A Linear Model for Setting Priority Points in Soft Real-Time Systems

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

Department of Computer Science
The University of North Carolina at Chapel Hill

Abstract. The earliest-deadline-first (EDF) scheduling algorithm, while not optimal for globally-scheduled hard real-time systems, can support any feasible task system with bounded lateness. Furthermore, global-EDF-like (GEL) scheduling algorithms, which prioritize jobs by assigning them fixed *priority points* based on per-task *relative priority points*, have been shown to share this same property. Some such algorithms exhibit better lateness bounds than G-EDF. This paper surveys existing research on bounded-lateness analysis for GEL schedulers, and formulates one such analysis technique called *compliant vector analysis* (*CVA*) as a *linear program* (*LP*). Using this LP, per-task relative priority points can be set to optimize for application-specific lateness constraints, and minimize average and/or maximal lateness bounds. An empirical study is presented that compares the lateness bounds of a variety of GEL schedulers and analysis techniques on randomly generated task systems.

1 Introduction

For some types of computing workloads, such as multimedia applications, it is not necessary to use schedulers amenable to *hard real-time analysis* that ensures that all deadlines are met. A larger system utilization can often be supported by instead using a scheduler that is amenable to *soft real-time analysis* that ensures that the *lateness* of any job, or the amount of time between its deadline and its completion, remains bounded.

Although bounded lateness is achievable by a wide range of real-time scheduling algorithms with no utilization loss [6], one class of such algorithms, called *G-EDF-like* (*GEL*) algorithms [7], is of particular interest because such algorithms are straightforward to implement. In a GEL algorithm, each task has a *relative priority point*, and each job has an *absolute priority point* defined by adding its tasks's relative priority point to its release. Jobs are scheduled globally on an earliest-absolute-priority-point-first basis. Devi and Anderson [2] proved that G-EDF itself has bounded lateness, and subsequently improvements to this original analysis were proposed in [3–5]. Erickson and Anderson [3] further proposed the *global fair lateness* (*G-FL*) scheduler, a GEL scheduler that provably provides the smallest maximum (over all tasks) lateness bound possible using current analysis for GEL schedulers. In addition, Leontyev and Anderson [6] analyzed a much more general class of schedulers in which not all processors are necessarily fully available. However, due to its generality, this analysis is not as tight as the analysis in [3].

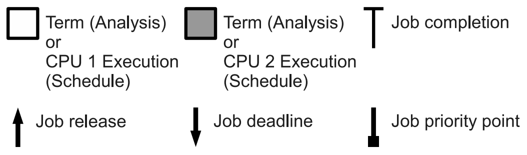

Term (Analysis) or CPU 1 Execution (Schedule)
Term (Analysis) or CPU 2 Execution (Schedule)
Job completion
Job release
Job deadline
Job priority point

Fig. 1. Key for all figures in this paper.

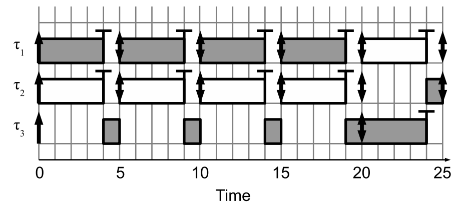
Lateness bounds under GEL schedulers can be computed using an algorithm described in [5], and the maximum lateness bound can be minimized by using G-FL as described in [3]. However, G-FL will ensure the *same* lateness bound for all tasks in a system, even though it is sometimes possible to reduce the lateness of some tasks while maintaining the same maximum lateness bound as G-FL. Furthermore, [3] does not propose a straightforward way to optimize criteria other than maximum lateness. In this paper, we propose to model current lateness analysis using linear programming in order to broaden the set of achievable lateness criteria.

2 SRT Lateness Bounds

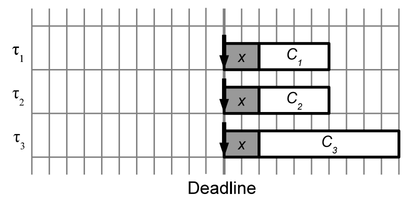
We consider a system τ of n sporadic tasks $\{\tau_1, \tau_2, \ldots, \tau_n\}$ running on m processors. Each task τ_i is defined by a tuple (C_i, T_i), where C_i is the worst-case per-job execution time of τ_i and T_i is the minimum separation time between jobs of τ_i. We assume *implicit deadlines*, *i.e.*, if a job of τ_i is released at time r, then it has a deadline at time $r + T_i$. The *utilization* of τ_i, $U_i = C_i/T_i$, is the long-term processor share it requires. Under any GEL scheduler, τ_i has a scheduler-defined *relative priority point* (*PP*) Y_i and a job of τ_i released at time r has an *absolute PP* at time $r + Y_i$. Jobs with earlier PPs are prioritized over those with later PPs. If a job completes at time c, then it has a *response time* of $c - r$ and a *lateness* of $c - (r + T_i)$.

