Characteristics of EDF Schedulability on Uniform Multiprocessors*

Shelby Funk **Sanjoy Baruah**
The University of North Carolina at Chapel Hill
{shelby, baruah}@cs.unc.edu

Abstract

In uniform multiprocessor platforms, the various processors comprising the multiprocessor platform may have different computing capacities. The focus of this paper is the design of efficient tests for determining whether the Earliest Deadline First Scheduling Algorithm (EDF) can successfully schedule a given real-time task system to meet all deadlines upon a specified uniform multiprocessor platform. Upon uniform multiprocessor platforms, we show that it is often far easier (from a computational complexity perspective) to determine feasibility than it is to check for EDF*-schedulability. In designing an* EDF*-schedulability test for uniform multiprocessors, therefore, our approach is as follows: for a given uniform multiprocessor platform, we attempt to efficiently identify all those uniform multiprocessor platforms such that any real-time instance feasible upon these platforms is guaranteed to be* EDF*-schedulable upon the platform under consideration.* EDF*-schedulability upon the given platform can then be determined by ascertaining whether the real-time system is feasible upon any of these platforms.*

1. Introduction

While real-time scheduling upon uniprocessor platforms has been extensively studied, it is only recently that *multi*processor real-time scheduling has attracted much attention. In this paper, we consider the scheduling of real-time task systems upon **uniform multiprocessors** — multiprocessor platforms in which all processors are not required to be identical to one another.

The Earliest Deadline First scheduling algorithm (EDF) [9, 3] is a very popular real-time scheduling algorithm that has been extensively studied for both uniprocessors and multiprocessors. While EDF has been shown [9, 3] to be optimal on uniprocessors, it is known [6, 4] that *no* on-line scheduling algorithm can be optimal upon multiprocessors. Despite this lack of optimality, EDF possesses many additional desirable properties that make it an attractive algorithm to use in scheduling multiprocessor real-time systems. Hence, the focus of this paper is *the design of efficient tests for determining whether* EDF *can successfully schedule a given real-time task system to meet all deadlines upon a specified uniform multiprocessor platform.*

A collection I of real-time jobs (in this paper, we will often refer to such collections as real-time *instances*) is said to be *feasible* upon a processor platform π if there is some schedule for I upon π that will result in all deadlines of I being met. For any scheduling algorithm A, I is defined to be A-*schedulable* upon π if A will schedule I upon π to meet all deadlines. Upon uniform multiprocessor platforms, it turns out that it is often far easier (from a computational complexity perspective) to determine feasibility than it is to check for EDF-schedulability (see Example 5 in Section 3 for an example of this phenomenon). In designing an EDF-schedulability test for uniform multiprocessors, therefore, our approach is as follows: for a given uniform multiprocessor platform π, we will derive an efficient algorithm to identify all those uniform multiprocessor platforms π' such that *any real-time instance feasible upon π' is guaranteed to be* EDF*-schedulable upon* π. To do so, we find it convenient to group uniform multiprocessors into *classes*: A uniform multiprocessor π' belongs to the class $\Psi(a, b)$ if its fastest processor has speed a and its total processor speed is b. The set of all such classes is denoted $\wp$. For each multiprocessor π, the **characteristic region** (CR_π) of π is the subset of $\wp$ that satisfies the following property: If $(a, b) \in CR_\pi$ and I is a real-time instance that is known to be feasible on a uniform multiprocessor $\pi' \in \Psi(a, b)$, then I is guaranteed to be EDF-schedulable on π. Moreover, if $(a, b) \notin CR_\pi$ there exists some uniform multiprocessor $\pi' \in \Psi(a, b)$ and some set of jobs I such that I is feasible on π' but not EDF-schedulable on π. Thus, CR_π completely defines the region for which feasibility on a multiprocessor within the region *guarantees* EDF-schedulability on π. To demonstrate that a given instance I is EDF-schedulable upon π, it therefore suffices to identify some platform π' whose class is in CR_π, and upon which I is feasible; as stated above, feasibility-analysis is often an easier problem than EDF-schedulability analysis upon uniform multiprocessors (particularly when one has some choice in determining the precise characteristics of the platform π').

The remainder of this paper is arranged as follows. In Section 2, we introduce the models and definitions that we will use in subsequent sections. In Section 3, we discuss previous

*Supported in part by the National Science Foundation (Grant Nos. CCR-9988327, ITR-0082866, and CCR-0204312).

results that pertain to the work introduced in this paper. In Section 4, we describe an $O(m)$ method for finding points in CR_π. Finally, in Section 5 we provide concluding remarks and outline future work in this area.

2. Models and Definitions

A *real-time instance*, $I = \{j_1, j_2, \ldots, j_n, \ldots\}$, is a (possibly infinite) collection of time-constrained jobs. Each job $j_i \in I$ is completely defined by the 3-tuple (a_i, c_i, d_i), where a_i is the job's arrival time, c_i is its worst-case execution requirement, and d_i its deadline. A *real-time system* is a real-time instance paired with a specific computer processing platform. The platform may be a *uniprocessor*, consisting of one processor, or it may be a *multiprocessor*, consisting of several processors. If the platform is a multiprocessor, the individual processors may all be the same or they may differ from one another. Much of the research concerning real-time scheduling on multiprocessors considers *identical multiprocessors*, in which all processors execute at the same speed. However, other types of multiprocessors, such as *uniform multiprocessors*, have also been considered [10, 11, 7]. On these platforms, each processor has an associated speed: if a job executes on a processor with speed s for one unit of time, the processor will perform s units of work. Notice that identical multiprocessors are a special case of uniform multiprocessors. Therefore, the results found in this paper also apply to identical multiprocessors.

The following notation is used to describe uniform multiprocessors.

