

Lemma 6. *$\vec{x}$ is a minimum near-compliant vector iff*

$$\forall i, x_i = \frac{G(\vec{x}, \vec{Y}) + S(\vec{Y}) - C_i}{m}. \tag{27}$$

Lemma 6, when combined with the fact that a minimum near-compliant vector is in fact compliant (proved as Corollary 2 below), allows us to constrain the form of the vectors considered. We state this observation as Corollary 1 below, which follows immediately from Lemma 6.

Corollary 1. *$\vec{x}$ is a minimum near-compliant vector iff there exists s such that*

$$\forall i, x_i = \frac{s - C_i}{m} \tag{28}$$

and

$$s = G(\vec{x}, \vec{Y}) + S(\vec{Y}). \tag{29}$$

Observe that s is independent of the task index i.

The following lemma, proved in the appendix, shows that we can unambiguously refer to *the* minimum near-compliant vector for the system.

Lemma 7. *If the minimum near-compliant vector for the task system exists, it is unique.*

Later in this section, we prove that the minimum near-compliant vector does indeed exist. We do so by formulating a feasible LP such that the optimal solution must contain the minimum near-compliant vector. In addition to showing that the minimum near-compliant vector exists, this LP is also useful for actually computing the minimum near-compliant vector.

In order to prove that the minimum near-compliant vector exists, we will show that the minimum near-compliant vector is the optimal solution to a more general LP based on Lemma 8 below.

Lemma 8. *$\vec{x}$ is a near-compliant vector if* (28) *holds and*

$$s \geq G(\vec{x}, \vec{Y}) + S(\vec{Y}). \tag{30}$$

Proof. Follows immediately from Definition 1. □

Recall that a compliant vector is described by (9)–(10) and (13)–(14). We now show that Lemma 8 and the equations it depends on can be reformulated as constraints for the LP mentioned above, similar to the LP shown by Ward et al. (2013). Throughout this section, whenever we use a subscript i (e.g., Y_i), there is one variable or constant for each task (e.g., a Y_i for each task). We assume that x_i, S_i, S_{sum}, G, and s are variables, and we also introduce the auxiliary variables b and z_i. All other values are constants (i.e., Y_i, U_i, C_i, T_i, D_i, U^+, and m), but Y_i will be used as a variable later in Sections 5 and 6.

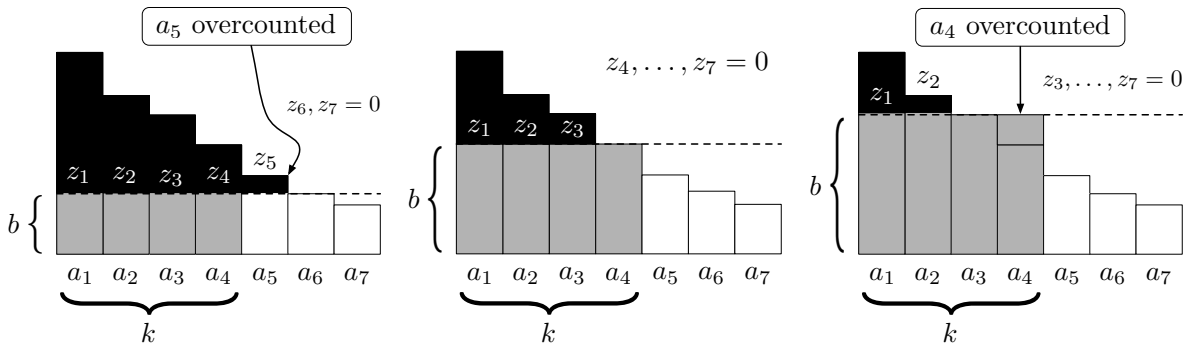
Constraint Set 1. *The linear constraints corresponding to the definition of $\vec{x}$ in* (28) *used in Lemma 8 are given by*

$$\forall i : x_i = \frac{s - C_i}{m}.$$

Constraint Set 2. *The linear constraints corresponding to the definition of $S_i(Y_i)$ in* (9) *are given by*

$$\forall i : S_i \geq 0; \quad S_i \geq C_i(1 - Y_i/T_i).$$

The two constraints in Constraint Set 2 model the two terms of the max in the definition of $S_i(Y_i)$ in (9). If $C_i(1 - Y_i/T_i) \leq 0$, ensuring that $S_i \geq S_i(Y_i)$ holds for each i. Because S_i is not bounded from above, Constraint Set 2



(a) b too small. (b) b correct. (c) b too large.

Figure 5: Illustration of the auxiliary variables used to sum the largest k elements in the set $A = \{a_1, \ldots, a_n\}$. The total of the gray and black shaded areas is equal to G. The gray areas correspond to kb while the black areas correspond to positive z_i's. When G is minimized, as in (b), G is equal to the sum of the largest k elements in A. As is shown in (a) and (c), if b is too small or too large then G will be larger than the maximum k elements in A. Note that elements of A are depicted in sorted order only for visual clarity.

only ensures that

$$\forall i, S_i \geq S_i(Y_i). \tag{31}$$

However, we will show in Lemma 10 below that this will still result in a near-compliant vector.

The next equation to be linearized, the definition of $G(\vec{x}, \vec{Y})$ in (13), is less straightforward. We first show how to minimize the sum of the largest k elements in a set $A = \{a_1, \ldots, a_n\}$ using only linear constraints, by an approach similar to one proposed by Ogryczak and Tamir (2003). The intuition behind this approach is shown in Figure 5. This figure corresponds to the following LP

$$\begin{array}{lllll} \text{Minimize :} & G & & & \\ \text{Subject to :} & G & = & kb + \sum_{i=1}^{n} z_i, & \\ & z_i & \geq & 0 & \forall i, \\ & z_i & \geq & a_i - b & \forall i, \end{array}$$

in which G, b, and z_i are variables and both k and a_i are constants. In Figure 5, the term kb corresponds to the gray-shaded area, and $\sum_{i=1}^{n} z_i$ corresponds to the black-shaded area. When G is minimized, it is equal to the sum of the largest k elements in A. This is achieved when $z_i = 0$ for each element a_i that is not one of the k largest elements in A, and b is at most the k^{th} largest element in A, as is shown in Figure 5 (b).

Using this technique, we can formulate the definition of $G(\vec{x}, \vec{Y})$ in (13), as a set of linear constraints.

Constraint Set 3. *The linear constraints corresponding to the definition of $G(\vec{x}, \vec{Y})$ in (13) are given by*

$$\begin{aligned} &G = b(U^+ - 1) + \sum_{\tau_i \in \tau} z_i, \\ &\forall i : z_i \geq 0, \\ &\forall i : z_i \geq x_i U_i + C_i - S_i - b. \end{aligned}$$

In some of the optimization objectives we consider in Sec. 6, G is not itself explicitly minimized, as in the example LP above. Furthermore, this constraint uses S_i in place of $S_i(Y_i)$. As a result, this set of constraints only ensures that

$$G \geq \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i). \tag{32}$$

As shown in Lemma 10 below, this is sufficient to ensure a near-compliant vector.

Constraint Set 4. *The linear constraint corresponding to the definition of $S(\vec{Y})$ in (10), is given by*

$$S_{\text{sum}} = \sum_{\tau_i \in \tau} S_i.$$

Constraint Set 5. *The linear constraint corresponding to (30) in Lemma 8 is given by*

$$s \geq G + S_{\text{sum}}.$$

As noted above, Constraint Sets 2–3 ensure only that (31) and (32) are inequalities rather than equalities. The next two lemmas, proved in the appendix, demonstrate that Constraint Sets 1–5 are still sufficient to ensure that $\vec{x}$ is a near-compliant vector.

Lemma 9. *For any assignment of variables satisfying Constraint Sets 1–4,*

$$G + S_{\text{sum}} \geq G(\vec{x}, \vec{Y}) + S(\vec{Y}). \tag{33}$$

Lemma 10. *For any assignment of variables satisfying Constraint Sets 1–5, $\vec{x}$ is a near-compliant vector.*

We now show that there is a single best minimum near-compliant vector, which is in fact a compliant vector, for the system by minimizing the objective function s under Constraint Sets 1–5. We show that an optimal value of s exists, and that the corresponding $\vec{x}$ under Constraint Set 1 is both the minimum near-compliant vector and in fact compliant.

We will first demonstrate that an optimal $\vec{x}$ exists. Observe from Constraint Set 1 that $\vec{x}$ is uniquely determined by the assignment of s. The next two lemmas, proved in the appendix, show that an optimal value of s exists. Lemma 11 demonstrates that there is a lower bound on s for any feasible solution, and Lemma 12 demonstrates that a feasible solution exists.

Lemma 11. *If $s < 0$, then Constraint Sets 1–5 are infeasible.*

Lemma 12. *Constraint Sets 1–5 are feasible.*

By Lemmas 11–12 and the fact that Constraint Sets 1–5 are linear, an optimal minimum value of s must exist. The next lemma shows that the corresponding $\vec{x}$ must be the minimum near-compliant vector.

