MISTA 2009

Modeling Preemptive EDF and FP by Integer Variables

Enrico Bini

Abstract The design of any system can be modeled by an optimization problem, where a decision must be taken to maximize an overall utility function within some constraints (that can be physical, contractual, etc.). In hard real-time systems the constraints are specified by the deadlines that are set for the completion of tasks. However classic schedulability tests are formulated by algorithms that prevent a visualization of the feasible region of the designer choices.

In this paper we formulate the EDF and FP exact schedulability conditions on a single processor through a combination of linear constraints. We believe that this alternate representation is better suited for optimization and can trigger the development of more effective design methodologies for real-time systems.

1 Introduction

The design of any object or system can be formalized by an optimization problem: some decisions must be taken such that an overall utility function is maximized within some constraints. In real-time systems the feasibility constraint is expressed by a deadline requirement: a task is required to complete no later than a specified time (*deadline*) after its activation. Hence, it becomes crucial to find a representation of this constraint that is well suited for optimization techniques.

Classic feasibility tests for Fixed Priority (FP) and Earliest Deadline First (EDF) schedulers [14,4] do not reveal the geometry of the feasible parameters. Hence some efforts must be dedicated to formulate the feasibility (deadline) constraint by alternate expressions that enable an efficient implementation of the system design by optimization techniques.

The schedule of jobs with known execution time is often represented by the start time of each job as variable. In fact once all the start times are given a schedule is unambiguously determined. Unfortunately this way of modeling becomes inappropriate when the number of variable associated to the start times becomes infinite or unknown, as in the following circumstances.

Enrico Bini
Scuola Superiore Sant'Anna, Pisa, Italy
E-mail: e.bini@sssup.it

1. If the lifetime of the system is potentially infinite and jobs are activated periodically (and not only a finite number of times) then we should have one start time variable per job. This leads to an infinite number of start times.
2. If the scheduling algorithm is preemptive then each job can be broken, by preempting jobs, into many contiguous chunks. This leads to an unknown number of start times (one for each non-preempted chunk).

Finally, the explicit assignment of the job start times may be inconsistent with the adopted scheduling algorithm, such as the EDF scheduling algorithm. EDF is indeed widely used in real-time systems to schedule jobs with deadline because of its optimality [17]: if it exists a job schedule that do not miss any deadline, then EDF will not miss any deadline.

Hence in this paper we show a representation of the EDF and FP schedulability condition in a way that is well suited for applying optimization techniques on the task parameters. We introduce our notation and terminology. We consider a set $\mathcal{T}$ of m *periodic tasks* executing on a single processor. Each task, denoted by τ_i, requires the execution of a job every *period* T_i. The j^{th} job of τ_i is often denoted by $\tau_{i,j}$. All the jobs of τ_i have the same *execution time* C_i. $\tau_{i,j}$ must complete not later than D_i time units after the activation. D_i is called *deadline* of the task τ_i. The *response time of a job* $R_{i,j}$ is the time that elapses from its activation to its completion. The response time of a task, simply called the *response time*, is $R_i = \sup_j R_{i,j}$. The *critical scenario* for τ_i is the scenario of task activations in which R_i is maximal. If tasks are scheduled by preemptive EDF or FP, the critical scenario for all tasks occurs when all tasks are activated simultaneously [17]. It is then convenient to set the instant when all the tasks start activating their first job equal to 0. Given these hypothesis, it follows that the *activation* of $\tau_{i,j}$ occurs at $a_{i,j} = (j-1)T_i$ and its *absolute deadline* is $d_{i,j} = (j-1)T_i + D_i$.

Task τ_i is also characterized by its *utilization* $U_i = \frac{C_i}{T_i}$. It represents the fraction of time that is required by the task. The sum of all the task utilizations $U = \sum_{i=1}^{m} U_i$ is the *total utilization* of the task set. It is straightforward to see that if $U > 1$, then some deadline is going to be missed because the processor is overloaded.