Definition 1 *Let $\pi = [s_1, s_2, \ldots, s_m]$ denote an m-processor multiprocessor platform with the i^{th} processor having speed s_i, where $s_i \geq s_{i+1}$ for $i = 1, \ldots, m-1$. The following notation is used to describe properties of π. (When the processor under consideration is unambiguous, the (π) may be removed from the notation.)*

$m(\pi)$**:** *the number of processors in π.*

$s_i(\pi)$**:** *the speed of the i^{th} fastest processor of π.*

$S_i(\pi)$**:** *the cumulative processing power of the i fastest processors of π, $S_i(\pi) \stackrel{\text{def}}{=} \sum_{k=1}^{i} s_k(\pi)$.*

$S(\pi)$**:** *the cumulative processing power of all processors of π, $S(\pi) = S_m(\pi) \stackrel{\text{def}}{=} \sum_{k=1}^{m(\pi)} s_k(\pi)$.*

$\lambda(\pi)$**:** *the "identicalness" of π,*

$$\lambda(\pi) \stackrel{\text{def}}{=} \max_{1 \leq k < m(\pi)} \frac{\sum_{i=k+1}^{m(\pi)} s_i(\pi)}{s_k(\pi)}.$$

Intuitively, the closer π is to being an identical system, the larger the value of λ. When π consists of m identical processors, $\lambda = (m-1)$. The value of λ can be arbitrarily small when the processors have extremely different speeds. For example, $\lambda < \epsilon$ for the m-processor platform where $s_{i+1} \leq \frac{\epsilon}{m} \cdot s_i$ for $i = 1, \ldots, m-1$. The parameter λ is used to analyze the behavior of algorithms that aggressively take advantage of the faster processor speeds.

For the results derived in this paper, some of the uniform multiprocessors do not have to be completely specified — we gain enough information if we know only the fastest speed and the total speed of the multiprocessor. Therefore, we group multiprocessors into the following classes.

Definition 2 **($\Psi(a, b)$)** *The class of uniform multiprocessors with fastest processor speed a and total computing capacity b is denoted $\Psi(a, b)$.*

In addition, we can gain information about the multiprocessor π if we consider multiprocessors that are "less powerful" than π. The following definition (from [1]) provides a structure by which some processors can be seen to be more powerful than others.

Definition 3 (π cleanly dominates π^* ($\pi \stackrel{*}{\trianglerighteq} \pi^*$) [1]) *Let π and π^* be two uniform multiprocessors. Then π* cleanly dominates *π^* if the following three conditions hold: (i) $m(\pi) \geq m(\pi^*)$, (ii) $s_i(\pi) = s_i(\pi^*)$ for all $1 \leq i < m(\pi^*)$, and (iii) $s_{m(\pi^*)}(\pi) \geq s_{m(\pi^*)}(\pi^*)$.*

In many cases, we can deduce certain properties about π by considering the multiprocessors cleanly dominated by π. In particular, in this paper we gain knowledge about EDF-schedulability on π by considering feasibility on a less powerful multiprocessor that π dominates.

Given a real-time system, a scheduling algorithm is used to determine which jobs get scheduled at each point in time. A real-time instance I is called *feasible* on a computing platform π if there exists some scheduling algorithm for which each job $j_i \in I$ receives c_i units of execution in the time interval $[a_i, d_i)$. If an instance I is feasible on a multiprocessor platform π and A is an algorithm that schedules I to meet all its deadlines, then I is called A-*schedulable* on π. One popular scheduling algorithm is *Earliest Deadline First* (EDF), which satisfies the following properties when implemented on an m-processor uniform multiprocessor.

- When more than m jobs are ready for execution, the m jobs with the earliest deadlines must be selected for execution.
- A job can never execute on more than one processor at any point in time.
- A processor idles only if no jobs are waiting to execute.
- Jobs with earlier deadlines are always assigned to faster processors.
- Idling processors are never faster than processors that are executing jobs.

Notice that the final two specifications imply that jobs may migrate form one processor to another. EDF is a well-known and efficient scheduling algorithm. In this paper, we provide a test that can be used to discern if an instance I will meet all its deadlines using EDF on a given multiprocessor π. The central point of this test is consideration of π's characteristic region — if an instance I is feasible on some multiprocessor π' in this region, then I is guaranteed to be EDF-schedulable on π.

Definition 4 (Characteristic region of π (CR_π)) *Let $\wp$ be the two-dimensional space of all classes of multiprocessors. Any point $(a, b) \in \wp$ represents the class of multiprocessors $\Psi(a, b)$ whose fastest processor speed is a and total processor speed is b. The characteristic region associated with any uniform multiprocessor π, denoted CR_π, is the subset of $\wp$ that satisfies the property that for any $(a, b) \in CR_\pi$, any instance I that is feasible on some $\pi' \in \Psi(a, b)$ is EDF-schedulable on π.*

Thus for every (a, b) *not* in the characteristic region of π, there is a real-time instance I that is feasible on some platform $\pi' \in \Psi(a, b)$ and is *not* EDF-schedulable on π.

This Research. This paper introduces an EDF-schedulability test on uniform multiprocessors. EDF-schedulability on a specific m-processor uniform multiprocessor π is determined by considering feasibility on a different multiprocessor π'. Moreover, the exact parameters of π' do not have to be known for this test. In particular, we consider π' as a member of a *class* of uniform multiprocessors.

For every m-processor uniform multiprocessor π, this paper introduces an $O(m)$-time method for dividing the space of classes of uniform multiprocessor, $\wp$, into three distinct regions: classes that are known to be in CR_π, classes that are known to be outside of CR_π, and classes whose membership in CR_π is currently undetermined. In many cases, the third class can be shown to be empty and the region CR_π can be completely determined in $O(m)$ time.

Previously, an $O(m^3)$ method existed to determine if $\Psi(a, b) \in CR_\pi$ for a specific (a, b) [1]. There was no method of determining the entire region CR_π, nor was there any way to determine if a class was *not* in CR_π. Thus, this paper both simplifies and extends existing research.

3. Previous Results

As stated in Section 1, our approach for determining whether a given real-time instance I is EDF-schedulable upon a uniform multiprocessor platform π is to instead determine whether I is *feasible* upon some other uniform multiprocessor platform π'. This is because in many cases feasibility is easier to determine than EDF-schedulability, particularly when we have some freedom to choose the parameters of π'. We illustrate by an example.

Example 5 *Consider a real-time system comprised of n periodic tasks [9] $T_1, T_2, \ldots, T_n$, with each T_i having execution requirement e_i and period p_i. The real-time instance I in this case is the infinite collection of all jobs generated by these n tasks.*

Clearly, I is feasible on the n-processor uniform multiprocessor platform π' with processor speeds $(e_1/p_1), (e_2/p_2), \ldots, (e_n/p_n)$ — simply execute all jobs in I generated by periodic task T_i upon the processor of speed (e_i/p_i). Notice that $\pi' \in \Psi(u_{\max}, U)$ where $u_{\max} = \max\{(e_1/p_1), (e_2/p_2), \ldots, (e_n/p_n)\}$ and $U = \sum_{i=1}^{n}(e_i/p_i)$.