Lemma 13. *For any assignment of variables that satisfies Constraint Sets 1–5 such that s obtains its minimum value, $\vec{x}$ is the minimum near-compliant vector.*

Proof. We use proof by contradiction. Assume an assignment of variables that satisfies Constraint Sets 1–5 such that $\vec{x}$ is not a minimum near-compliant vector and s obtains its minimum value.

We show an assignment of variables, denoted with a prime (e.g. s' for the new assignment of s), that is also feasible, but with $s' < s$.

Let

$$s' = G(\vec{x}, \vec{Y}) + S(\vec{Y}), \tag{34}$$

$$\forall i, x'_i = \frac{s' - C_i}{m}, \tag{35}$$

$$\forall i, S'_i = S_i(Y_i), \tag{36}$$

$$G' = G(\vec{x}', \vec{Y}), \tag{37}$$

$$b' = (U^+ - 1)^{th} \text{ largest value of } x'_i U_i + C_i - S_i, \tag{38}$$

$$\forall i, z'_i = \max\{0, x'_i U_i + C_i - S_i - b'\}, \tag{39}$$

$$S'_{\text{sum}} = S(\vec{Y}). \tag{40}$$

We show

$$\begin{aligned} s > \{&\text{By Definition 1, Constraint Set 1, and Lemma 10,} \\ &\text{because } \vec{x} \text{ is not minimum}\} \\ &G(\vec{x}, \vec{Y}) + S(\vec{Y}) \end{aligned}$$

$$
\begin{aligned}
&= \{\text{By (34)}\} \\
&\quad s'. \qquad (41)
\end{aligned}
$$

We now show that the new assignment of variables satisfies Constraint Sets 1–5.

Constraint Set 1 holds by (35).

Constraint Set 2 holds by (36).

Constraint Set 3 holds by (37)–(39).

Constraint Set 4 holds by (36) and (40).

To see that Constraint Set 5 holds, note that

$$
\begin{aligned}
s' &= \{\text{By (34)}\} \\
&\quad G(\vec{x},\vec{Y}) + S(\vec{Y}) \\
&= \{\text{By the definition of } G(\vec{x},\vec{Y}) \text{ in (13)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + S(\vec{Y}) \\
&= \{\text{By Constraint Set 1}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{s - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + S(\vec{Y}) \\
&> \{\text{By (41)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{s' - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + S(\vec{Y}) \\
&= \{\text{By (34)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} (x'_i U_i + C_i - S_i(Y_i)) + S(\vec{Y}) \\
&= \{\text{By the definition of } G(\vec{x},\vec{Y}) \text{ in (13)}\} \\
&\quad G(\vec{x}',\vec{Y}) + S(\vec{Y}) \\
&= \{\text{By (37) and (40)}\} \\
&\quad G' + S'_{\text{sum}}.
\end{aligned}
$$

Because the new assignment satisfies Constraint Sets 1–5 with $s' < s$ (by (41)), the original assignment did not achieve the minimum value of s. □

The previous lemmas are sufficient to demonstrate that the minimum near-compliant vector for a system must exist. Our remaining proof obligation is to show that the minimum near-compliant vector is in fact compliant. Lemma 14, proved in the appendix, demonstrates that for the optimal assignment of variables, s must be at least the maximum C_i in the system. We denote this maximum C_i as $C_{\max}$.

Lemma 14. *If $U^+ > 1$ and $s < C_{\max}$, then Constraint Sets 1–5 are infeasible.*

Lemma 14 is used in Lemma 15 below, which demonstrates that for any assignment of variables satisfying Constraint Sets 1–5, $\vec{x}$ is compliant. By Lemma 15, the minimum near-compliant vector must also be compliant.

Lemma 15. *For any assignment of variables satisfying Constraint Sets 1–5, $\vec{x}$ is compliant.*

Proof. If $U^+ = 1$, then the lemma follows trivially from Definition 1. We therefore assume $U^+ > 1$.

Because the assignment of variables satisfies Constraint Sets 1–5, by Lemma 14,

$$s \geq C_{\max}. \tag{42}$$

For arbitrary i,

$$\begin{aligned} x_i &= \{\text{By Constraint Set 1}\} \\ &\quad \frac{s - C_i}{m} \\ &\geq \{\text{By (42)}\} \\ &\quad \frac{C_{\max} - C_i}{m} \\ &\geq \{\text{Because } \forall i, C_{\max} \geq C_i\} \\ &\quad 0. \end{aligned}$$

Therefore, by Definition 1, $\vec{x}$ is compliant. □

Corollary 2. *The minimum near-compliant vector for any task system is compliant.*

5 Global Fair Lateness Scheduling

Having shown that a minimum compliant vector exists for any combination of feasible task system and scheduler, we now turn our attention to determining the scheduler that minimizes the maximum lateness bound for a system.

As described later in Section 6, the LP described by Constraint Sets 1–5 can be used to select optimal relative PPs for an arbitrary linear lateness constraint. However, determining the relative PPs requires the use of an LP solver. In this section, we present a particular scheduler that has a simple expression for its relative PPs and that is optimal with respect to minimizing the maximum lateness bound for a task system.

In order to provide the best analysis for a given scheduler, we observe that some GEL schedulers are identical with respect to the scheduling decisions made at runtime, even though the CVA bounds may not be identical. We formally define and motivate this notion below in Definition 3 and Lemma 16.

Definition 3. *Two PP assignments $\vec{Y}$ and $\vec{Y}'$ are* equivalent *if there exists a constant c such that $Y_i = Y_i' + c$ for any i. Two GEL schedulers are* equivalent *if their respective PP assignments are equivalent.*

Lemma 16. *If two GEL schedulers are equivalent with PP assignments $\vec{Y}$ and $\vec{Y}'$, respectively, then the response time bounds derived by using $\vec{Y}$ will also apply to a system scheduled using $\vec{Y}'$, and vice versa.*

Proof. Using either $\vec{Y}$ or $\vec{Y}'$ will result in the same scheduler decisions, because each absolute PP has been increased or decreased by the same constant. □

We now define in Definition 4 a scheduler that, although it may not itself have the lowest maximum CVA bound, is equivalent to a scheduler that does. The value of $\vec{Y}$ in Definition 4 is in concise form, and is provably equivalent to a GEL scheduler with the lowest possible maximum CVA bound. A system designer can use the definition of $\vec{Y}$ in Definition 4 when setting scheduler parameters, and the lowest available maximum CVA bound will apply to the resulting system.

Definition 4. *The **Global Fair Lateness (G-FL)** scheduler is the GEL scheduler with relative PP assignment*

$$\forall i, Y_i = D_i - \frac{m-1}{m} C_i.$$

We first show as Theorem 2 below that the PPs for an arbitrary GEL scheduler can be modified to ensure that all lateness bounds are the same, without increasing the maximum lateness bound for the scheduler, and the resulting scheduler is equivalent to G-FL. We then discus how to obtain the best lateness bounds for any scheduler equivalent to G-FL.

Theorem 2. *Let V be an arbitrary assignment of variables satisfying Constraint Sets 1–5. There exists an assignment V' (with each variable denoted with a prime) such that V' also satisfies Constraint Sets 1–5, the scheduler using $\vec{Y}'$ is equivalent to G-FL, and the maximum lateness bound using $\vec{x}'$ in Theorem 1 is no greater than using $\vec{x}$.*

Proof. By Lemma 15, $\vec{x}$ is compliant. Therefore, by Theorem 1, the maximum lateness bound for the system is

$$\begin{aligned}
&\max\left(Y_j + x_j + C_j - D_j\right) \\
&= \{\text{By Constraint Set 1}\} \\
&\max\left(Y_j + \frac{s - C_j}{m} + C_j - D_j\right) \\
&= \{\text{Rearranging; observe that } s \text{ does not depend on task index } j\} \\
&\frac{s}{m} + \max\left(Y_j + \frac{m-1}{m}C_j - D_j\right) \qquad (43)
\end{aligned}$$

We present the following assignment of variables for V':

$$Y'_i = \max\left(Y_j + \frac{m-1}{m}C_j - D_j\right) - \frac{m-1}{m}C_i + D_i \qquad (44)$$

$$s' = s, \qquad (45)$$

$$\forall i, x'_i = \frac{s' - C_i}{m}, \qquad (46)$$

$$\forall i, S'_i = S_i(Y'_i), \qquad (47)$$

$$G' = G(\vec{x}', \vec{Y}'), \qquad (48)$$

$$b' = (U^+ - 1)^{th} \text{ largest value of } x'_i U_i + C_i - S'_i, \qquad (49)$$

$$\forall i, z_i = \max\{0, x'_i U_i + C_i - S'_i - b'\}, \qquad (50)$$

$$S_{\text{sum}} = S(\vec{Y}'). \qquad (51)$$

Consider an arbitrary task τ_i. By rearranging the following expression (which clearly holds),

$$Y_i + \frac{m-1}{m}C_i - D_i \le \max_{\tau_j \in \tau}\left(Y_j + \frac{m-1}{m}C_j - D_j\right),$$

we have

$$Y_i \le \max_{\tau_j \in \tau}\left(Y_j + \frac{m-1}{m}C_j - D_j\right) - \frac{m-1}{m}C_i + D_i,$$

so by (44),

$$Y_i \le Y'_i. \qquad (52)$$

Therefore, by the definition of $S_i(Y_i)$ in (9),

$$S_i(Y'_i) \le S_i(Y_i). \qquad (53)$$

Also, by (45)–(46),

$$x_i = x_i'. \tag{54}$$

Constraint Set 1 holds for V' by (46).