In our analysis, we assume that time is continuous. For each task τ_i, we also assume that $U_i \leq 1$ and that $\sum_{\tau_i \in \tau} U_i \leq m$. As demonstrated in [2, 6], these conditions are necessary to achieve bounded lateness. We further assume that $n > m$, because otherwise each task can execute on its own processor, and no job of task τ_i will have a response time exceeding C_i.

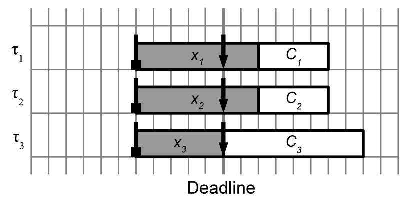
Example. We will illustrate several forms of lateness-bound analysis through an example task system $\tau = \{(4,5), (4,5), (8,20)\}$ on $m = 2$ processors. We will show two types of figures: analysis and actual schedules. We emphasize that the figures depicting analysis do not show actual schedules. All presented forms of analysis determine how long a job can execute after its PP. Such analysis pessimistically assumes that each job cannot begin execution until after some non-negative constant after its PP. Both this constant and the worst-case execution time of the task are terms of the lateness bound. In our analysis figures, we align deadlines rather than PPs to depict lateness bounds. The key for all figures is shown in Fig. 1.



(a) Example G-EDF schedule.



(b) Devi and Anderson's lateness analysis



(c) CVA lateness analysis (see Sec. 2.2).

Fig. 2. (a) shows a G-EDF schedule of the example task system $\tau = \{(4,5),(4,5),(8,20)\}$, and (b) and (c) show the lateness analysis of G-EDF from [2] and CVA [3–5], respectively. In (b) and (c), the deadlines of all tasks are aligned to depict how lateness bounds compare among tasks.

2.1 Devi and Anderson's Lateness Bound

Lateness bounds for G-EDF were first provided by Devi and Anderson [2]. Let $C_{\min}$ be the smallest C_i value of τ_i in τ. (In our example, $C_{\min} = 4$.) Letting

$$x = \frac{\sum_{m-1 \text{ largest}} C_i - C_{\min}}{m - \sum_{m-2 \text{ largest}} U_i}, \tag{1}$$

Devi and Anderson [2] established $x + C_i$ as a lateness bound for τ_i. In our example,

$$x = \frac{C_3 - C_1}{m} = \frac{8-4}{2} = 2. \tag{2}$$

Therefore, τ_1 and τ_2 each have a lateness bound of $2+4=6$, and τ_3 has a lateness bound of $2+8=10$. A G-EDF schedule of τ is depicted in Fig. 2(a), and Devi and Anderson's analysis is depicted in Fig. 2(b).

2.2 Compliant Vector Analysis

Erickson *et. al.* [4] introduced *compliant-vector analysis* (*CVA*), an improved method to analyze lateness under G-EDF. This analysis was later extended in [3, 5] to apply to systems with arbitrary *PP* assignments. We present this more general analysis and apply it to G-EDF, which is modeled by setting $Y_i = T_i$ for each τ_i.

CVA is similar to the analysis described in [2], but rather than defining a single x for the task system so that the response time of each task τ_i is at most $T_i + x + C_i$, a vector $x = \langle x_1, x_2, \ldots, x_n \rangle$ is derived that ensures that the response time of each task τ_i is at most

$$Y_i + x_i + C_i. \tag{3}$$

Note that response times are determined based on Y_i rather than T_i.

We apply the following optimization rule before analysis.

> **PP Reduction Rule.** If all relative PPs in the system are decreased by the same constant, then the ordering of absolute PPs will not change, so the resulting schedule will not change. Under CVA, lateness bounds are minimized for a given GEL scheduler when, analytically, all relative PPs are reduced by the smallest relative PP, so that the smallest relative PP becomes zero [3].