In this paper, we compare the scheduling of an instance I using any algorithm on a given multiprocessor π' to the EDF-scheduling of the same instance on a different, more powerful, multiprocessor π. This technique, called **resource augmentation** was originally considered by Kalyanasundaram and Pruhs [8] and was later explored in more detail by Phillips, Stein, Torng and Wein [12]. Phillips, et al., considered a specific type of resource augmentation in which all processors of an m-processor identical multiprocessor π are sped up by some factor $\alpha > 1$: the resulting platform is denoted $\alpha \times \pi$. They used this concept to develop an EDF-schedulability test for identical multiprocessors.

Theorem 6 ([12]) *Let I be feasible on the identical multiprocessor π. Then I is EDF-schedulable on $(2 - \frac{1}{m}) \times \pi$.*

Thus, if processors are sped up by a factor of $(2 - 1/m)$ then feasibility using by *any* algorithm on the original platform can ensure schedulability using the on-line algorithm EDF on the faster platform. When considering uniform multiprocessors, the processor speeds do not have to all be increased by the same factor. The following theorem relates feasibility on some π' to EDF-schedulability on π — considering only the class of uniform multiprocessors in which π' resides.

Theorem 7 ([5]) *Let π' be a uniform multiprocessor in $\Psi(a, b)$. Assume π is a uniform multiprocessor that satisfies the following property*

$$S(\pi) \geq b + \lambda(\pi) \cdot a. \tag{1}$$

Then any instance I that is feasible on π' is EDF-schedulable on π.

This theorem can be used to determine some of the points in CR_π. By definition, if Inequality 1 is satisfied, then $\Psi(a, b) \in CR_\pi$. Of course, since a is a processor speed, it must be a positive value. Also, since b is the cumulative processing power, b must be at least as large as a. Inequality 1 combined with these restrictions on a and b can be used to define a set R_π that is a subset of CR_π.

Definition 8 (R_π) *Let π be any uniform multiprocessor. The set of points satisfying the conditions of Theorem 7 is denoted R_π. Specifically, $R_\pi \stackrel{\text{def}}{=} \{(a, b) | 0 < a \leq b \leq S(\pi) - \lambda(\pi) \cdot a\}$.*

Example 9 *Let $\pi = [50, 11, 4, 4]$. Then $\lambda = \max\{19/50, 8/11, 4/4\} = 1$ and $S(\pi) = 69$. Therefore, $R_\pi = \{(a, b) | 0 < a \leq b \leq 69 - a\} \subseteq CR_\pi$. Figure 1 illustrates this area. If (a, b) is in the shaded area and I is feasible on some $\pi' \in \Psi(a, b)$ then I is EDF-schedulable on π.*

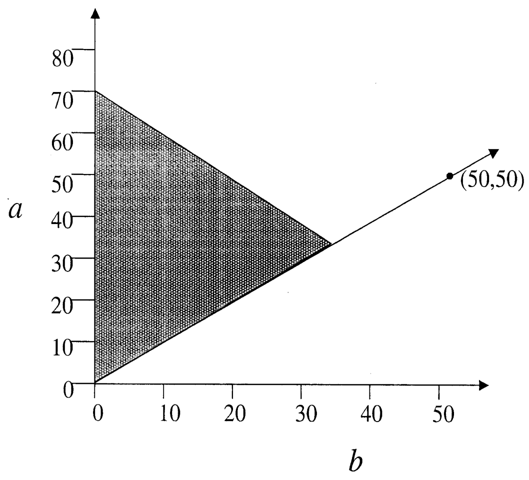

Figure 1. The region $b \leq S(\pi) - \lambda(\pi) \cdot a$ for $\pi = [50, 11, 4, 4]$

Unfortunately, Figure 1 does not illustrate the entire characteristic region of π. Consider, for example, the case where $\pi' = [50] \in \Psi(50, 50)$ (*i.e.*, π' is a uniprocessor). In this case, Inequality 1 does not hold ($50 > 69 - 50$). However, any instance I feasible on π' is clearly EDF-schedulable on π' by the optimality of EDF on uniprocessors. The following theorem allows us to conclude that an instance is EDF-schedulable on π if we know it is EDF-schedulable on π'.

Theorem 10 ([1]) *Let π and π^* be two multiprocessor platforms with $\pi \overset{*}{\trianglerighteq} \pi^*$, and assume I is EDF-schedulable on π^*. Then I is EDF-schedulable on π.*

Applying Theorem 10 to Example 9, any instance schedulable on $\pi' = [50]$ must be EDF-schedulable on $\pi = [50, 11, 4, 4]$ since $\pi \overset{*}{\trianglerighteq} \pi'$. Therefore, $(50, 50) \in CR_\pi$ and the shaded region in Figure 1 must be a *proper* subset of CR_π. This example demonstrates that Theorem 10 can be used to extend the region provided by Inequality 1. In fact, this technique provides us with the following lower bound on the region CR_π.

Definition 11 (cr_π) *Let π be any uniform multiprocessor. The set of points $(a, b) \in R_{\pi^*}$ for some π^* cleanly dominated by π must be in CR_π. We denote this set cr_π:*

$$cr_\pi \overset{\text{def}}{=} \bigcup_{\pi \overset{*}{\trianglerighteq} \pi^*} R_{\pi^*} \subseteq CR_\pi.$$

Since R_{π^*} is defined by a simple linear equation for each π^*, using these regions may provide a very efficient method of finding the elements of CR_π.