Sometime we denote the task τ_i also by the triplet (C_i, T_i, D_i) and the task set $\mathcal{T}$ by $(\mathbf{C}, \mathbf{T}, \mathbf{D})$, which are the vectors of computation times, periods, and deadlines, respectively.

All these parameters C_i, T_i, and D_i are real-valued. We extend the notion of least common multiple also to positive real numbers meaning that, given $a, b \in \mathbb{R}_+$ then

$$\mathsf{lcm}(a, b) = \inf\{x \in \mathbb{R}_+ : \exists p, q \in \mathbb{N}_+ \; x = p\,a = q\,b\}$$

that is recursively defined also for a finite set of positive real number $\mathcal{A} = \{a_i\}_{i=1}^{m}$ as follows

$$\mathsf{lcm}(\mathcal{A}) = \mathsf{lcm}(a_i, \mathsf{lcm}(\mathcal{A} \setminus \{a_i\}))$$

We consider both EDF and FP scheduling algorithms [17]. EDF assigns the highest priority to the job which has the earliest absolute deadline $d_{i,j}$. If two jobs have the same absolute deadline, tie is broken arbitrarily. In FP all jobs belonging to the same task have the same *task priority*. When using FP, we assume that tasks are indexed by decreasing priority: $i < j$ implies that τ_i has higher priority than τ_j. Tasks may not have the same priority.

1.1 Related Works

The space of feasible EDF deadlines was derived by Bini and Buttazzo [7] although the main result is justified based on geometric motivations and not formally proven. In the same paper also a convex restriction of the space of feasible deadlines is derived. Baruah and Bini [3] proposed a partitioning algorithm on multiprocessors, based on convenient formalization of the uniprocessor schedulability conditions. Hermant and George [12] proposed an effective simplification of the schedulability test using the simplex method.

Several authors [10, 2, 13] independently proposed different algorithms for computing the minimum deadline of a newly arrived task, assuming the existing task set is feasibly schedulable by EDF. The problem of these methods is that they can hardly be extended to reduce a set of arbitrary deadlines, but can only be applied to a single task at a time, following a given order, as suggested by [13].

When an FP scheduler is adopted, Bini and Buttazzo described the space of computation times, given periods and deadlines [6]. Seto et al [18] described the space of feasible periods. Bini and Di Natale [9] proposed a branch and bound algorithm for assigning the optimal feasible period assignment.

2 EDF schedulability analysis

The schedulability analysis is developed to check whether some deadlines can be missed or not. The classic way of performing schedulability analysis is based on the "demand-supply" approach. In simple words, by this approach it is required that the work demanded by the task set does not exceed the time supplied by the execution platform.

The necessary and sufficient schedulability condition for EDF was derived by Baruah et al. [4,5].

Theorem 1 (Lemma 3 in [5]) *The task set* $\mathcal{T} = \{(C_i, T_i, D_i) : i = 1, \ldots, m\}$ *is schedulable by EDF* ***if and only if****:*

$$\forall t \geq 0 \quad \sum_{i=1}^{m} \max\left\{0, \left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor\right\} C_i \leq t \tag{1}$$

The necessary and sufficient condition of Eq. (1) cannot be used in practice, since it requires to check an infinite number of inequalities. Several works addressed the problem of reducing the number of instants where the inequality of Eq. (1) can be checked while preserving the necessity of the condition. First it can be observed that for any pair of adjacent absolute deadlines d_a and d_b, if Eq. (1) is true at d_a then it is also true $\forall t \in [d_a, d_b[$, because the left hand side of Eq. (1) remains constant whereas the right hand side increases. This allows to restrict the test to the set of all the absolute deadlines. The set of deadlines can be further reduced to a finite set by using linear upper bounds of the demand [5].