We note that a more complicated alternative optimization rule is presented in [4], which is not compatible with the PP Reduction Rule because it assumes that $Y_i = T_i$. For simplicity of presentation we do not describe this rule here, but we consider it in our experimental study in Sec. 3.

In order to find lateness bounds for a system, the vector x must be computed. In [3], the authors demonstrate that there exists a constant s such that $x_i = v_i(s)$ for each i where

$$v_i(s) = \frac{s - C_i}{m}. \tag{4}$$

As a result, determining the correct value for s is sufficient to determine the value of the entire vector x. In order to determine this value of s, the authors of [3] define

$$S_i = C_i \cdot \max\left\{0, 1 - \frac{Y_i}{T_i}\right\}, \tag{5}$$

$$G(s) = \sum_{m-1 \text{ largest}} (v_i(s)U_i + C_i - S_i), \tag{6}$$

and

$$S(\tau) = \sum_{\tau_i \in \tau} S_i. \tag{7}$$

They demonstrate that the smallest lateness bounds valid under CVA occur iff

$$s = G(s) + S(\tau). \tag{8}$$

An algorithm to compute the necessary s, along with a proof that a unique such s exists, is provided in [5]. It works by finding the zero of

$$M(s) = G(s) + S(\tau) - s. \tag{9}$$

By the uniqueness of s, $M(s)$ will have one zero. Because each of (4)–(7) above is piecewise linear with respect to s, $M(s)$ is piecewise linear with respect to s. The computation algorithm works by attempting to find a zero between each pair of points where the slope of $M(s)$ changes.

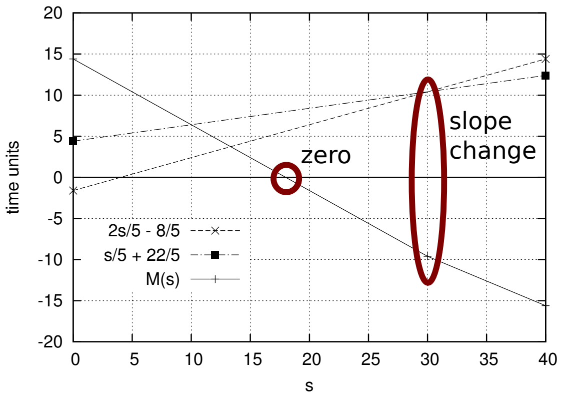


Fig. 3. Graph of expressions used to compute CVA bounds for G-EDF.

Here we will demonstrate the computation algorithm via our example system. For G-EDF, each relative PP Y_i is initially T_i. However, after applying the PP Reduction Rule, $Y_1 = Y_2 = 0$ and $Y_3 = 15$. By (5),

$$S_1 = S_2 = 4 \cdot (1 - 0) = 4 \tag{10}$$

and

$$S_3 = 8 \cdot \left(1 - \frac{15}{20}\right) = 2. \tag{11}$$

Thus, by (6):

$$\begin{aligned} G(s) &= \max\left\{\frac{s-4}{2} \cdot \frac{4}{5} + 4 - 4, \frac{s-8}{2} \cdot \frac{8}{20} + 8 - 2\right\} \\ &= \max\left\{\frac{2s}{5} - \frac{8}{5}, \frac{s}{5} + \frac{22}{5}\right\}, \end{aligned} \tag{12}$$

and by (7),

$$S(\tau) = 4 + 4 + 2 = 10. \tag{13}$$

Because $S(\tau)$ is a constant and s is a simple linear function, by (9), the slope of $M(s)$ can only change where the slope of $G(s)$ changes. The expressions in (12) and the resulting $M(s)$ are depicted in Fig. 3.

In order to determine where the slope of $M(s)$ can change, we now compute where the two expressions in (12) intersect:

$$\begin{aligned} \frac{2s}{5} - \frac{8}{5} &= \frac{s}{5} + \frac{22}{5} \\ \frac{s}{5} &= 6 \\ s &= 30. \end{aligned}$$

Therefore, for $s \leq 30$,

$$G(s) = \frac{s}{5} + \frac{22}{5}, \tag{14}$$

and for $s > 30$,

$$G(s) = \frac{2s}{5} - \frac{8}{5}. \tag{15}$$

We will first try to find a zero of $M(s)$ assuming $s \leq 30$. In this case,

$$\begin{aligned} M(s) &= \{\text{By (9)}\} \\ &\quad G(s) + S(\tau) - s \\ &= \{\text{By (13) and (14)}\} \\ &\quad \frac{s}{5} + \frac{22}{5} + 10 - s \\ &= \{\text{Rearranging}\} \\ &\quad -\frac{4s}{5} + \frac{72}{5}. \end{aligned} \tag{16}$$