4. The Characteristic Region of π

For every m-processor uniform multiprocessor π, methods introduced in this section divide the space $\wp$ of classes of uniform multiprocessors into three regions, all of which can be efficiently identified in $O(m)$ time. The first region contains the classes of multiprocessors that are proven to be in CR_π. If I is feasible on any multiprocessor that falls within this region, then I is EDF-schedulable on π. The second region contains classes of multiprocessors that are definitely not in CR_π. For every (a, b) in this region, we find a real-time instance I that is feasible on some uniform multiprocessor $\pi' \in \Psi(a, b)$ but is not EDF-schedulable on π. Finally, we have not yet determined whether multiprocessors in the third region are in CR_π or not, though we conjecture that all points in this region are *not* in CR_π. There are many multiprocessors for which this third region is empty. If this third region is empty for a given multiprocessor π then the critical region is fully determined by methods introduced in this paper. In particular, we will show that the critical region can be completely determined for every *identical* multiprocessor.

The previous sections provide insight into finding points in the characteristic region of a given uniform multiprocessor π. Namely, if $(a, b) \in R_{\pi^*}$ for some π^* cleanly dominated by π, then $(a, b) \in CR_\pi$. This section is divided into three parts. First, we will introduce a simple $O(m)$ method for determining the set cr_π of points that are guaranteed to be in CR_π. Then we will introduce a simple $O(m)$ method for determining points that are outside CR_π. Finally, we clearly indicate which points (a, b) are currently not covered by these tests — *i.e.*, the points that are neither proven to be in cr_π nor proven to be outside of CR_π. Figure 2 shows all three regions for the uniform multiprocessor $\pi = [50, 11, 4, 4]$.

4.1. Finding a lower bound for CR_π

While this section is primarily concerned with finding points *inside* CR_π, we first eliminate points that clearly cannot be in CR_π.

Lemma 12 *Let π be any uniform multiprocessor. Then $CR_\pi \subseteq \{(a, b) | 0 < a \leq b \wedge a \leq s_1(\pi)\}$.*

Proof. Since a is a processor speed, it must be the case that $a > 0$. Also, since b is the cumulative speed, we must also have $b \geq a$. We show that $a \leq s_1(\pi)$ by construction.

Let $\pi' \in \Psi(a_0, b_0)$ for some $a_0 > s_1(\pi)$. The instance consisting of one job $I = \{(0, a_0, 1)\}$ is feasible on π' if the job is executed on its fastest processor. However, I is not feasible on π since the job will miss its deadline even if it is executed on π's fastest processor. Therefore, (a_0, b_0) cannot be in CR_π for any $a_0 > s_1(\pi)$. □

We now determine which points are in cr_π. Recall that cr_π is the union of the regions R_{π^*} for some π^* cleanly dominated by π. Lemma 14 begins by showing the following set of points must be above or on the border of R_{π^*} for any $\pi \overset{*}{\trianglerighteq} \pi^*$.

Definition 13 (A_π) *Let $\pi = [s_1, s_2, \ldots, s_m]$ be any uniform*

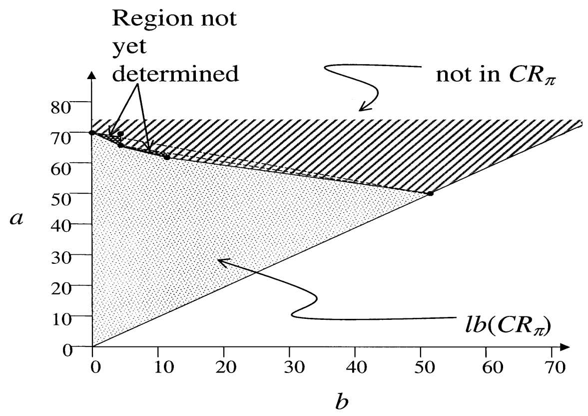

Figure 2. The characteristic region associated with $\pi = [50, 11, 4, 4]$

multiprocessor. Then

$$A_\pi \stackrel{\text{def}}{=} \{(s_1, S_1), (s_2, S_2), \ldots, (s_m, S_m), (s_{m+1}, S_{m+1})\},$$

where $(s_{m+1}, S_{m+1}) \stackrel{\text{def}}{=} (0, S(\pi))$.

Lemma 14 *Let π and π^* be any uniform multiprocessors such that $\pi \stackrel{*}{\trianglerighteq} \pi^*$. Then for every $(s_i(\pi), S_i(\pi)) \in A_\pi$, $S_i(\pi) \geq S(\pi^*) - \lambda(\pi^*) \cdot s_i(\pi)$.*

Proof. If $S(\pi^*) < s_1(\pi)$ then π^* is a uniprocessor and the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$ becomes $b = S(\pi^*)$. In this case, $S_i(\pi) \geq s_1(\pi) \geq S(\pi^*)$ for every $1 \leq i \leq m+1$, so all points in A_π are clearly above the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$.

Now assume $S(\pi^*) \geq s_1(\pi)$. Let β be the largest index such that $S_\beta(\pi) \leq S(\pi^*)$. By definition of β, $S_i(\pi) > S(\pi^*)$ for every $\beta < i \leq m(\pi) + 1$. Furthermore, since $\lambda(\pi^*) \geq 0$, we know $S(\pi^*) \geq S(\pi^*) - \lambda(\pi^*) \cdot s_i(\pi)$. Therefore, $(s_i(\pi), S_i(\pi))$ is above the line for every $\beta < i \leq m(\pi) + 1$.

It remains to show that the property holds when $1 \leq i \leq \beta$. Since $\pi \stackrel{*}{\trianglerighteq} \pi^*$, we know that $s_i(\pi) = s_i(\pi^*)$ for $i \leq i \leq \beta$. Therefore $(s_i(\pi), S_i(\pi)) = (s_i(\pi^*), S_i(\pi^*))$ for $1 \leq i \leq \beta$. Assume there is some point $(s_k(\pi^*), S_k(\pi^*))$ below the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$. Then

$$\begin{aligned} S_k(\pi^*) &< S(\pi^*) - \lambda(\pi^*) \cdot s_k(\pi^*) \\ \Rightarrow \lambda(\pi^*) \cdot s_k(\pi^*) &< S(\pi^*) - S_k(\pi^*) \\ \Rightarrow \lambda(\pi^*) &< \frac{\sum_{i=k+1}^{m(\pi^*)} s_i(\pi^*)}{s_k(\pi^*)}. \end{aligned}$$

The final inequality violates the definition of $\lambda(\pi^*)$. Therefore, $(s_i(\pi^*), S_i(\pi^*))$ is above the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$ for every $1 \leq i \leq m(\pi^*)$. In particular, $(s_i(\pi), S_k(\pi))$ is above the line for every $1 \leq i \leq \beta$. □

Lemma 14 states that the points in A_π are either above or on the upper border of R_{π^*} for any $\pi \stackrel{*}{\trianglerighteq} \pi^*$. Therefore, A_π may be useful in understanding the shape of cr_π. In fact, below we define a specific subset, H_π, of A_π. This subset is the lower portion of the convex hull of the points of A_π. The piecewise linear function $L_\pi(a)$ formed by connecting consecutive points in H_π can be used to fully define cr_π.