2.1 Integer Problem Formulation

Unfortunately, the condition expressed by Theorem 1 is not well suited to be used in optimizations, because the presence of the floor $\lfloor \cdot \rfloor$ operator breaks any property of

the constraints that is desirable for optimization (such as linearity). For this reason, some efforts have been devoted to the derivation of alternative way to formulate the necessary and sufficient condition for EDF schedulability.

The following Theorem provides a convenient way to formulate an equivalent condition of Theorem 1 that is expressed by a combination of linear constraints.

Theorem 2 *The task set* $\mathcal{T} = \{(C_i, T_i, D_i) : i = 1, \ldots, m\}$ *is schedulable by EDF* ***if and only if****:*

$$\forall \mathbf{k} \in \mathbb{N}^m \setminus \{\mathbf{0}\} \quad \exists i \in I_{\mathbf{k}} \quad (T_i - C_i)k_i - \sum_{j \neq i} C_j k_j \geq T_i - D_i \tag{2}$$

where

$$I_{\mathbf{k}} = \{j : k_j \neq 0\}$$

is the set of non-zero indexes in $\mathbf{k}$.

Proof We will prove that Equations (1) and (2) are equivalent.

Eq. (1) $\Rightarrow$ Eq. (2). We are given a vector $\mathbf{k} \in \mathbb{N}^m$ different that $\mathbf{0} \in \mathbb{N}^m$ and we must find an index i in $I_{\mathbf{k}} = \{j : k_j \neq 0\}$ using the hypothesis of Eq. (1).

Since $\mathbf{k} \neq \mathbf{0}$ then $I_{\mathbf{k}} \neq \emptyset$. Let us define

$$\forall j \in I_{\mathbf{k}} \qquad d_j = D_j + (k_j - 1)T_j$$

which is the absolute deadline of the k_j job of τ_j. Notice that from $k_j \geq 1$ it follows that $d_j \geq 0$.

We claim that the index i satisfying Eq. (2) is such that

$$d_i = \max_{j \in I_{\mathbf{k}}} \{d_j\} \tag{3}$$

This value is well defined because $I_{\mathbf{k}} \neq \emptyset$.

Now we exploit the Equation (1) for $t = d_i$. We have

$$\begin{aligned}
&\sum_{j=1}^{m} \max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\} C_j \leq d_i \\
&\max\left\{0, \left\lfloor \frac{d_i + T_i - D_i}{T_i} \right\rfloor\right\} C_i + \sum_{j \neq i} \max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\} C_j \leq d_i \\
&\max\{0, k_i\} C_i + \sum_{j \neq i} \max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\} C_j \leq D_i + (k_i - 1)T_i \\
&(T_i - C_i)k_i - \sum_{j \neq i} \max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\} C_j \geq T_i - D_i
\end{aligned} \tag{4}$$

Equation (4) is quite similar to Equation (2) that we want to prove. To conclude the demonstration we need to find a lower bound of $\max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\}$. We have

$$\begin{aligned}
\forall j \in I_{\mathbf{k}} \quad \max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\} &\geq \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor \geq \\
&\geq \left\lfloor \frac{d_j + T_j - D_j}{T_j} \right\rfloor = k_j
\end{aligned} \tag{5}$$

Similarly

$$\forall j \notin I_{\mathbf{k}} \quad \max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\} \geq 0 = k_j \tag{6}$$

Finally, from Equation (4) by using the lower bounds of Equations (5) and (6), it follows that for the index i selected such that Eq. (3) holds, we have

$$(T_i - C_i)k_i - \sum_{j \neq i} C_j k_j \geq (T_i - C_i)k_i - \sum_{j \neq i} \max\left\{0, \left\lfloor \frac{d_i + T_j - D_j}{T_j} \right\rfloor\right\} C_j \geq T_i - D_i$$

as required.