Setting $M(s)$ to 0 in (16),

$$s = \left(\frac{72}{5}\right)\left(\frac{5}{4}\right) = 18. \tag{17}$$

$s = 18$ satisfies our assumption that $s \leq 30$, so 18 is indeed a zero of $M(s)$. As demonstrated in [4], the zero of $M(s)$ is unique, so we do not need to consider $s > 30$. Therefore, by (4),

$$x_1 = x_2 = \frac{18-4}{2} = 7, \tag{18}$$

and by (3), the maximum response time of any job of τ_1 or τ_2 is $0+7+4 = 11$, which gives a maximum lateness bound of $11 - 5 = 6$. Also, by (4)

$$x_3 = \frac{18-8}{2} = 5, \tag{19}$$

and by (3), the maximum response time of any job of τ_3 is $15+5+8 = 28$, which gives a maximum lateness bound of $28 - 20 = 8$. These results are depicted graphically in Fig. 2(c).

2.3 G-FL

In [3], the authors also propose G-FL, an algorithm that has the optimal maximum lateness bound over all GEL schedulers using CVA. Rather than using $Y_i = T_i$, G-FL uses $Y_i = T_i - \frac{m-1}{m}C_i$. Therefore, in our example system,

$$Y_1 = Y_2 = 5 - \frac{1}{2} \cdot 4 = 3 \tag{20}$$

and

$$Y_3 = 20 - \frac{1}{2} \cdot 8 = 16. \tag{21}$$

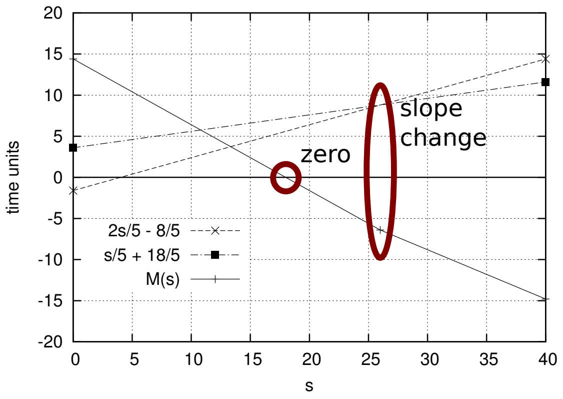


Fig. 4. Graph of expressions used to compute CVA bounds for G-FL.

However, by the PP Reduction Rule from Sec. 2.2, we can instead use $Y_1 = Y_2 = 0$ and $Y_3 = 13$ in the analysis. Here, we repeat the analysis from Sec. 2.2 as applied to G-FL.

By (5),

$$S_1 = S_2 = 4 \cdot (1 - 0) = 4 \tag{22}$$

and

$$S_3 = 8 \cdot \left(1 - \frac{13}{20}\right) = \frac{14}{5}. \tag{23}$$

Thus, by (6):

$$\begin{aligned} G(s) &= \max\left\{\frac{s-4}{2} \cdot \frac{4}{5} + 4 - 4, \frac{s-8}{2} \cdot \frac{8}{20} + 8 - \frac{14}{5}\right\} \\ &= \max\left\{\frac{2s}{5} - \frac{8}{5}, \frac{s}{5} + \frac{18}{5}\right\}, \end{aligned} \tag{24}$$

and by (7),

$$S(\tau) = 4 + 4 + \frac{14}{5} = \frac{54}{5}. \tag{25}$$

The expressions in (24) and the resulting $M(s)$ are displayed in Fig. 4. As before, $M(s)$ can change slope only when $G(s)$ changes slope, so we now compute where the two expressions in (24) intersect:

$$\begin{aligned} \frac{2s}{5} - \frac{8}{5} &= \frac{s}{5} + \frac{18}{5} \\ \frac{s}{5} &= \frac{26}{5} \\ s &= 26. \end{aligned}$$