Constraint Set 2 holds for V' by (47).

Constraint Set 3 holds for V' by (48)–(50).

Constraint Set 4 holds for V' by (47) and (51).

To see that Constraint Set 5 holds for V', note that

$$\begin{aligned}
s' &= \{\text{By (45)}\} \\
&\quad s \\
&\leq \{\text{By Constraint Set 5}\} \\
&\quad G + S_{\text{sum}} \\
&\geq \{\text{By Lemma 9}\} \\
&\quad G(\vec{x}, \vec{Y}) + S(\vec{Y}) \\
&= \{\text{By the definition of } G(\vec{x}, \vec{Y}) \text{ in (13), and the definition of } S(\vec{Y}) \text{ in (10)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + \sum_{\tau_i \in \tau} S_i(Y_i) \\
&\geq \{\text{By (53); observe that each } S_i(Y_i) \text{ in the first summation also appears in} \\
&\quad \text{the second.}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \left(x_i U_i + C_i - S_i(Y_i')\right) + \sum_{\tau_i \in \tau} S_i(Y_i) \\
&= \{\text{By the definition of } S(\vec{Y}) \text{ in (10), and the definition of } G(\vec{x}, \vec{Y}) \text{ in (13),} \\
&\quad \text{and by (54)}\} \\
&\quad G(\vec{x}', \vec{Y}') + S(\vec{Y}') \\
&= \{\text{By (48) and (51)}\} \\
&\quad G' + S'_{\text{sum}}.
\end{aligned}$$

For an arbitrary τ_i, the lateness bound under Theorem 1 corresponding to the V' becomes

$$x_i' + Y_i' + C_i - D_i = \{\text{By (44) and (54)}\}$$

$$
\begin{aligned}
& x_i + \max_{\tau_j \in \tau} \left(Y_j + \frac{m-1}{m} C_j - D_j \right) + \frac{1}{m} C_i. \\
&= \{\text{By Constraint Set 1}\} \\
& \frac{s - C_i}{m} + \max_{\tau_j \in \tau} \left(Y_j + \frac{m-1}{m} C_j - D_j \right) + \frac{1}{m} C_i \\
&= \{\text{Cancelling}\} \\
& \frac{s}{m} + \max_{\tau_j \in \tau} \left(Y_j + \frac{m-1}{m} C_j - D_j \right). \qquad (55)
\end{aligned}
$$

By (43) and (55), the maximum lateness bound under V is identical to the maximum lateness bound under V'. Furthermore, by Definition 4 and (44), the GEL scheduler using $\vec{Y}'$ is equivalent to G-FL. □

Theorem 2 demonstrates that G-FL is equivalent to a GEL scheduler with a lateness bound under Theorem 1 no greater than that of any other GEL scheduler. We would like to determine the "best" equivalent scheduler to G-FL so that we can obtain the smallest bounds applicable to G-FL. Towards that end, in the next lemma, we show that lateness bounds from equivalent schedulers can be compared in a straightforward manner.

Lemma 17. *Suppose two PP assignments $\vec{Y}$ and $\vec{Y}'$ are equivalent and denote their corresponding lateness bounds, when using minimum near-compliant vectors $\vec{x}$ and $\vec{x}'$, as $\vec{L}$ and $\vec{L}'$. There is a constant k such that, for all i, $L_i = L'_i + k$. Each L_i and L'_i differ by a system-wide constant.*

Proof.

$$
\begin{aligned}
L_i - L'_i &= \{\text{By Theorem 1 and Corollary 2}\} \\
& x_i + Y_i + C_i - D_i - (x'_i + Y'_i + C_i - D_i) \\
&= \{\text{By Corollary 1, for some } s \text{ and } s'\} \\
& \frac{s - C_i}{m} + Y_i + C_i - D_i - \left(\frac{s' - C_i}{m} + Y'_i \right) \\
&= \{\text{Cancelling}\} \\
& \frac{s}{m} + Y_i - (\frac{s'}{m} + Y'_i) \\
&= \{\text{Rearranging}\} \\
& \frac{s - s'}{m} + (Y_i - Y'_i) = \qquad \{\text{By Definition 3}\} \\
& \frac{s - s'}{m} + c. \qquad (56)
\end{aligned}
$$

Because s, s', and c do not depend on task index i, we let $k = \frac{s-s'}{m} + c$, and the lemma follows. □

We now describe additional constraint sets that can be used with Constraint Sets 1–5 in order to determine the lateness bounds under the equivalent scheduler to GEL with the smallest lateness bounds.

Constraint Set 6. *The constraints to ensure that $\vec{Y}$ meets our assumption in (8) are*

$$\forall i, Y_i \geq 0.$$

In order to determine the best GEL scheduler equivalent to G-FL, we add an auxillary variable c and the following constraint. Because c can be any arbitrary value, this constraint specifies that any $\vec{Y}$ equivalent to G-FL is acceptable.

Constraint Set 7. *The constraints allowing any $\vec{Y}$ equivalent to G-FL are*

$$\forall i, Y_i = c + D_i - \frac{m-1}{m} C_i.$$

By minimizing s under Constraint Sets 1–7, we can obtain the desired lateness bounds, and therefore the best maximum lateness bound under any GEL scheduler.

6 Alternate Optimization Criteria

G-FL was proven in Sec. 5 to be optimal relative to the specific criterion of minimizing maximum lateness bound under CVA. Under G-FL, the system implementer does not need to use an LP solver to define PPs but instead can assign PPs using Def. 4. For G-FL, the LP solver is only necessary in order to analyze the lateness bounds of the system. In this section, we show how to use linear programming in order to achieve alternative lateness criteria. For example, we show how to minimize *average* lateness, or to minimize maximum proportional lateness. In order to achieve these criteria, it is necessary to use a set of PPs that *differ* from G-FL, and the system implementer must use the LP solver to determine the PPs.

Next we show how Constraint Sets 1–6 can be coupled with objective functions, and possibly additional constraint sets, to find optimal priority point settings under CVA with respect to alternative criteria. We define several different schedulers based on their lateness criteria. In all cases, our criterion is to minimize some lateness metric, such as maximum lateness. We will denote each criterion with two letters indicating the type of lateness to be minimized. The first is "*A*" for *average* or "*M*" for *maximum*, and the second is "*L*" for *lateness* or "*P*" for *proportional lateness*. No definition is provided for ML (maximum lateness), because G-FL optimizes the same criterion, as discussed above.

Where two criteria are provided, the first is optimized, then the second. For example, ML-AL below minimizes the average lateness *subject to having the smallest maximum lateness possible*.

Minimizing Maximum Proportional Lateness: MP. As described in Section 5, G-FL has the smallest maximum lateness bound for any GEL scheduler under CVA. However, for some applications, more tardiness may be acceptable for tasks that have longer relative deadlines. Therefore, it may be desirable to minimize the maximum *proportional* lateness, rather than the maximum lateness.

In order to minimize maximum proportional lateness, we define an auxiliary variable $I_{\max}$ that corresponds to the maximum proportional lateness for the task system. We add a set of constraints to ensure the appropriate value for $I_{\max}$ and then minimize it.

$$\begin{aligned} \text{Minimize :} \quad & I_{\max} \\ \text{Subject to :} \quad & \forall i : (Y_i + x_i + C_i - D_i)/(D_i) \leq I_{\max} \\ & \text{Constraint Sets 1–6} \end{aligned}$$

Minimizing Average Lateness: AL. G-FL guarantees the smallest maximum lateness bound available under CVA. However, depending on the nature of the application, it may be more desirable to obtain the smallest average lateness bound, rather than the maximum.

The following LP may be solved to minimize average lateness under CVA.

$$\begin{aligned} \text{Minimize :} \quad & \sum_{\tau_i \in \tau} Y_i + x_i \\ \text{Subject to :} \quad & \text{Constraint Sets 1–6} \end{aligned}$$

Note that average lateness is given by $\sum_{\tau_i \in \tau}(Y_i + x_i + C_i - D_i)/n$, but C_i, D_i, and n are all constants that are not necessary to include in the optimization objective.

While AL is optimal with respect to average lateness, as is shown experimentally in Section 7, the lateness of some tasks may be larger than the maximum lateness bound of G-FL, which we denote $L_{\max}$. Next, we show how to optimize the average lateness of all tasks while maintaining a maximum lateness no greater than $L_{\max}$.

Minimizing Average Lateness from G-FL: ML-AL. Although G-FL provides the smallest maximum lateness bound available under CVA, it does so by giving all tasks the same lateness bound. It may be possible to reduce the lateness bounds for some tasks without altering the maximum lateness bound for the system. Therefore, if the requirement to run an offline LP solver to determine PPs is not problematic, further optimizing the lateness bounds can be desirable. Here we show how to minimize the average lateness bound for the system with respect to having the

same maximum lateness bound as G-FL.