Definition 15 **($H_\pi, L_\pi(a)$)** *Let $\pi = [s_1, s_2, \ldots, s_m]$ be any uniform multiprocessor. Define the set $H_\pi = \{(s_{h_1}, S_{h_1}), (s_{h_2}, S_{h_2}), \ldots, (s_{h_{m'}}, S_{h_{m'}})\} \subseteq A_\pi$ recursively as follows.*

- $s_{h_1} = s_1(\pi)$
- *If $s_{h_i} \neq 0$, then $s_{h_{i+1}}$ is the fastest processor speed less than s_{h_i} such that all points in A_π are either on or above the line defined by (s_{h_i}, S_{h_i}) and $(s_{h_{i+1}}, S_{h_{i+1}})$. Specifically,*

$$s_{h_{i+1}} = \max \Big\{ s_k(\pi) < s_{h_i} \mid$$
$$\forall 1 \leq \ell \leq m+1, S_\ell(\pi) \geq S_k - \frac{S_k - S_{h_i}}{s_{h_i} - s_k}(s_\ell - s_k) \Big\}.$$

- *If $s_{h_i} = 0$ then $i = m'$ (thus, H_π is completely defined).*

Finally, define the function $L_\pi(a)$ with domain $0 < a \leq s_1(\pi)$ to be the piecewise linear function found by joining consecutive points in H_π.

$$L_\pi(a) = S_{h_{i+1}} - \frac{S_{h_{i+1}} - S_{h_i}}{s_{h_i} - s_{h_{i+1}}}(a - s_{h_{i+1}})$$

$$\text{for } s_{h_{i+1}} \leq a \leq s_{h_i} \text{ for } 1 \leq i \leq m'.$$

Lemma 17 states that all points on or below $L_\pi(a)$ are in R_{π^*} for some $\pi \stackrel{*}{\trianglerighteq} \pi$. The proof uses the following lemma which proves specific properties about the upper bound of R_π.

Lemma 16 *Let $\pi = [s_1, s_2, \ldots, s_m]$ be any multiprocessor and let* L *be a line on the (a, b) plane with the property that all points in A_π are on or above* L*. If $(0, S(\pi))$ is on* L *and there is some $1 \leq k \leq m$ such that (s_k, S_k) is also on the line* L *then the equation for the line* L *is $b = S(\pi) - \lambda(\pi) \cdot a$. Moreover, $\lambda(\pi) = (\sum_{i=k+1}^{k} s_i)/s_k$.*

Proof. The slope of L is $(S(\pi) - S_k)/(0 - s_k) = -(\sum_{i=k+1}^{m} s_i)/s_k$. Therefore the equation of this line is $b = S(\pi) - \frac{\sum_{i=k+1}^{m} s_i}{s_k} \cdot a$. Since all points in A_π are on or above L, the following inequality must hold for $1 \leq \ell \leq m$.

$$\begin{aligned} S_\ell &\geq S(\pi) - \frac{\sum_{i=k+1}^{m} s_i}{s_k} \cdot s_\ell \\ \Rightarrow \frac{\sum_{i=k+1}^{m} s_i}{s_k} \cdot s_\ell &\geq S(\pi) - S_\ell \\ \Rightarrow \frac{\sum_{i=k+1}^{m} s_i}{s_k} &\geq \frac{\sum_{i=\ell+1}^{m} s_i}{s_\ell}. \end{aligned}$$

Therefore, $\frac{\sum_{i=k+1}^{m} s_i}{s_k} = \max_{1 \leq \ell \leq m} \left\{ \frac{\sum_{i=\ell+1}^{m} s_i}{s_\ell} \right\} = \lambda(\pi)$. □

Lemma 17 *Let π be any uniform multiprocessor and let the lower portion of the convex hull of A_π be $H_\pi = \{(s_{h_1}, S_{h_1}), (s_{h_2}, S_{h_2}), \ldots, (s_{h_{m'}}, S_{h_{m'}})\}$. Then for any $1 \leq i < m'$, there exists a multiprocessor π^* such that $\pi \overset{*}{\trianglerighteq} \pi^*$, and the points (s_{h_i}, S_{h_i}) and $(s_{h_{i+1}}, S_{h_{i+1}})$ are both on the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$.*

Proof. Let L be the line defined by the points (s_{h_i}, S_{h_i}) and $(s_{h_{i+1}}, S_{h_{i+1}})$. The equation of this line is $b = S_{h_{i+1}} - \frac{S_{h_{i+1}} - S_{h_i}}{s_{h_i} - s_{h_{i+1}}}(a - s_{h_{i+1}})$. Let α be the b-intercept of L — *i.e.*,

$$\alpha = S_{h_{i+1}} + s_{h_{i+1}} \cdot \frac{S_{h_{i+1}} - S_{h_i}}{s_{h_i} - s_{h_{i+1}}}.$$

Since $S_{h_{i+1}} \geq S_{h_i}$ and $s_{h_i} > s_{h_{i+1}}$, we see $\alpha \geq S_{h_{i+1}} \geq s_1(\pi)$. Furthermore, the point $(0, S(\pi))$ is in A_π and therefore must be on or above L. Therefore, $s_1(\pi) \leq \alpha \leq S(\pi)$ and there exists a uniform multiprocessor cleanly dominated by π with total capacity α — let π^* be this multiprocessor.

Note that $\alpha \geq S_{h_{i+1}} > S_{h_i}$, therefore (s_{h_i}, S_{h_i}) and $(s_{h_{i+1}}, S_{h_{i+1}})$ are both in A_{π^*}. Thus, L contains the points $(0, S(\pi^*))$ and $(s_k(\pi^*), S_k(\pi^*))$ for some $1 \leq k \leq m(\pi^*)$. If all points in A_{π^*} can be proven to be on or above L then, by Lemma 16, L *must* be the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$.