Therefore, for $s \leq 26$,

$$G(s) = \frac{s}{5} + \frac{18}{5}, \tag{26}$$

and for $s > 26$,

$$G(s) = \frac{2s}{5} - \frac{8}{5}. \tag{27}$$

We will first try to find a zero of $M(s)$ assuming $s \leq 26$. In this case,

$$\begin{aligned} M(s) &= \{\text{By (9)}\} \\ &\quad G(s) + S(\tau) - s \\ &= \{\text{By (25) and (26)}\} \\ &\quad \frac{s}{5} + \frac{18}{5} + \frac{54}{5} - s \\ &= \{\text{Rearranging}\} \\ &\quad -\frac{4s}{5} + \frac{72}{5}. \end{aligned} \tag{28}$$

Compared to Sec. 2.2, notice that the increased S_3 term in $G(s)$ cancels out the increased S_3 term in $S(\tau)$, so (16) and (28) are identical.

Setting $M(s)$ to 0 in (28),

$$s = \left(\frac{72}{5}\right)\left(\frac{5}{4}\right) = 18. \tag{29}$$

$s = 18$ satisfies our assumption that $s \leq 26$, so 18 is indeed a zero of $M(s)$. As before, because the zero of $M(s)$ must be unique, we do not need to consider $s > 26$. Therefore, by (4),

$$x_1 = x_2 = \frac{18-4}{2} = 7, \tag{30}$$

and by (3), the maximum response time of any job of τ_1 or τ_2 is $0+7+4 = 11$, which gives a maximum lateness bound of $11-5 = 6$. Also, by (4)

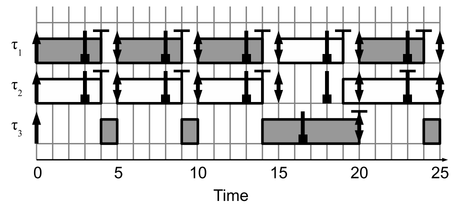
$$x_3 = \frac{18-8}{2} = 5. \tag{31}$$

By (3), the maximum response time of any job of τ_3 is $13+5+8 = 26$, which gives a maximum lateness bound of $28-20 = 6$. Thus, all tasks have the same lateness bound, as claimed.

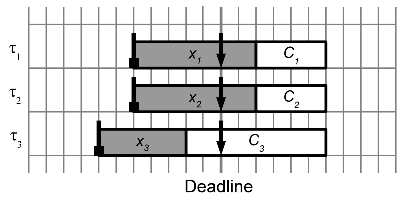
A G-FL schedule of the example system is depicted in Fig. 5(a), and its associated analysis is depicted in Fig. 5(b). Although all analytical bounds are identical, due to the pessimism of the bounds the actual lateness is not necessarily identical.

2.4 Linear Programming

Next, we show how CVA can be formulated as a LP. Such a formulation allows system designers to optimize lateness bounds for arbitrary (linear) optimization objectives by adjusting PPs. We show how to use this LP formulation of CVA to minimize average lateness, and to minimize average lateness while maintaining the same maximum



(a) G-FL schedule.



(b) G-FL lateness bounds.

Fig. 5. Illustration of the G-FL schedule (a) and lateness bounds (b). Note that the deadlines are aligned in (b) to depict how lateness compares among tasks.

lateness as G-FL. Furthermore, using this LP formulation, average lateness can be minimized while ensuring that application-specific lateness tolerances are all satisfied (if it possible under CVA).

Recall that under CVA, to find the lateness bound of a task τ_i, $x_i = v_i(s)$ must be computed using Equations (4)-(8). We show how these equations can be reformulated as constraints in a LP that minimizes average lateness (or possibly other objectives). We assume that x_i, Y_i, S_i, $S(\tau)$, $G(s)$, and s are variables, and we also introduce the auxiliary variables b and z_i. All other values are constants (*i.e.*, U_i, C_i, and m).