The following LP may be solved to minimize the average lateness under CVA while maintaining the same maximum lateness bound as G-FL.

$$\begin{aligned} \text{Minimize :} \quad & \sum_{\tau_i \in \tau} Y_i + x_i \\ \text{Subject to :} \quad & \forall i : Y_i + x_i + C_i - D_i \leq L_{\max}, \text{[1]} \\ & \text{Constraint Sets 1–6} \end{aligned}$$

As before, the constants C_i, D_i and n are omitted from the objective function.

Minimizing Average Proportional Lateness: AP. For applications where performance is more sensitive to the average lateness rather than the maximum lateness, but where tasks with longer deadlines can permit more tardiness, it may be desirable to minimize average proportional lateness.

The LP for minimizing average lateness can be modified in a straightforward manner to minimize average proportional lateness.

$$\begin{aligned} \text{Minimize :} \quad & \sum_{\tau_i \in \tau} (x_i + Y_i)/D_i \\ \text{Subject to :} \quad & \text{Constraint Sets 1–6} \end{aligned}$$

As was the case with average lateness, unnecessary constant terms have been omitted from the objective function.

Minimizing Average Proportional Lateness from Smallest Maximum Proportional Lateness: MP-AP. Just as it is desirable to reduce average lateness when it is possible to do so without increasing maximum lateness, it is desirable to reduce average proportional lateness when it is possible to do so without increasing maximum proportional lateness.

As we did with average lateness constrained by the maximum lateness from G-FL, we can also minimize the average proportional lateness constrained by the maximum proportional lateness from MP.

$$\begin{aligned} \text{Minimize :} \quad & \sum_{\tau_i \in \tau} (x_i + Y_i)/D_i \\ \text{Subject to :} \quad & \forall i : (Y_i + x_i + C_i - D_i)/D_i \leq I_{\max}, \text{[2]} \\ & \text{Constraint Sets 1–6} \end{aligned}$$

Once again, unnecessary constant terms have been omitted from the objective function.

We note that the LP formulation of CVA can be used and extended to other optimization objectives, perhaps most notably, application-specific optimization objectives. For example, an LP solver can be used to assign PPs to ensure

[1] Application-specific per-task lateness tolerances could be used instead of $L_{\max}$.

[2] As before, application-specific per-task proportional lateness tolerances could be used instead of $I_{\max}$.

application-specific lateness tolerances are satisfied (if possible under CVA), or to maximize total system utility under some linear definitions of lateness-based utility.

7 Experiments

In this section, we present experiments that demonstrate how G-FL and the LP-based schedulers described in Sec. 6 can improve lateness bounds over existing scheduling algorithms. In these experiments, we evaluated the lateness bounds of randomly generated task systems. We generated random task sets using a similar experimental design as in previous studies (e.g., (Erickson and Anderson, 2012)). We generated implicit-deadline task sets in which per-task utilizations were distributed either uniformly or bimodally. For task sets with uniformly distributed utilizations, per-task utilizations were chosen to be *light*, *medium* or *heavy*, which correspond to utilizations uniformly distributed in the range $[0.001, 0.1]$, $[0.1, 0.4]$, or $[0.5, 0.9]$, respectively. For task systems with bimodally distributed utilizations, per-task utilizations were chosen 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 $5/9$. The periods of all tasks were generated using an integral uniform distribution between $[3\,\text{ms}, 33\,\text{ms}]$, $[10\,\text{ms}, 100\,\text{ms}]$ and $[50\,\text{ms}, 250\,\text{ms}]$ for tasks with *short*, *moderate*, and *long* periods, respectively. We considered a system with $m = 8$ processors, as clustered scheduling typically is preferable to global scheduling for larger processor counts (Brandenburg, 2011). For each per-task utilization and period distribution combination, 1,000 task sets were generated for each total system utilization value in $\{1.25, 1.50, \ldots, 8.0\}$. We did not consider task systems with utilizations of one or less, as they are schedulable on one processor.

For each generated task system, we evaluated the average and maximum per-task lateness bounds under Devi and Anderson's analysis of G-EDF (Devi and Anderson, 2008) (EDF-DA), CVA analysis of G-EDF scheduling (EDF-CVA) by selecting the best equivalent scheduler as for G-FL in Section 5, CVA analysis of G-EDF using an alternative optimization rule[3] from Erickson et al. (2010a) (EDF-CVA2), G-FL, and most of the LP-based schedulers discussed in Sec. 6 (ML-AL, AP, and MP-AP). We do not evaluate MP because MP-AP is preferable. We present representative results in Figures 6–9. In those figures, we show the mean average and maximum lateness bounds for each total system utilization value over all generated task systems. Note that the lateness-bound results are analytical, and that in an actual schedule observed latenesses may be smaller.

> **Observation 1.** For task systems will small utilizations, the equivalent scheduler optimization of EDF-CVA outperforms the optimization in EDF-CVA2 (Erickson et al., 2010a); however, the converse is true for large utilizations.

[3] When $Y_i = T_i$ for all i, $S_i(Y_i) = 0$, and rather than defining $G(\vec{x}, \vec{Y})$ as the largest sum of $U^+ - 1$ values of $x_i U_i + C_i$, we can instead define $G(\vec{x}, \vec{Y})$ as the largest sum of only $U^+ - 2$ values of $x_i U_i + C_i$ plus an additional C_i.

This can be seen in Figures 6–9. Although the optimization from Erickson et al. (2010a) in EDF-CVA2 performs very well for task systems with large utilizations, it is only applicable if $Y_i = T_i$, for all i. Therefore, it is likely to not be useful for systems with deadlines significantly different from minimum separation times. The techniques proposed here are applicable to any GEL scheduler and are fully compatible with arbitrary deadlines.

Observation 2. All of the GEL schedulers we considered with PPs different from G-EDF, had smaller average lateness bounds than G-EDF.

The GEL schedulers we considered with PPs different from G-EDF typically had smaller average proportional lateness bounds than G-FL.

All these scheduling algorithms optimize, with respect to CVA, either the average or maximum (proportional) lateness of all tasks by moving PPs. Therefore, these algorithms should have smaller average lateness bounds than G-EDF. This can be observed in in part (a) of Figures 6–9.

Observation 3. The maximum lateness bounds of ML-AL and G-FL are the same, but the average lateness bound of ML-AL is at worst the average lateness bound of G-FL.

Based on the constraints and the optimization objective of ML-AL, the average and maximum lateness bounds are provably no worse than G-FL. As is seen in Figures 6 and 8, the improvement in average lateness in the task systems seen in our experiments was usually only a few ms.

Observation 4. Average lateness bounds were lower under AL than under G-FL and ML-AL. This improvement in average lateness is made possible by allowing for increased maximum lateness.

Average proportional lateness bounds were lower under AP than under MP-AP. This improvement in average proportional lateness is made possible by allowing for increased maximum proportional lateness.

As a result of the LP objective function, AL is optimal with respect to average lateness under CVA. In Figures 6 and 8, we see that the average lateness bound of AL is always smaller than all other schedulers, often by 10-20 ms or more. However, the maximum lateness bounds of AL are larger than G-FL and ML-AL. In most observed cases, lateness bounds of AL were less than or commensurate with G-EDF lateness bounds as determined by either CVA or Devi and Anderson's analysis, though in some cases the maximum lateness was greater than G-EDF by 10-20 ms. From these results, AL may be practical in many applications.

Similarly, due to its optimization criteria, AP is optimal with respect to average proportional lateness under CVA, so its maximum is typically several deadlines shorter than other schedulers. In almost all cases, AP had a lower average

proportional lateness than all other schedulers[4].

We note that the lateness bounds of AL in comparison to G-FL and ML-AL, and the bounds of AP in comparison to MP-AP, demonstrate that the LP solver has considerable flexibility in choosing priority points to optimize for certain lateness criteria. If some tasks have larger lateness tolerances than others, the PPs of the more tolerant tasks can be increased to improve the lateness bounds of the less tolerant tasks. This gives system designers much more flexibility to optimize the scheduler for application-specific needs.

> **Observation 5.** Benefits to average or maximum proportional lateness come at a cost to average or maximum lateness, and vice versa.

If two tasks have the same lateness bound but different relative deadlines, the task with the smaller relative deadline will have larger proportional lateness. The reverse is true if the proportional lateness bounds are the same. Therefore, improving lateness can worsen proportional lateness or vice versa. This effect is particularly strong when deadlines of different tasks have a large variance. This can be seen in all figures.