Since $\pi \overset{*}{\trianglerighteq} \pi^*$, at most one point in A_{π^*} is not in A_π — namely $(s_{m(\pi^*)}(\pi^*), S(\pi^*))$. Therefore, since all points in A_π are known to be on or above L, this is the only point in A_{π^*} that is not known to satisfy this property. We can easily see that this point is above L because the b-intercept of L is $(0, S(\pi^*))$ and its slope is negative. Therefore, L must be the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$. □

By Lemma 17 we see that all points on or below $L_\pi(a)$ are in some region R_{π^*} for some π^* cleanly dominated by π. It remains to be shown that no other points can be in any R_{π^*}.

Theorem 18 *Let $\pi = [s_1, s_2, \ldots, s_m]$ be any uniform multiprocessor and let the lower portion of the convex hull of A_π be $H_\pi = \{(s_{h_1}, S_{h_1}), (s_{h_2}, S_{h_2}), \ldots, (s_{h_{m'}}, S_{h_{m'}})\}$. Define the function $L_\pi(a)$ to be the piecewise linear function connecting the points of H_π. Then*

$$cr_\pi = \{(a, b) | 0 < a \leq s_1 \wedge a \leq b \leq L_\pi(a)\}.$$

Proof. Lemma 17 demonstrates that $\{(a, b) | 0 < a \leq s_1 \wedge a \leq b \leq L_\pi(a)\} \subseteq cr_\pi$. Furthermore, Lemma 12 shows that the inequalities $0 < a \leq s_1$ and $a \leq b$ must hold for any point in CR_π. It remains to show that $b \leq L_\pi(a)$ for any $(a, b) \in cr_\pi$. We show this by contradiction.

Let π^* be a multiprocessor cleanly dominated by π. Let L denote the line $b = S(\pi^*) - \lambda(\pi^*) \cdot a$. Assume there exists a point (a_0, b_0) and such that $b_0 > L_\pi(a_0)$ and (a_0, b_0) is on L. Since $S(\pi^*) \leq S(\pi)$, the point $(0, S(\pi^*))$ must be on or below $L_\pi(a)$. Thus L and $L_\pi(a)$ must have at least on point of intersection between 0 and a_0. Furthermore, this must be a single point of intersection, because no line containing a segment of $L_\pi(a)$ can have any point above $L_\pi(a)$ (by the construction of $L_\pi(a)$). Let (a^*, b^*) be the first point of intersection between the two lines. Note that for $a > a^*$, $L_\pi(a)$ is below L. Since 2 straight non-collinear lines can have only one point of intersection, $L_\pi(a)$ must remain below L until its slope changes. But the slope changes only at points in $H_\pi \subseteq A_\pi$. This implies that there is a point in A_π below L. This contradicts Lemma 14. Therefore, there can be no point in cr_π above $L_\pi(a)$ □

Theorem 18 provides a simple method for determining a lower bound for the characteristic region of π — namely find the convex hull of the points in A_π. The set A_π can be found in $O(m)$ time. Notice that the points of A_π are pre-sorted because the processor speeds are assumed to be indexed in non-increasing order. Therefore, H_π can be found in $O(m)$ time using a convex hull algorithm such as Graham's scan (for example, see pages 889-905 of [2]). Finally, the function $L_\pi(a)$ can be found in $O(|H_\pi|)$ time. Therefore, the computational complexity of finding cr_π is $O(m)$.

4.2. Identifying points outside CR_π

While we have shown that $cr_\pi \subseteq CR_\pi$, we have not yet shown that any points *outside* cr_π are definitely not in CR_π. The points shown to be in CR_π are all on or below the line $L_\pi(a)$. In this section, we show that for any $1 < k \leq m + 1$, all points above the line between (s_1, s_1) and (s_k, S_k) are not in CR_π. We do this by finding a specific real-time instance I that is feasible on a uniform multiprocessor π' but is not EDF-schedulable on π, where π' is in a class of multiprocessors that is above the given line.

Theorem 19 *Let $\pi = [s_1, s_2, \ldots, s_m]$ be any uniform multiprocessor and let $s_k < a_0 \leq s_1$ for some $1 < k \leq m + 1$. If (a_0, b_0) is above the line between (s_1, s_1) and (s_k, S_k) then $(a_0, b_0) \notin CR_\pi$.*

Proof. This theorem is proved in Appendix A. For each point (a_0, b_0) above the line, the proof illustrates that for a particular instance I and multiprocessor $\pi' \in \Psi(a_0, b_0)$, I that is feasible on π' but is not EDF-schedulable on π. Therefore, by definition $(a_0, b_0) \notin CR_\pi$. □

Each line can be found in constant time and there are at most m lines, therefore the region that is definitely outside CR_π can also be found in $O(m)$ time.

4.3. Identifying points whose membership in CR_π has not been determined

We have seen that all points on or below the piecewise linear function $L_\pi(a)$ are in CR_π and that all points above any line segment connecting (s_1, s_1) to (s_k, S_k) are not in CR_π. Therefore, the remaining points are those that are above $L_\pi(a)$ and below every line segment between (s_1, s_1) and (s_k, S_k). For simplicity, we will call this region M. Figure 2 illustrates this region for the example multiprocessor $\pi = [50, 11, 4, 4]$. We shall see that this region is always comprised of a group of triangles.

Use H_π to partition $0 < a \leq s_1$ and consider the points in M one partition at a time, beginning with $s_{h_2} \leq a \leq s_{h_1}$. By

Theorem 18, we know that every point on or below the line segment between (s_{h_1}, S_{h_1}) and (s_{h_2}, S_{h_2}) is in CR_π. Furthermore, by Theorem 19, every point above this line segment is proved not to be in CR_π. Therefore, all points $(a, b) \in M$ have the property that $a < s_{h_2}$.