Constraint Set 1 *The linear constraints corresponding to (4) are given by*

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

Constraint Set 2 *The linear constraints corresponding to (5) are given by*

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

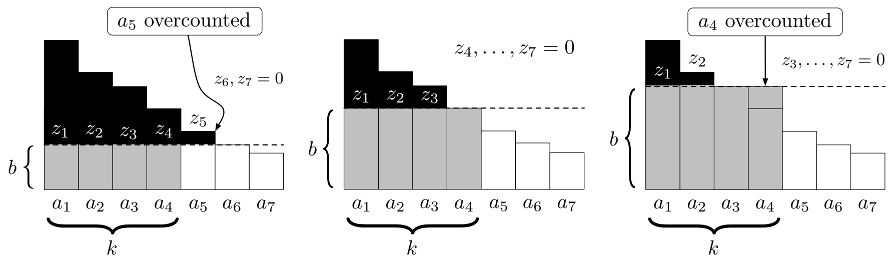
The two constraints in Constraint Set 2 model the two terms of the max in (5). If $C_i(1 - Y_i/T_i) \leq 0$, then the constraint $S_i \geq 0$ ensures that the value of S_i is correct. If $C_i(1 - Y_i/T_i) \geq 0$, then the constraint $S_i \geq C_i(1 - Y_i/T_i)$ ensures the value of S_i is correct.

The next equation to be linearized, (6), 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 presented in [8]. The intuition behind this approach is shown in Fig. 6. This figure corresponds to the following linear program

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

in which G, b, and z_i are variables and both k and a_i are constants. In Fig. 6, 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 Fig. 6 (b).

Using this technique, we can formulate (6) as a set of linear constraints.



a_5 overcounted
$z_6, z_7 = 0$
z_1 z_2 z_3 z_4 z_5
b
a_1 a_2 a_3 a_4 a_5 a_6 a_7
k
$z_4, \dots, z_7 = 0$
z_1 z_2 z_3
b
a_1 a_2 a_3 a_4 a_5 a_6 a_7
k
a_4 overcounted
$z_3, \dots, z_7 = 0$
z_1 z_2
b
a_1 a_2 a_3 a_4 a_5 a_6 a_7
k

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

Fig. 6. Illustration of the auxiliary variables used to sum the largest k elements in the set $A = \{a_1, \dots, a_n\}$. 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.

Constraint Set 3 *The linear constraints corresponding to (6) are given by*

$$G(s) = b(m-1) + \sum_{\tau_i \in \tau} z_i,$$
$$\forall i : z_i \geq 0,$$
$$\forall i : z_i \geq x_i U_i + C_i - S_i - b.$$

In the optimization objectives we consider, $G(s)$ is not itself explicitly minimized, as in the example linear program. However, the objective functions we consider minimize average lateness subject to some constraints, and because the lateness of all tasks is a function of $G(s)$, $G(s)$ is also minimized.[1]

Constraint Set 4 *The linear constraint corresponding to (7) is given by*

$$S(\tau) = \sum_{\tau_i \in \tau} S_i.$$

Constraint Set 5 *The linear constraint corresponding to (9) is given by*

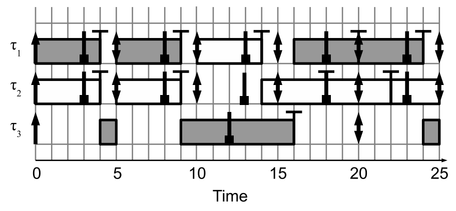
$$G(s) + S(\tau) = s.$$

Constraint Sets 1–5 model CVA through a set of linear constraints. Next we show how these constraints can be coupled with an optimization objective to find optimal priority point settings under CVA. We refer to schedulers produced by setting priority points using our LP formulation of CVA, as *G-LP*. First, we show how G-LP can be optimized to minimize average lateness.

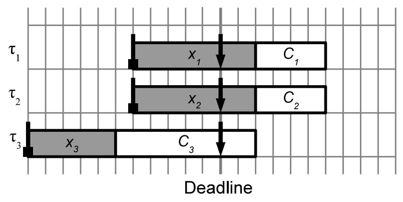
Minimizing Average Lateness. The following linear program may be solved to minimize average lateness under CVA.

$$\text{Minimize}: \sum_{\tau_i \in \tau} Y_i + x_i$$
$$\text{Subject to}: \text{Constraint Sets 1–5}$$

[1] If $G(s)$ is not minimized, then the resulting lateness bounds are still correct, just not as tight.



(a) G-LP Schedule.



(b) G-LP Analysis.

Fig. 7. Illustration of the G-FL schedule (a) and lateness bounds (b). Note that the deadlines are aligned in (b) to depict how lateness compares among tasks.

Note that average lateness is given by $\sum_{\tau_i \in \tau}(Y_i + x_i + C_i - T_i)/n$, but C_i, T_i, and n are all constants that are not necessary to include in the optimization objective. We denote G-LP optimized for average lateness as G-LP-AL.