8 Conclusion

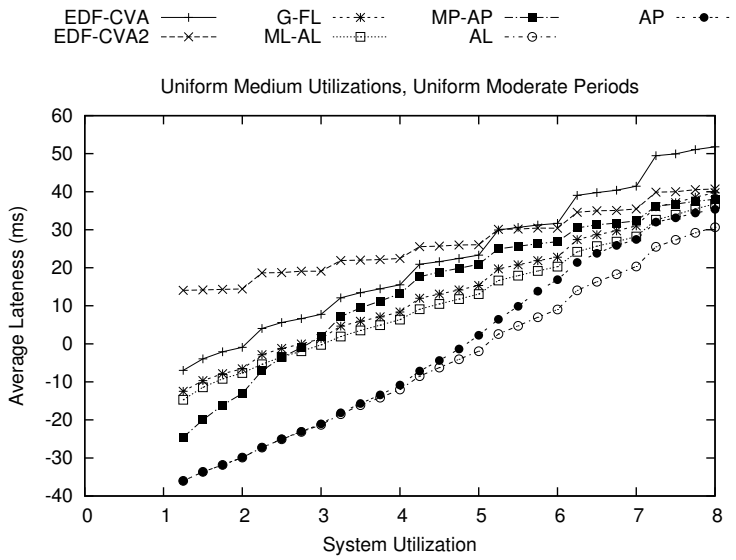
We have demonstrated that G-FL provides maximum absolute lateness bounds no larger than those currently available for G-EDF and have provided experimental evidence that G-FL is superior to G-EDF with regards to maximum lateness bounds. Furthermore, G-FL provides equal lateness bounds for all tasks in the system, and therefore provides a closer relationship between deadlines and response time bounds than G-EDF currently does. The implementation of G-FL is identical to that of G-EDF with arbitrary deadlines, and G-FL maintains the desirable JLSP property (enabling known synchronization techniques.) Therefore, G-FL is a better choice than G-EDF for SRT systems.

We have also demonstrated that LP techniques can be used to optimize for other linear constraints beyond minimizing maximum lateness bounds. These techniques can dominate G-FL at the cost of an offline computation step.

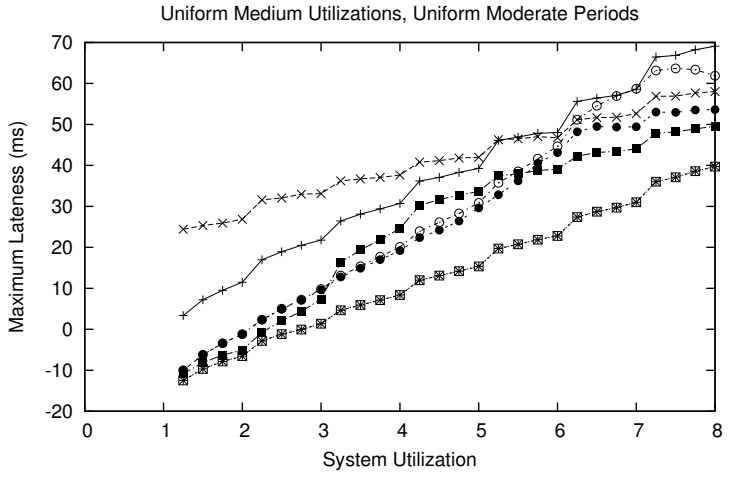
References

Anderson, J.H., Srinivasan, A.: Mixed pfair/erfair scheduling of asynchronous periodic tasks. J. Comput. Syst. Sci. 68(1), 157–204 (Feb 2004)

[4]With uniform heavy utilizations and uniform long periods, EDF-CVA2 provided a very slightly smaller average proportional latenesss bound when the system utilization was 8. Recall that the technique used by EDF-CVA2 only applies when proportional PPs equal minimum separation times.

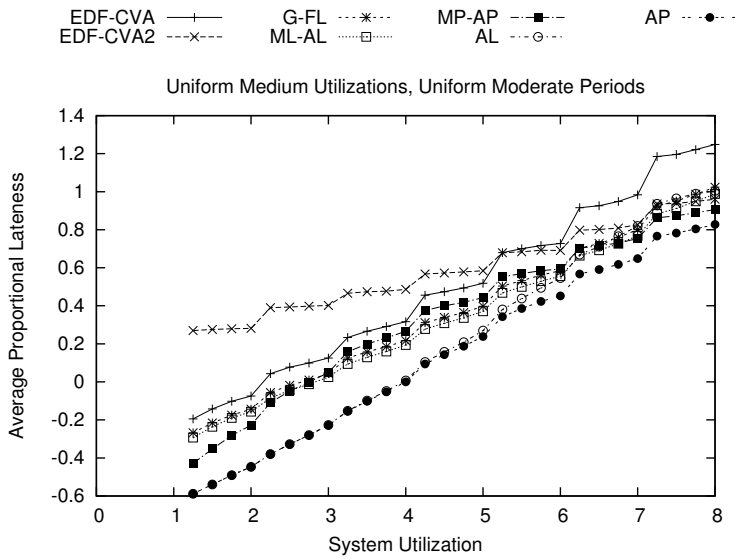


(a) Average Lateness

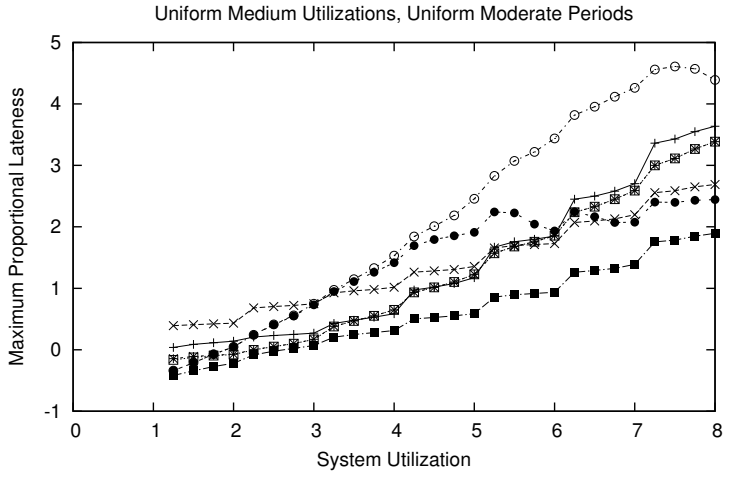


(b) Maximum Lateness

Figure 6: **(a)** Average and **(b)** maximum lateness bound with repsect to system utilization for task systems with uniform medium utilizations and uniform moderate periods.

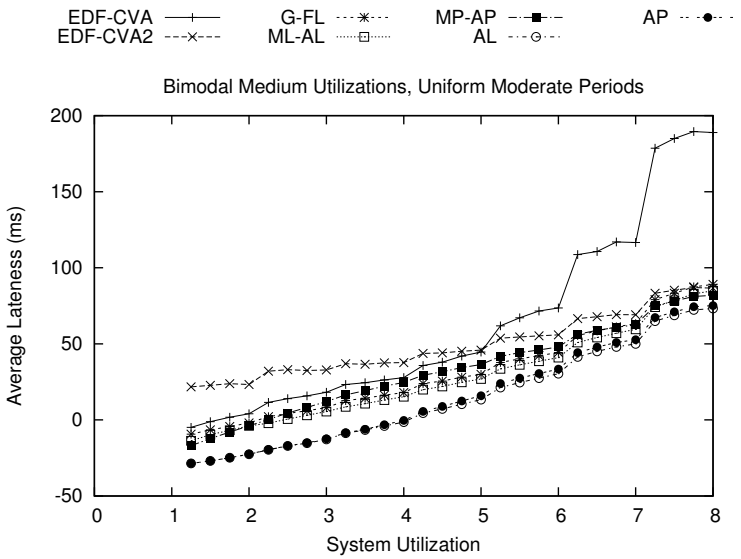


(a) Average Proportional Lateness

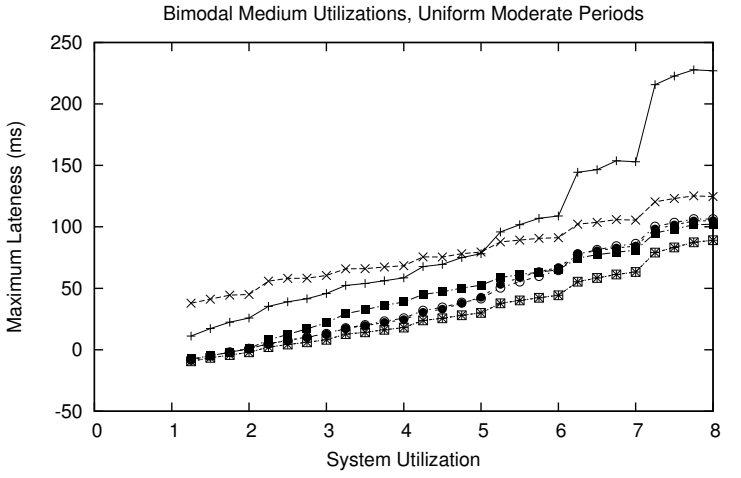


(b) Maximum Proportional Lateness

Figure 7: **(a)** Average and **(b)** maximum proportional lateness bound with respect to system utilization for task systems with uniform medium utilizations and uniform moderate periods.