Eq. (2) $\Rightarrow$ Eq. (1). Given $t \geq 0$, we build the mapping $t \mapsto \mathbf{k} = (k_1, \ldots, k_m) \in \mathbb{N}^m$ defined by

$$k_j = \max\left\{0, \left\lfloor \frac{t + T_j - D_j}{T_j} \right\rfloor\right\}$$

Given this mapping, the Equation (1) that needs to be proved can be rewritten as

$$\forall t \geq 0 \quad \sum_{j=1}^{m} k_j C_j \leq t \tag{7}$$

For all the $t \geq 0$ such that $\mathbf{k} = \mathbf{0}$ we have

$$\sum_{j=1}^{m} k_j C_j = 0 \leq t$$

On the other hand, for all the $t \geq 0$ such that $\mathbf{k} \neq \mathbf{0}$ we can invoke Equation (2) from which it follows that we have an index $i \in I_{\mathbf{k}}$ such that

$$(T_i - C_i)k_i - \sum_{j \neq i} C_j k_j \geq T_i - D_i$$

$$\sum_{j=1}^{m} C_j k_j \leq (k_i - 1)T_i + D_i = \left(\left\lfloor \frac{t + T_i - D_i}{T_i} \right\rfloor - 1\right) T_i + D_i \leq \frac{t - D_i}{T_i} T_i + D_i$$

$$\sum_{j=1}^{m} C_j k_j \leq t$$

Hence Equation (7) is demonstrated and the Theorem is proved.

Theorem 2 formulates the EDF schedulability by introducing the integers $\mathbf{k}$. However the "physical" interpretation of Eq. (2) is lost, whereas Eq. (1) has the clear interpretation of requiring that the amount of demand should not exceed the time available to schedule the tasks. To extract a posteriori the physical interpretation of Theorem 2 then we rewrite Eq. (2) as follows

$$\forall \mathbf{k} \in \mathbb{N}^m \setminus \{\mathbf{0}\} \quad \exists i : k_i \geq 1 \quad \mathbf{C} \cdot \mathbf{k} \leq (k_i - 1)T_i + D_i = d_{i,k_i}$$

that is equivalent to

$$\forall \mathbf{k} \in \mathbb{N}^m \setminus \{\mathbf{0}\} \quad \mathbf{C} \cdot \mathbf{k} \leq \max_{k_i \geq 1}\{d_{i,k_i}\} \tag{8}$$

Basically Eq. (8) asserts that the task set $\mathcal{T}$ is schedulable by EDF if and only if for all possible number of jobs of each task (represented by the vector $\mathbf{k}$), the amount of all required computations $\mathbf{C} \cdot \mathbf{k}$ does not exceed the largest absolute deadline.

In the constrained deadline case (i.e. $\forall i \quad D_i \leq T_i$) the schedulability test can be simplified as reported in the following result.

Lemma 1 *The task set* $\mathcal{T} = \{(C_i, T_i, D_i) : i = 1, \ldots, m\}$ *with constrained deadlines is schedulable by EDF* ***if and only if****:*

$$\forall \mathbf{k} \in \mathbb{N}^m \setminus \{\mathbf{0}\} \quad \exists i = 1, \ldots, m \quad (T_i - C_i)k_i - \sum_{j \neq i} C_j k_j \geq T_i - D_i \tag{9}$$

Proof We proceed by showing that if $\forall i \quad D_i \leq T_i$ then Equations (2) and (9) are equivalent.

Eq. (2) $\Rightarrow$ Eq. (9). This is the simpler implication. In fact since $I_{\mathbf{k}} \subseteq \{1, \ldots, m\}$, then the same index i that makes Eq. (2) true, makes true also Eq. (9).

Eq. (9) $\Rightarrow$ Eq. (2). Given $\mathbf{k}$, let i be the index in $\{1, \ldots, m\}$ that satisfies Eq. (9). Let us assume, by absurd, that $i \notin I_{\mathbf{k}}$ that means $k_i = 0$. From Eq. (9) we have

$$\sum_{j=1}^{m} C_j k_j \leq D_i - T_i \leq 0 \quad \Rightarrow \quad \mathbf{k} = \mathbf{0}$$

that is impossible from the hypothesis. Hence the lemma is proven.