While G-LP-AL is optimal with respect to average lateness, as is shown experimentally in Sec. 3, 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. The following linear program may be solved to minimize the average lateness under CVA while maintaining the same maximum lateness bound as G-FL.

$$\begin{aligned} \text{Minimize}: &\sum_{\tau_i \in \tau} Y_i + x_i \\ \text{Subject to}: &\ \forall i: Y_i + x_i + C_i - T_i \leq L_{max},^{2} \\ &\ \text{Constraint Sets 1–5} \end{aligned}$$

As before, the constants C_i, T_i and n are omitted from the objective function. We denote the scheduler produced by this program as G-LP-FL.

For our previous example task system, both G-LP-AL and G-LP-FL set the priority point of τ_3 to $Y_3 = 12$, instead of $Y_3 = 16$ as in G-FL. This priority point assignment improves the lateness bound of τ_3 from 6 under G-FL to 2 for G-LP. A schedule and a visual depiction of the analysis are given in Fig. 7. Table 1 summarizes the lateness bounds under different schedulers and analysis techniques for our running example. Note that, in general, G-LP-AL and G-LP-FL do not produce the same schedules, as will be evident in Sec. 3.

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

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

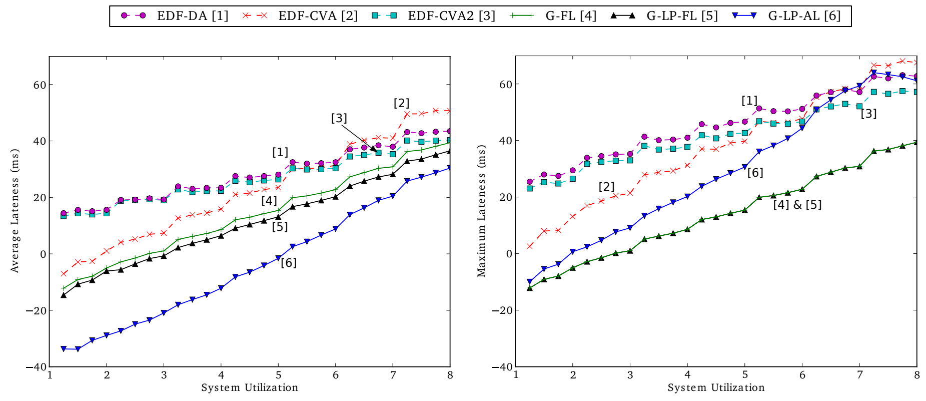
Scheduler	Analysis	τ_1	τ_2	τ_3
G-EDF	Devi and Anderson [2]	6	6	10
G-EDF	CVA [3]	6	6	8
G-FL	CVA [3]	6	6	6
G-LP-AL	CVA [3] & LP	6	6	2
G-LP-FL	CVA [3] & LP	6	6	2

Table 1. Lateness bounds under different schedulers and analysis assumptions for the example task system $\tau = \{(4,5),(4,5),(8,20)\}$ on $m = 2$ processors.

3 Experiments

In this section, we present experiments that demonstrate how G-LP can improve lateness bounds over existing scheduling algorithms. We generated random task sets using a similar experimental design as in previous studies (*e.g.*, [3]). We generated implicit-deadline task sets in which per-task utilizations were distributed either uniformly, bimodally, or exponentially. 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 $5/9$. For task systems with exponentially distributed utilization, per-task utilizations were distributed exponentially with a mean of 0.10, 0.25 or 0.50. 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 [1]. 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 [2] (EDF-DA), CVA analysis of G-EDF scheduling (EDF-CVA) using the PP Reduction Rule, CVA analysis of G-EDF using the alternative optimization rule in [4] (EDF-CVA2), G-FL, and G-LP when optimizing for average lateness subject to the constraint that no task has a lateness bound greater than it would have under G-FL (G-LP-FL), and G-LP when optimizing for average lateness without constraining maximum lateness (G-LP-AL). In Fig. 8, we show the mean average and maximum lateness bounds for each total system utilization value over all generated task systems from a representative combination of per-task utilization and period distributions, medium and moderate, respectively. Note that the lateness bound results are analytical, and that in an actual schedule observed latenesses may be smaller.



(a) Average lateness.　　(b) Maximum lateness.

Fig. 8. Average and maximum lateness bounds under the different analysis techniques and schedulers discussed for a system with moderate periods and uniform medium utilizations.