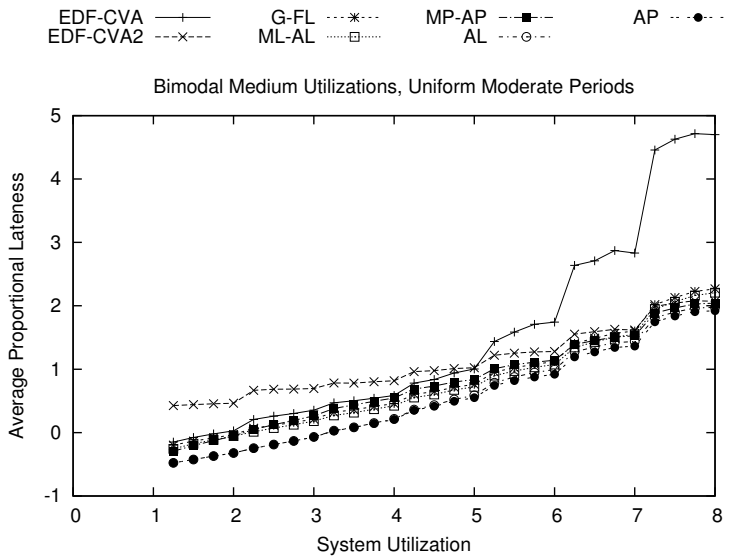


(a) Average Lateness

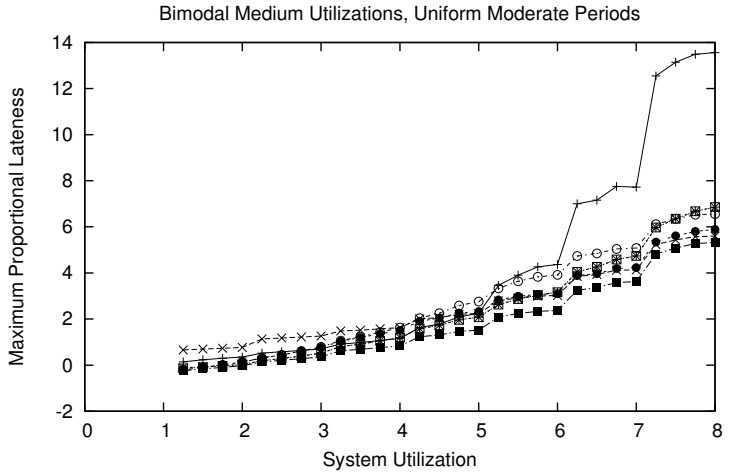


(b) Maximum Lateness

Figure 8: **(a)** Average and **(b)** maximum lateness bound with respect to system utilization for task systems with bimodal medium utilizations and uniform moderate periods.



(a) Average Proportional Lateness



(b) Maximum Proportional Lateness

Figure 9: **(a)** Average and **(b)** maximum proportional lateness bound with respect to system utilization for task systems with bimodal medium utilizations and uniform moderate periods.

Back, H., Chwa, H.S., Shin, I.: Schedulability analysis and priority assignment for global job-level fixed-priority multiprocessor scheduling. In: IEEE Real-Time and Embedded Technology and Applications Symposium. pp. 297–306 (2012)

Baker, T., Cirinei, M., Bertogna, M.: EDZL scheduling analysis. Real-Time Systems 40, 264–289 (2008)

Baruah, S.K., Cohen, N.K., Plaxton, C.G., Varvel, D.A.: Proportionate progress: A notion of fairness in resource allocation. Algorithmica 15, 600–625 (1996)

Baruah, S.K., Mok, A.K., Rosier, L.E.: Preemptively scheduling hard-real-time sporadic tasks on one processor. In: RTSS. pp. 182–190 (1990)

Bastoni, A., Brandenburg, B.B., Anderson, J.H.: An empirical comparison of global, partitioned, and clustered multiprocessor EDF schedulers. In: RTSS. pp. 14–24 (2010)

Brandenburg, B.B.: Scheduling and Locking in Multiprocessor Real-Time Operating Systems. Ph.D. thesis, The University of North Carolina at Chapel Hill (2011)

Chwa, H.S., Back, H., Chen, S., Lee, J., Easwaran, A., Shin, I., Lee, I.: Extending task-level to job-level fixed priority assignment and schedulability analysis using pseudo-deadlines. In: RTSS. pp. 51–62 (2012)

Devi, U.C., Anderson, J.H.: Tardiness bounds under global EDF scheduling on a multiprocessor. Real-Time Sys. 38(2), 133–189 (2008)

Erickson, J.P., Anderson, J.H.: Fair lateness scheduling: Reducing maximum lateness in g-edf-like scheduling. In: ECRTS. pp. 3–12 (2012)

Erickson, J.P., Devi, U., Baruah, S.K.: Improved tardiness bounds for global EDF. In: ECRTS. pp. 14–23 (2010a)

Erickson, J.P., Guan, N., Baruah, S.K.: Tardiness bounds for global EDF with deadlines different from periods. In: OPODIS. pp. 286–301 (2010b), revised version at `http://cs.unc.edu/~jerickso/opodis2010-tardiness.pdf`

Kenna, C., Herman, J., Brandenburg, B.B., Mills, A.F., Anderson, J.H.: Soft real-time on multiprocessors: Are analysis-based schedulers really worth it? In: RTSS. pp. 99–103 (2011)

Lee, J., Easwaran, A., Shin, I.: Maximizing contention-free executions in multiprocessor scheduling. In: RTAS. pp. 235–244 (2011)

Lee, S.K.: On-line multiprocessor scheduling algorithms for real-time tasks. In: IEEE Region 10 Annual International Conference. pp. 607–611 (1994)

Leontyev, H., Anderson, J.H.: Generalized tardiness bounds for global multiprocessor scheduling. Real-Time Sys. 44(1), 26–71 (2010)

Leontyev, H., Chakraborty, S., Anderson, J.H.: Multiprocessor extensions to real-time calculus. In: RTSS. pp. 410–421 (2009)

Megel, T., Sirdey, R., David, V.: Minimizing task preemptions and migrations in multiprocessor optimal real-time schedules. In: RTSS. pp. 37–46 (2010)

Ogryczak, W., Tamir, A.: Minimizing the sum of the k largest functions in linear time. Inf. Process. Lett. 85(3), 117–122 (2003)

Regnier, P., Lima, G., Massa, E., Levin, G., Brandt, S.: RUN: Optimal multiprocessor real-time scheduling via reduction to uniprocessor. In: RTSS. pp. 104–115 (2011)

Srinivasan, A., Baruah, S.K.: Deadline-based scheduling of periodic task systems on multiprocessors. Inf. Process. Lett. 84(2), 93–98 (2002)

Ward, B.C., Erickson, J.P., Anderson, J.H.: A linear model for setting priority points in soft real-time systems. In: Alanfest. pp. 192–205 (March 2013)

A Notation

b	LP auxilliary variable used while computing $G(\vec{x},\vec{Y})$.
C_i	Worst-case execution time for τ_i.
$C_{\max}$	Maximum C_i for τ
D_i	Deadline for τ_i.
G	LP variable corresponding to $G(\vec{x},\vec{Y})$.
$G(\vec{x},\vec{Y})$	Extra demand from certain analyzed tasks (see (13)).
H	Set of jobs with priority at least that of J_i.
I_i	Proportional lateness bound for τ_i.
$I_{\max}$	Maximum proportional lateness bound for τ.
J_i	Arbitrary job under analysis.
L_i	Lateness bound for τ_i.
$L_{\max}$	Maximum lateness bound of G-FL for τ.
m	Processor count.
R_i	Response time bound for τ_i.
s	LP variable corresponding to $G(\vec{x},\vec{Y})+S(\vec{Y})$
$S_i(Y_i)$	Extra demand term for τ_i due to $D_i < T_i$ (see (9)).
S_{sum}	LP variable corresponding to $S(\vec{Y})$.
$S(\vec{Y})$	Sum of $S_i(Y_i)$ terms over task system (see (10)).
t_b	End of last idle interval before y_i.
t_d	Beginning of last idle interval before y_i.
T_i	Minimum separation time for τ_i.
Y_i	Relative PP for τ_i.
U_i	Utilization for τ_i.
U^+	Ceiling of utilization sum for τ.
$W_j(t)$	Total remaining work at time t for jobs of τ_j in H.
$W(t)$	Total remaining work at time t for jobs in H.
x_i	Component of response time bound $Y_i+x_i+C_i$ for τ_i (see (27)).
y_i	PP of J_i.
z_i	LP auxilliary variable used while computing $G(\vec{x},\vec{Y})$.
δ	Constant used to describe already completed work or early completion.
τ	Task system.
τ_i	Task i.

B Additional Proofs

Lemma 1. *If $\ell \geq Y_i$, then*

$$DBF(\tau_i,Y_i,\ell) \leq U_i\ell + C_i\left(1-\frac{Y_i}{T_i}\right).$$

Proof. As first demonstrated in Baruah et al. (1990), for arbitrary-deadline G-EDF,

$$\text{DBF}(\tau_i,\ell) = C_i \times \max\left\{0, \left\lfloor \frac{\ell - D_i}{T_i} \right\rfloor + 1\right\}.$$

Therefore, for arbitrary GEL schedulers, we instead define

$$\mathrm{DBF}(\tau_i, Y_i, \ell) = C_i \times \max\left\{0, \left\lfloor \frac{\ell - Y_i}{T_i} \right\rfloor + 1\right\}. \tag{57}$$

If $\ell \geq Y_i$, then

$$\begin{aligned}
U_i\ell + C_i\left(1 - \frac{Y_i}{T_i}\right) &\geq \{\text{By definition of the floor function}\} \\
&\quad U_i\left(T_i\left\lfloor \frac{\ell - Y_i}{T_i} \right\rfloor + Y_i\right) + C_i\left(1 - \frac{Y_i}{T_i}\right) \\
&= \{\text{Rearranging}\} \\
&\quad C_i\left\lfloor \frac{\ell - Y_i}{T_i} \right\rfloor + \frac{C_iY_i}{T_i} + C_i - \frac{C_iY_i}{T_i} \\
&= \{\text{Cancelling and rearranging}\} \\
&\quad C_i\left(\left\lfloor \frac{\ell - Y_i}{T_i} \right\rfloor + 1\right) \\
&= \{\text{By (57)}\} \\
&\quad \mathrm{DBF}(\tau_i, Y_i, \ell)
\end{aligned}$$

□

Lemma 2. *$DBF(\tau_i, Y_i, \ell) \leq U_i\ell + S_i(Y_i)$.*

Proof. We consider two cases.