From now on we assume to have constrained deadlines to simplify the presentation since the arbitrary deadline model does not present significant additional difficulties.

The schedulability condition expressed in this way is not practical, since it requires to test on an infinite set (i.e. $\mathbb{N}^m \setminus \{\mathbf{0}\}$). However there are methods to reduce the test to a finite set without losing necessity [7]. We do not discuss here these methods any further.

The schedulability condition of Eq. (9), can also been seen as a covering problem. We define

$$\mathsf{domK}_i = \{\mathbf{k} \in \mathbb{Z}^m : \sum_{j \neq i} C_j k_j - (T_i - C_i)k_i \leq D_i - T_i\} \tag{10}$$

that basically is the set of integers that falls in an half-space. Then the schedulability condition of Lemma 1 is equivalent to the following

$$\mathbb{N} \setminus \{\mathbf{0}\} \subseteq \bigcup_{i=1}^{m} \mathsf{domK}_i \tag{11}$$

3 FP schedulability analysis

Here we also report briefly a similar representation of the schedulability constraint that applies to FP schedulers.

In preemptive FP schedulers any task is not affected by the lower priority tasks because lower priority jobs can never execute when some higher priority job has not yet finished. Hence the FP schedulability analysis is developed per task and the schedulability of the entire task set $\mathcal{T}$ is ensured by the simultaneous verification of the schedulability condition for each task. For this reason, without loss of generality, in this section we consider only the schedulability of the lowest priority task τ_m. Moreover we examine only the constrained deadline case, since the extension to the arbitrary deadline case follows standard techniques [15].

Two main classes of tests exist for FP schedulability analysis: the response time analysis [11, 14, 1] and the time demand analysis [16].

In the response time analysis [11, 14, 1] the following algorithm

$$\begin{cases} R_m^{(0)} & \leftarrow C_m \\ R_m^{(k+1)} & \leftarrow C_m + \sum_{i=1}^{m-1} \left\lceil \frac{R_m^{(k)}}{T_i} \right\rceil C_i \end{cases} \tag{12}$$

is iterated until the sequence $R_m^{(k)}$ converges to the response time $R_m \leq D_m$ of τ_m (in this case τ_m is schedulable) or at some iteration it exceeds the deadline D_k (in this case τ_m is not schedulable).

The time demand analysis [16] does not go through the computation of the response time. Basically it looks for an instant within 0 and D_m that is large enough to accommodate both the computation time C_m of τ_m and the maximum number of interferences from higher priority tasks. This idea is represented by the following Theorem

Theorem 3 (Theorem 1 in [16]) *The task τ_m with constrained deadline $D_m \leq T_m$ is schedulable by FP if and only if*

$$\exists t \in [0, D_m] \quad C_m + \sum_{i=1}^{m-1} \left\lceil \frac{t}{T_i} \right\rceil C_i \leq t \tag{13}$$

Both the approaches has the same defect of Theorem 1: the presence of a discontinuous operator (the ceiling $\lceil \cdot \rceil$ in this case) makes difficult to use the schedulability test as a description of the feasible region.

Fortunately, Seto et al. [18] found an alternate expression of the feasibility region. Below we report their result, derived in the implicit deadline hypothesis, extended to constrained deadline case.

Theorem 4 (Proposition 2.1 in [18]) *The task τ_m is schedulable by FP* ***if and only if***

$$\exists \mathbf{k} \in \mathbb{N}^{m-1} \quad \begin{cases} C_m + \sum_{j=1}^{m-1} C_j k_j \leq D_m \\ C_m + \sum_{j=1}^{m-1} C_j k_j \leq k_i T_i \quad i = 1, \ldots, m-1 \end{cases} \tag{14}$$

4 Future Works

In this paper we have illustrated methods for expressing the schedulability constraints (both EDF and FP) by means of a combination of linear constraints. The next step will be the formulation of the design goal by means of a utility function to be maximized, and the optimal solution of the design problem, following similar approaches as proposed by Seto et al. [19] or Bini and Cervin [8].