Observation 1. EDF-CVA2 dominates EDF-DA with respect to both average and maximum lateness bounds. For task systems will small utilizations, the PP Reduction Rule of EDF-CVA outperforms the optimization in EDF-CVA2 [4]; however, the converse is true for large utilizations. Furthermore, for large utilizations, EDF-DA can have lower lateness bounds than EDF-CVA.

This can be seen in insets (a) and (b) of Fig. 8. EDF-CVA2 always dominates EDF-DA, and EDF-CVA has slightly smaller lateness bounds than EDF-DA for task systems with total utilization less than six, while for utilizations greater than six, EDF-DA has better lateness bounds than EDF-CVA. Although the optimization from [4] in EDF-CVA2 performs very well for task systems with large utilizations, it is only applicable if $Y_i = T_i$, for all i. The PP Reduction Rule is applicable to any GEL scheduler.

Observation 2. All three of the GEL schedulers we considered with PPs different from G-EDF, namely, G-FL, G-LP-AL and G-LP-FL, had smaller average lateness bounds than G-EDF as determined via either CVA or Devi and Anderson's analysis.

These three scheduling algorithms optimize, with respect to CVA, either the average or maximum lateness of all tasks by moving PPs. Therefore, these algorithms should have smaller average lateness bounds than G-EDF. This can be observed in inset (a) of Fig. 8.

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

Based on the constraints and the optimization objective of G-LP-FL, the average and maximum lateness bounds are provably no worse than G-FL. As is seen in inset (a) of Fig. 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 G-LP-AL than under G-FL and G-LP-FL. This improvement in average lateness is made possible by allowing for increased maximum lateness.

As a result of the LP optimization, G-LP-AL is optimal with respect to average lateness under CVA. In inset (a) of Fig. 8, we see that the average lateness of G-LP-AL is always smaller than all other schedulers, often by 10-20 ms or more. However, the maximum lateness bounds of G-LP-AL are larger than G-FL and G-LP-FL. In most observed cases, such as in Fig. 8 (b), lateness bounds of G-LP-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, G-LP-AL may be practical in many applications.

We note that the lateness bounds of G-LP-AL in comparison to G-FL and G-LP-FL 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.

4 Conclusions

In this paper, we surveyed existing schedulers and analysis techniques for soft real-time systems with bounded deadline lateness, and showed how to model compliant vector analysis (CVA) as a linear program. The linear formulation of CVA allows the priority points of tasks to be optimally (with respect to CVA) placed to minimize, for example, average or maximum lateness. Scheduling algorithms that result from moving priority points improve average lateness over existing algorithms. Furthermore, the linear formulation of CVA gives system designers flexibility to adapt the scheduling algorithm that is used to the lateness requirements of specific applications. We also generated random task systems and evaluated lateness bounds under a number of different scheduling algorithms and analysis techniques.

References

1. Björn B. Brandenburg. *Scheduling and Locking in Multiprocessor Real-Time Operating Systems*. PhD thesis, University of North Carolina, Chapel Hill, NC, 2011.
2. UmaMaheswari C. Devi and James H. Anderson. Tardiness bounds under global EDF scheduling on a multiprocessor. *Real-Time Systems*, 38(2):133–189, 2008.
3. Jeremy P. Erickson and James H. Anderson. Fair lateness scheduling: Reducing maximum lateness in G-EDF-like scheduling. In *ECRTS*, pages 3–12, 2012.
4. Jeremy P. Erickson, UmaMaheswari C. Devi, and Sanjoy K. Baruah. Improved tardiness bounds for global EDF. In *ECRTS*, pages 14–23, 2010.
5. Jeremy P. Erickson, Nan Guan, and Sanjoy K. Baruah. Tardiness bounds for global EDF with deadlines different from periods. In *OPODIS*, pages 286–301, 2010. Revised version at http://cs.unc.edu/ jerickso/opodis2010-tardiness.pdf.

6. Hennadiy Leontyev and James H. Anderson. Generalized tardiness bounds for global multiprocessor scheduling. *Real-Time Systems*, 44(1):26–71, 2010.
7. Hennadiy Leontyev, Samarjit Chakraborty, and James H. Anderson. Multiprocessor extensions to real-time calculus. In *RTSS*, pages 410–421, 2009.
8. Wlodzimierz Ogryczak and Arie Tamir. Minimizing the sum of the *k* largest functions in linear time. *Information Processing Letters*, 85(3):117–122, 2003.