We first consider $\ell \in [0, Y_i)$. In that case, $\ell - Y_i < 0$, and $\left\lfloor \frac{\ell - Y_i}{T_i} \right\rfloor + 1 \leq 0$. Therefore, by the definition of $S_i(Y_i)$ in (9), and (57), $\mathrm{DBF}(\tau_i, Y_i, \ell) = 0 \leq U_i\ell + S_i(Y_i)$.

We then consider $\ell \geq Y_i$. By the definition of $S_i(Y_i)$ in (9),

$$U_i\ell + S_i(Y_i) \geq U_i\ell + C_i\left(1 - \frac{Y_i}{T_i}\right).$$

Therefore, by Lem. 1,

$$\mathrm{DBF}(\tau_i, Y_i, \ell) \leq U_i\ell + S_i(Y_i).$$

□

Lemma 5. *If $\vec{x}$ is compliant and each job of τ_j with higher priority than J_i finishes with response time no greater than $Y_j + x_j + C_j$, then J_i finishes with response time no greater than $Y_i + x_i + C_i$.*

Proof. Let δ denote the sum of the amount of execution of J_i before y_i and the amount of time by which it finishes early, so that $C_i - \delta$ units remain at time y_i. By Lem. 4, the remaining work at time y_i for all jobs in H (including J_i itself) is at most $G(\vec{x},\vec{Y}) + S(\vec{Y})$. Therefore, there can be at most $G(\vec{x},\vec{Y}) + S(\vec{Y}) - (C_i - \delta)$ units of competing work from other tasks. (This expression actually bounds all competing work, but is sufficient as an upper bound for work from other tasks.) Because each $Y_i \geq 0$, no job releases after y_i affect the scheduling of J_i. As depicted in Figure 10, a CPU becomes available for τ_i at the earliest time that any CPU completes its competing work. Therefore, in the worst case, all CPUs execute competing work with maximum parallelism, and a CPU becomes available for τ_i at time

$$t_c = y_i + \frac{G(\vec{X},\vec{Y}) + S(\vec{Y}) - C_i + \delta}{m} \tag{58}$$

We consider two cases, depicted in Figure 11:

Case 1 (Figure 11(a)). If some predecessor of J_i is running at time t_c, we denote the immediate predecessor of J_i as J_i'. Because all competing work from other tasks has completed, J_i can begin execution immediately upon completion of J_i'. Therefore, because J_i' has a response time no greater than the assumed response bound (11), and J_i must have been released at least T_i units after J_i', J_i' must complete no later than $Y_i + x_i + C_i - T_i$ units after the release of J_i. Because we assumed $C_i \leq T_i$, J_i must complete within $Y_i + x_i + C_i - T_i + C_i \leq Y_i + x_i + C_i$ units of its release, and the lemma is proven.

Case 2 (Figure 11(b)). If no predecessor of J_i is running at time t_c, J_i is eligible for execution at time t_c unless it has already completed. At most $C_i - \delta$ units of execution remain, so it completes by time

$$\begin{aligned} y_i + \frac{G(\vec{x},\vec{Y}) + S(\vec{Y}) - C_i + \delta}{m} + C_i - \delta &\leq \{\text{Because } \delta > \tfrac{\delta}{m}\} \\ & \quad y_i + \frac{G(\vec{x},\vec{Y}) + S(\vec{Y}) - C_i}{m} + C_i \\ &\leq \{\text{By Def. 1}\} \\ & \quad y_i + x_i + C_i, \end{aligned}$$

and the lemma is proven. □

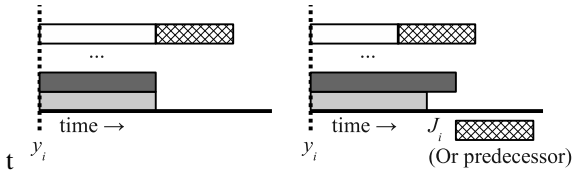
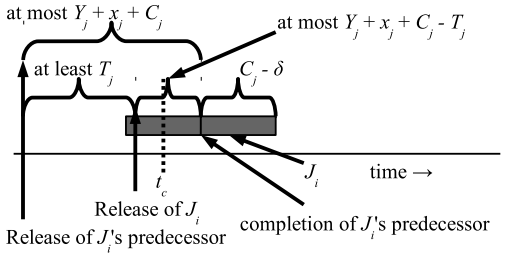
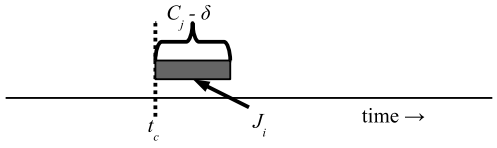


Figure 10: Possible competing work schedules after y_i.



(a) Case 1



(b) Case 2

Figure 11: Cases considered in Lemma 5

Lemma 6. *$\vec{x}$ is a minimum near-compliant vector iff*

$$\forall i, x_i = \frac{G(\vec{x},\vec{Y}) + S(\vec{Y}) - C_i}{m}. \tag{27}$$

Proof. We consider two cases, depending on whether (27) holds.

(27) Holds. If (27) holds, then by Def. 1, $\vec{x}$ is near-compliant. Let $\delta > 0$ and $\vec{x'}$ be such that, for some j,

$$x'_j = x_j - \delta, \tag{59}$$

and for $i \neq j$, $x'_i = x_i$. Then, by the definition of $G(\vec{x},\vec{Y})$ in (13),

$$G(\vec{x'},\vec{Y}) \geq G(\vec{x},\vec{Y}) - U_j\delta. \tag{60}$$

Therefore,

$$\begin{aligned}
\frac{G(\vec{x'},\vec{Y})+S(\vec{Y})-C_j}{m} &\geq \{\text{By (60)}\} \\
&\quad \frac{G(\vec{x},\vec{Y})-U_j\delta+S(\vec{Y})-C_j}{m} \\
&= \{\text{Rearranging}\} \\
&\quad \frac{G(\vec{x},\vec{Y})+S(\vec{Y})-C_j}{m} - \frac{U_j\delta}{m} \\
&= \{\text{By (27)}\} \\
&\quad x_j - \frac{U_j\delta}{m} \\
&> \{\text{Because } U_j > 0, m > 1, \text{ and } \delta > 0, \text{ and by (59)}\} \\
&\quad x'_j
\end{aligned}$$

Therefore, by Def. 1, $\vec{x'}$ is not near-compliant. Since j and δ were arbitrary, $\vec{x}$ is a minimum near-compliant vector.

(27) Does Not Hold. Suppose (27) does not hold. If, for some j, $x_j < \frac{G(\vec{x},\vec{Y})+S(\vec{Y})-C_j}{m}$, then by Def. 1, $\vec{x}$ is not near-compliant. Thus, if $\vec{x}$ is near-compliant, then for some j,

$$x_j > \frac{G(\vec{x},\vec{Y})+S(\vec{Y})-C_j}{m}. \tag{61}$$

If there is more than one such j, we choose one arbitrarily. We define $\vec{x'}$ such that

$$x'_j = \frac{G(\vec{x},\vec{Y})+S(\vec{Y})-C_j}{m}, \tag{62}$$

and for $i \neq j$,

$$x'_i = x_i. \tag{63}$$

By the definition of $G(\vec{x},\vec{Y})$ in (13) and (61)–(62),

$$G(\vec{x'},\vec{Y}) \leq G(\vec{x},\vec{Y}). \tag{64}$$

Therefore,

$$\begin{aligned} x'_j &= \{\text{By (62)}\} \\ &\quad \frac{G(\vec{x},\vec{Y}) + S(\vec{Y}) - C_i}{m} \\ &\geq \{\text{By (64)}\} \\ &\quad \frac{G(\vec{x'},\vec{Y}) + S(\vec{Y}) - C_i}{m}, \end{aligned}$$

and, for $i \neq j$,

$$\begin{aligned} x'_i &= \{\text{By (63)}\} \\ &\quad x_i \\ &\geq \{\text{By Def. 1}\} \\ &\quad \frac{G(\vec{x},\vec{Y}) + S(\vec{Y}) - C_i}{m} \\ &\geq \{\text{By (64)}\} \\ &\quad \frac{G(\vec{x'},\vec{Y}) + S(\vec{Y}) - C_i}{m}. \end{aligned}$$

Thus, $\vec{x'}$ is near-compliant, so $\vec{x}$ is not a *minimum* near-compliant vector. □

Lemma 7. *If the minimum near-compliant vector for the task system exists, it is unique.*

Proof. We use proof by contradiction. Assume that two distinct minimum near-compliant vectors exist for the task system, $\vec{x}$ with

$$\forall i, x_i = \frac{s - C_i}{m}, \tag{65}$$

and $\vec{x'}$ with

$$\forall i, x'_i = \frac{s' - C_i}{m}. \tag{66}$$

Without loss of generality, assume $s' < s$.