Now consider $s_{h_i} \leq a \leq s_{h_{i-1}}$ for some $3 \leq i \leq m'$. By Theorem 18, we know that every point below the line segment between $(s_{h_{i-1}}, S_{h_{i-1}})$ and (s_{h_i}, S_{h_i}) is in CR_π. Also, by Theorem 19, every point above the line segment between $(s_{h_{i-1}}, S_{h_{i-1}})$ and (s_1, s_1) is proved not to be in CR_π. Furthermore, this is the lowest segment that borders points not in CR_π because H_π is the lower edge of the convex hull. Therefore, if we let $g_k(a)$ be the line between (s_1, s_1) and (s_k, S_k), the region M is the set of points above the line segment between $(s_{h_{i-1}}, S_{h_{i-1}})$ and (s_{h_i}, S_{h_i}) and below or on $g_{h_i}(a)$ for $3 \leq i \leq m'$. For each i, the region between these two lines is the triangle defined by the three points $(s_{h_{i-1}}, S_{h_{i-1}})$, (s_{h_i}, S_{h_i}), and $(s_{h_{i-1}}, g_{h_i}(s_{h_{i-1}}))$.

Notice that M is empty when $|H_\pi| = 2$ — in this case $CR_\pi = cr_\pi$. Whenever $\lambda(\pi) = \sum_{i=2}^{m} s_i/s_1$, $CR_\pi = cr_\pi$ and hence CR_π is fully defined. For example, the characteristic region of $\pi = [2, 2, 2, 1]$ (with $\lambda(\pi) = 5/2$) is completely identified. Similarly, if π is an identical multiprocessor, then CR_π is completely identified.

5. Conclusion

EDF is a very popular and efficient scheduling algorithm. The results presented in this paper allow EDF to be used on a new type of platform with confidence that all deadlines will be met. EDF-schedulability on a processing platform π is determined by considering feasibility on a less powerful platform π'. In many cases, feasibility is easier to determine than schedulability using a specific scheduling algorithm. Therefore, this paper takes the difficult question of whether EDF can be used to schedule a real-time instance I on π and reduces it to the relatively easier question of whether *any* scheduling algorithm can be used to schedule I on a different (less powerful) platform π'.

A central concept of this paper is the development of a characteristic region, CR_π. The region contains *classes* of uniform multiprocessors — any real-time instance feasible on some multiprocessor within one of these classes will be EDF-schedulable on π. In $O(m)$ time, one can divide all classes of uniform multiprocessors into three regions: those that are definitely in CR_π, those that are definitely not in CR_π, and those whose membership in CR_π is currently not determined. Moreover, in many cases the third region is empty. In the future, we hope to fully determine the shape of CR_π for *all* uniform multiprocessors. We conjecture that CR_π contains only those points that we have already proven to be in CR_π (*i.e.*, we believe $CR_\pi = cr_\pi$).

References

[1] BARUAH, S. Robustness results concerning EDF scheduling upon uniform multiprocessors. In *Proceedings of the EuroMicro Conference on Real-Time Systems* (Vienna, Austria, June 2002), IEEE Computer Society Press, pp. 95–102.

[2] CORMEN, T., LEISERSON, C., AND RIVEST, R. *Introduction to Algorithms*. Prentice Hall, 1998.

[3] DERTOUZOS, M. Control robotics : the procedural control of physical processors. In *Proceedings of the IFIP Congress* (1974), pp. 807–813.

[4] DERTOUZOS, M., AND MOK, A. K. Multiprocessor scheduling in a hard real-time environment. *IEEE Transactions on Software Engineering 15*, 12 (1989), 1497–1506.

[5] FUNK, S., GOOSSENS, J., AND BARUAH, S. On-line scheduling on uniform multiprocessors. In *Proceedings of the IEEE Real-Time Systems Symposium* (December 2001), IEEE Computer Society Press, pp. 183–192.

[6] HONG, K., AND LEUNG, J. On-line scheduling of real-time tasks. In *Proceedings of the Real-Time Systems Symposium* (Huntsville, Alabama, December 1988), IEEE, pp. 244–250.

[7] HORVATH, E. C., LAM, S., AND SETHI, R. A level algorithm for preemptive scheduling. *Journal of the ACM 24*, 1 (1977), 32–43.

[8] KALYANASUNDARAM, B., AND PRUHS, K. Speed is as powerful as clairvoyance. In *36th Annual Symposium on Foundations of Computer Science (FOCS'95)* (Los Alamitos, Oct. 1995), IEEE Computer Society Press, pp. 214–223.

[9] LIU, C., AND LAYLAND, J. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM 20*, 1 (1973), 46–61.

[10] LIU, J. W. S., AND LIU, C. L. Bounds on scheduling algorithms for heterogeneous computing platforms. In *Proceedings of the IFIP Congress* (Stockholm, Sweden, 1974), vol. 30, North-Holland Publishing Company, pp. 483–485.

[11] LIU, J. W. S., AND YANG, A. T. Optimal scheduling of independent tasks on heterogeneous computing systems. In *Proceedings ACM National Conference* (N. Y., 1974), vol. 1, ACM, pp. 38–45.

[12] PHILLIPS, C. A., STEIN, C., TORNG, E., AND WEIN, J. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing* (El Paso, Texas, 4–6 May 1997), pp. 140–149.

A. Proof of Theorem 19

We include the proof of Theorem 19 in the appendix for completeness.

Theorem 19 Let $\pi = [s_1, s_2, \ldots, s_m]$ be any uniform multiprocessor and let $s_k < a_0 \leq s_1$ for some $1 < k \leq m+1$ (where $(s_{m+1}, S_{m+1}) = (0, S_m)$). If (a_0, b_0) is above the line between (s_1, s_1) and (s_k, S_k) then $(a_0, b_0) \notin CR_\pi$.

Proof. The equation of the line between (s_1, s_1) and (s_k, S_k) is

$$f(a) = s_1 + \frac{S_k - s_1}{s_1 - s_k} \cdot (s_1 - a).$$

Therefore, to prove the theorem, it suffices to show that for every point (a_0, b_0) where $b_0 > f(a_0)$, there exists an instance I that is feasible on some multiprocessor $\pi' \in \Psi(a_0, b_0)$ but is not EDF-schedulable on π.

Let ρ be any positive integer such that

$$a_0 \geq \frac{s_1(b_0 - a_0)}{\rho S_{k-1}}.$$

Let π' be the uniform multiprocessor consisting of $((\rho - 1)m + k)$ processors with the following speeds:

1 processor with speed a_0,

1 processor with speed $\frac{s_i(b_0-a_0)}{\rho S_{k-1}}$ for each $1 \leq i \leq k-1$,

$(\rho - 1)$ processors with speed $\frac{s_i(b_0-a_0)}{\rho S_m}$ for each $1 \leq i \leq m$.