References

1. Neil C. Audsley, Alan Burns, Mike Richardson, Ken W. Tindell, and Andy J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.
2. Patricia Balbastre, Ismael Ripoll, and Alfons Crespo. Optimal deadline assignment for periodic real-time tasks in dynamic priority systems. In *Proceedings of the* 18^{th} *Euromicro Conference on Real-Time Systems*, pages 65–74, Dresden, Germany, July 2006.
3. Sanjoy Baruah and Enrico Bini. Partitioned scheduling of sporadic task systems: an ILP-based approach. In *Conference on Design and Architectures for Signal and Image Processing*, Bruxelles, Belgium, November 2008.
4. Sanjoy K. Baruah, Rodney Howell, and Louis Rosier. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.
5. Sanjoy K. Baruah, Aloysius K. Mok, and Louis E. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 182–190, Lake Buena Vista (FL), U.S.A., December 1990.
6. Enrico Bini and Giorgio C. Buttazzo. Schedulability analysis of periodic fixed priority systems. *IEEE Transactions on Computers*, 53(11):1462–1473, November 2004.
7. Enrico Bini and Giorgio C. Buttazzo. The space of EDF deadlines; the exact region and a convex approximation. *Real-Time Systems*, 2008.
8. Enrico Bini and Anton Cervin. Delay-aware period assignment in control systems. In *Proceedings of the* 29^{th} *IEEE Real-Time Systems Symposium*, Barcelona, Spain, December 2008.
9. Enrico Bini and Marco Di Natale. Optimal task rate selection in fixed priority systems. In *Proceedings of the* 26^{th} *IEEE Real-Time Systems Symposium*, pages 399–409, Miami (FL), U.S.A., December 2005.
10. Giorgio Buttazzo and Fabrizio Sensini. Optimal deadline assignment for scheduling soft aperiodic task in hard real-time environments. *IEEE Transactions on Computers*, 48(10):1035–1052, October 1999.
11. Paul K. Harter. Response times in level-structured systems. Technical Report CU-CS-269-84, Department of Computer Science, University of Colorado, USA, 1984.
12. Jean-François Hermant and Laurent George. A norm approach for the partitioned edf scheduling of sporadic task systems. In *Proceedings of the* 21^{st} *Euromicro Conference on Real-Time Systems*, Dublin, Ireland, July 2009.
13. Hoai Hoang, Giorgio Buttazzo, Magnus Jonsson, and Stefan Karlsson. Computing the minimum EDF feasible deadline in periodic systems. In *Proceedings of the* 12^{th} *IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, pages 125–134, Sydney, Australia, August 2006.
14. Mathai Joseph and Paritosh K. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, October 1986.
15. John P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadline. In *Proceedings of the* 11^{th} *IEEE Real-Time Systems Symposium*, pages 201–209, Lake Buena Vista (FL), U.S.A., December 1990.
16. John P. Lehoczky, Lui Sha, and Ye Ding. The rate-monotonic scheduling algorithm: Exact characterization and average case behavior. In *Proceedings of the* 10^{th} *IEEE Real-Time Systems Symposium*, pages 166–171, Santa Monica (CA), U.S.A., December 1989.
17. Chung Laung Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.
18. Danbing Seto, John P. Lehoczky, and Lui Sha. Task period selection and schedulability in real-time systems. In *Proceedings of the* 19^{th} *IEEE Real-Time Systems Symposium*, pages 188–198, Madrid, Spain, December 1998.
19. Danbing Seto, John P. Lehoczky, Lui Sha, and Kang G. Shin. On task schedulability in real-time control systems. In *Proceedings of the* 17^{th} *IEEE Real-Time Systems Symposium*, pages 13–21, Washington (DC), U.S.A., December 1996.