We show

$$s = \{\text{By Corollary 1}\}$$

$$
\begin{aligned}
& G(\vec{x},\vec{Y}) + S(\vec{Y}) \\
&= \{\text{By the definition of } G(\vec{x},\vec{Y}) \text{ in (13)}\} \\
& \quad \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + S(\vec{Y}) \\
&= \{\text{By (65)}\} \\
& \quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{s - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + S(\vec{Y}) \\
&= \{\text{Rearranging}\} \\
& \quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{s' - C_i}{m} \right) U_i + \left(\frac{s - s'}{m} \right) U_i + C_i - S_i(Y_i) \right) + S(\vec{Y}) \\
&< \{\text{Because } U^+ \leq m \text{ and } U_i \leq 1 \text{ holds for each } i\} \\
& \quad s - s' + \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{s' - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + S(\vec{Y}) \\
&= \{\text{By (66)}\} \\
& \quad s - s' + \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + S(Y) \\
&= \{\text{By the definition of } G(\vec{x},\vec{Y}) \text{ in (13))}\} \\
& \quad s - s' + G(\vec{x}',\vec{Y}) + S(\tau) \\
&= \{\text{By (66)}\} \\
& \quad s - s' + s' \\
&= \{\text{Cancelling}\} \\
& \quad s. \qquad (67)
\end{aligned}
$$

(67) is a contradiction, so the lemma is proven. □

Lemma 9. *For any assignment of variables satisfying Constraint Sets 1–4,*

$$G + S_{\text{sum}} \geq G(\vec{x},\vec{Y}) + S(\vec{Y}). \qquad (68)$$

Proof.

$$
\begin{aligned}
G + S_{\text{sum}} &\geq \{\text{By (32) and Constraint Set 4}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i) + \sum_{\tau_i \in \tau} S_i \\
&\geq \{\text{By (31); observe that each } S_i \text{ appearing in the first summation} \\
&\quad \text{also appears in the second}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + \sum_{\tau_i \in \tau} S_i(Y_i) \\
&= \{\text{By the definition of } S(\vec{Y}) \text{ in (10), and the definition of } G(\vec{x}, \vec{Y}) \text{ in (13)}\} \\
&\quad G(\vec{x}, \vec{Y}) + S(\vec{Y}).
\end{aligned}
$$

□

Lemma 10. *For any assignment of variables satisfying Constraint Sets 1–5, $\vec{x}$ is a near-compliant vector.*

Proof. Consider an arbitrary assignment of variables satisfying Constraint Sets 1–5. For arbitrary i,

$$
\begin{aligned}
x_i &= \{\text{By Constraint Set 1}\} \\
&\quad \frac{s - C_i}{m} \\
&\geq \{\text{By Constraint Set 5}\} \\
&\quad \frac{G + S_{\text{sum}} - C_i}{m} \\
&\geq \{\text{By Lemma 9}\} \\
&\quad \frac{G(\vec{x}, \vec{Y}) + S(\vec{y}) - C_i}{m}.
\end{aligned} \tag{69}
$$

By (69) and Definition 1, $\vec{x}$ is a near-compliant vector. □

Lemma 11. *If $s < 0$, then Constraint Sets 1–5 are infeasible.*

Proof. We use proof by contradiction. Assume Constraint Sets 1–5 are satisfied by some assignment of variables with

$$s < 0. \tag{70}$$

$$\begin{aligned}
s &\geq \{\text{By Constraint Set 5}\} \\
&\quad G + S_{\text{sum}} \\
&\geq \{\text{By Lemma 9}\} \\
&\quad G(\vec{x}, \vec{Y}) + S(\vec{Y}) \\
&= \{\text{By (10) and (13)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + \sum_{\tau_i \in \tau} S_i(Y_i) \\
&= \{\text{By Constraint Set 1}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{s - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + \sum_{\tau_i \in \tau} S_i(Y_i) \\
&\geq \{\text{By (70)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{-C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + \sum_{\tau_i \in \tau} S_i(Y_i) \\
&= \{\text{Rearranging}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{C_i(m - U_i)}{m} \right) - S_i(Y_i) \right) + \sum_{\tau_i \in \tau} S_i(Y_i) \\
&\geq \{\text{Because each } S_i(Y_i) \geq 0 \text{ by the definition of } S_i(Y_i) \text{ in (9); observe that each} \\
&\quad S_i(Y_i) \text{ in the first summation also appears in the second}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \frac{C_i(m - U_i)}{m} \\
&\geq \{\text{Because } U_i \leq 1 < m\} \\
&\quad 0.
\end{aligned} \tag{71}$$

(70) contradicts (71), so the lemma is proven. □

Lemma 12. *Constraint Sets 1–5 are feasible.*

Proof. We present an assignment of variables that satisfies Constraint Sets 1–5. Let $C_{\max}$ denote the largest value of C_i in the system. Let

$$s = m^2 C_{\max} + mS(\vec{Y}), \tag{72}$$

$$\forall i, x_i = \frac{s - C_i}{m}, \tag{73}$$

$$\forall i, S_i = S_i(Y_i), \tag{74}$$

$$G = G(\vec{x}, \vec{Y}), \tag{75}$$

$$b = (U^+ - 1)^{th} \text{ largest value of } x_i U_i + C_i - S_i, \tag{76}$$

$$\forall i, z_i = \max\{0, x_i U_i + C_i - S_i - b\}, \tag{77}$$

$$S_{\text{sum}} = S(\vec{Y}). \tag{78}$$

Constraint Set 1 holds by (73).

Constraint Set 2 holds by (74).

Constraint Set 3 holds by (75)–(77).

Constraint Set 4 holds by (74) and (78).

To see that Constraint Set 5 holds, note that

$$\begin{aligned}
G + S_{\text{sum}} &= \{\text{By (75) and (78)}\} \\
&\quad G(\vec{x}, \vec{Y}) + S(\vec{Y}) \\
&= \{\text{By (13)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + S(\vec{Y}) \\
&= \{\text{By (72)–(73)}\} \\
&\quad \sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{m^2 C_{\max} + mS(\vec{Y}) - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + S(\vec{Y}) \\
&< \{\text{Because } U^+ \leq m, \forall i, C_i \leq C_{\max} \wedge U_i \leq 1\} \\
&\quad \sum_{m - 1 \text{ largest}} (mC_{\max} + S(\vec{Y}) + C_{\max}) + S(\vec{Y}) \\
&= \{\text{Rearranging}\} \\
&\quad (m^2 - 1) C_{\max} + mS(\vec{Y})
\end{aligned}$$

$< \{\text{By (72)}\}$

$s.$

□

Lemma 14. *Let $C_{\max}$ denote the largest value of C_i in the system. If $U^+ > 1$ and $s < C_{\max}$, then Constraint Sets 1–5 are infeasible.*

Proof. We use proof by contradiction. Assume $U^+ > 1$ and Constraint Sets 1–5 are satisfied by some assignment of variables with

$$s < C_{\max}. \tag{79}$$

$s \geq \{\text{By Constraint Set 5}\}$

$G + S_{\text{sum}}$

$\geq \{\text{By Lemma 9}\}$

$G(\vec{x}, \vec{Y}) + S(\vec{Y})$

$= \{\text{By the definition of } S(\vec{Y}) \text{ in (10), and the definition of } G(\vec{x}, \vec{Y}) \text{ in (13)}\}$

$$\sum_{U^+ - 1 \text{ largest}} (x_i U_i + C_i - S_i(Y_i)) + \sum_{\tau_i \in \tau} S_i(Y_i)$$

$= \{\text{By Constraint Set 1}\}$

$$\sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{s - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + \sum_{\tau_i \in \tau} S_i(Y_i)$$

$\geq \{\text{By (79)}\}$

$$\sum_{U^+ - 1 \text{ largest}} \left(\left(\frac{C_{\max} - C_i}{m} \right) U_i + C_i - S_i(Y_i) \right) + \sum_{\tau_i \in \tau} S_i(Y_i)$$

$\geq \{\text{Because } C_{\max} \geq C_i \text{ for all } i\}$

$$\sum_{U^+ - 1 \text{ largest}} (C_i - S_i(Y_i)) + \sum_{\tau_i \in \tau} S_i(Y_i)$$

$\geq \{\text{Because each } S_i(Y_i) \geq 0 \text{ by the definition of } S_i(Y_i) \text{ in (9); observe that each}$

$S_i(Y_i)$ in the first summation also appears in the second.$\}$

$$\sum_{U^+-1 \text{ max}} C_i$$

$$= \{\text{By the definition of } C_{\max}, \text{ because } U^+ > 1\}$$

$$C_{\max}. \tag{80}$$

(79) contradicts (80), so the lemma is proven. □