By construction, $\pi' \in \Psi(a_0, b_0)$.

Choose any positive numbers ϵ and $\delta_{h,i}$, where $1 \leq h \leq \rho$ and $1 \leq i \leq m$, such that

$\epsilon < (b_0 - f(a_0)) \frac{s_1 - s_k}{\rho S_{k-1}(s_1 - a_0)}$,

$\delta_{h,i} < \delta_{h,i+1}$ for $1 \leq h \leq \rho$ and $1 \leq i < m$,

$\delta_{h,m} < \delta_{h+1,1}$ for $1 \leq h < \rho$, and

$\delta_{\rho,k-1} < \epsilon$,

and let

$$\begin{aligned} I = \Big\{ j_{h,i} = \left(0, \frac{s_i(b_0 - a_0)}{\rho S_m}, 1 + \delta_{h,i}\right) \Big| \\ 1 \leq h < \rho, 1 \leq i \leq m \Big\} \cup \\ \Big\{ j_{\rho,i} = \left(0, \frac{s_i(b_0 - a_0)}{\rho S_{k-1}}, 1 + \delta_{\rho,i}\right) \Big| 1 \leq i < k \Big\} \cup \\ \Big\{ j_{\rho,k} = (0, a_0(1+\epsilon), 1+\epsilon) \Big\}. \end{aligned}$$

(Note that if $\rho = 1$ then π' consists of k processors and I consists of k jobs.)

To show that I is feasible on π' consider the schedule where $j_{\rho,k}$ is scheduled on the processor with speed a_0 and the remaining jobs are scheduled on the processors with speed equal to their execution requirement. Thus, $j_{\rho,k}$ completes at $t = (1+\epsilon)$ and the remaining jobs complete at $t = 1$.

Now consider the EDF schedule of I on π. The jobs in I are listed in increasing order of deadlines. Therefore, assuming $\rho > 1$, jobs $j_{1,1}, j_{1,2}, \ldots, j_{1,m}$ will be scheduled on processor with speeds $s_1, s_2, \ldots, s_m$, respectively. Furthermore, all these jobs will complete simultaneously at $t = (b_0 - a_0)/(\rho S_m)$, at which point the next m jobs will be scheduled. This will repeat $\rho - 1$ times with the final group of m jobs completing at $t = (\rho-1)(b_0 - a_0)/(\rho S_m)$. (Note if $\rho = 1$, this expression evaluates to 0 and will not affect the subsequent calculations.)

Once the $(\rho - 1)m$ jobs with the earliest deadlines have executed, the final k jobs are scheduled in a similar manner — job $j_{\rho,i}$ will be scheduled on the processor with speed s_i for $1 \leq i \leq k$. Jobs $j_{\rho,1}, j_{\rho,2}, \ldots, j_{\rho,k-1}$ will all complete simultaneously after $(b_0 - a_0)/(\rho S_{k-1})$ time units at which point $j_{\rho,k}$ will migrate to the fastest processor. Assuming $j_{\rho,k}$ meets its deadline, we have the following inequality:

$$\frac{(\rho-1)(b_0 - a_0)}{\rho S_m} + \frac{(b_0 - a_0)}{\rho S_{k-1}} + \frac{a_0(1+\epsilon) - \frac{b_0 - a_0}{\rho S_{k-1}} s_k}{s_1} \leq 1 + \epsilon.$$

Subtracting the leftmost term from both sides and multiplying by ρS_{k-1} gives

$$\begin{aligned} b_0 - a_0 + \frac{\rho S_{k-1}}{s_1} a_0(1+\epsilon) - \frac{s_k}{s_1}(b_0 - a_0) \leq \\ \rho S_{k-1}(1+\epsilon) - \frac{(\rho-1) S_{k-1}(b_0 - a_0)}{S_m} \end{aligned}$$

$$\begin{aligned} \Rightarrow b_0 \left(1 - \frac{s_k}{s_1}\right) \leq \\ \rho S_{k-1}(1+\epsilon) - \frac{\rho S_{k-1}}{s_1} a_0 (1+\epsilon) + \\ a_0 \left(1 - \frac{s_k}{s_1}\right) - \frac{(\rho-1) S_{k-1}(b_0 - a_0)}{S_m} \end{aligned}$$

Multiplying by $s_1/(s_1 - s_k)$ gives

$$\begin{aligned} b_0 &\leq \rho S_{k-1} \frac{s_1 - a_0}{s_1 - s_k} + a_0 + \epsilon \rho S_{k-1} \frac{s_1 - a_0}{s_1 - s_k} - \\ &\qquad \frac{(\rho-1) s_1 S_{k-1}(b_0 - a_0)}{S_m(s_1 - s_k)} \\ &= S_{k-1} \frac{s_1}{s_1 - s_k} + a_0 \left(1 - \frac{S_{k-1}}{s_1 - s_k}\right) + \epsilon \rho S_{k-1} \frac{s_1 - a_0}{s_1 - s_k} - \\ &\qquad \frac{(\rho-1) S_{k-1}}{s_1 - s_k} \left(s_1 - a_0 + \frac{s_1(b_0 - a_0)}{S_m}\right) \\ &\leq s_1 + s_1 \left(\frac{S_{k-1}}{s_1 - s_k} - 1\right) + a_0 \left(1 - \frac{S_{k-1}}{s_1 - s_k}\right) + \\ &\qquad \epsilon \rho S_{k-1} \frac{s_1 - a_0}{s_1 - s_k} \\ &= s_1 + \frac{S_k - s_1}{s_1 - s_k}(s_1 - a_0) + \epsilon \rho S_{k-1} \frac{s_1 - a_0}{s_1 - s_k} \end{aligned}$$

Note that the first two terms of the right hand side of the final inequality equal $f(a_0)$. Therefore, we have

$$(b_0 - f(a_0)) \frac{s_1 - s_k}{\rho S_{k-1}(s_1 - a_0)} \leq \epsilon.$$

This contradicts our choice of ϵ. Therefore, $j_{\rho,k}$ cannot meet its deadline. $\square$