2009 21st Euromicro Conference on Real-Time Systems

Competitive Analysis of Energy-Constrained Real-Time Scheduling[*]

Vinay Devadas Fei Li Hakan Aydin
Department of Computer Science
George Mason University
Fairfax, VA 22030
{vdevadas, lifei, aydin }@cs.gmu.edu

Abstract

In this paper, we undertake the competitive analysis of the online real-time scheduling problems under a given hard energy constraint. Specifically, we derive worst-case performance bounds that apply to any online algorithm, when compared to an optimal algorithm that has the knowledge of the input sequence in advance. First, by focusing on uniform value-density settings, we prove that no online algorithm can achieve a competitive factor greater than $1 - \frac{e_{max}}{E}$, where e_{max} is the upper bound on the size of any job and E is the available energy budget. Then we propose a variant of *EDF* algorithm, *EC-EDF*, that is able to achieve this upper bound. We show that a priori information about the largest job size in the *actual input sequence* makes possible the design of a *semi-online algorithm* $EC\text{-}EDF^*$ which achieves a constant competitive factor of 0.5. This turns out to be the best achievable competitive factor in these settings. We also extend our analysis to other settings, including those with non-uniform value densities and Dynamic Voltage Scaling capability.

1 Introduction

An algorithm is said to be *online* if it must make its decisions at run-time, without having any information about the future input. Design and analysis of online algorithms is a well-established field with direct applications in a wide range of areas such as load balancing, scheduling, circuit design, server performance evaluation, memory hierarchy design, search, portfolio selection, and revenue management [13]. In online settings, the performance of an algorithm is often assessed by comparing it to that of an *optimal* and *clairvoyant* algorithm that knows the entire input *in advance*. This framework, known as *competitive analysis*, is considered as a standard analysis and evaluation tool in Computer Science. The reader is referred to excellent surveys such as [13, 30] for an overview of existing techniques and results.

For *underloaded* real-time systems, where a feasible schedule for the workload is *known* to exist, the preemptive Earliest Deadline First (EDF) algorithm is optimal in the hard real-time sense, meaning that EDF is guaranteed to meet all the deadlines even when it processes and schedules jobs *as* they arrive without any knowledge of future release times [18]. This remarkable feature does not extend to multiprocessor environments, for which it is well-known that no online algorithm can guarantee to generate a feasible schedule if future job release times are not known [17].

A real-time system is said to be *overloaded*, if there does not exist a schedule where all jobs meet their deadlines. In these settings where deadline misses are unavoidable (even for a clairvoyant algorithm), the goal is typically to maximize a soft real-time performance metric. A common approach in *firm* real-time systems is to associate a *value* with each job to quantify its contribution to the overall system performance [8, 9, 14]. The value of the job is added to the overall performance metric (*cumulative,* or, *total* system value) *if and only if* it meets its deadline – no value is accrued for partial executions that are not completed before the task deadline.

An online scheduling algorithm is said to have a *competitive factor* r, $0 \leq r \leq 1$, *if and only if* it is guaranteed to achieve a cumulative value at least r times the cumulative value achievable by a clairvoyant algorithm on *any finite input job sequence* [8, 14, 23]. An online algorithm is said to be *competitive*, if it has a constant competitive factor strictly greater than zero. In general, the higher the competitive ratio, the better performance guarantees provided by an online algorithm. As such, the competitive analysis technique aims to establish performance bounds that hold even under *worst-case* scenarios for online algorithms, when compared to an optimal clairvoyant algorithm [13, 30].

Another technique in competitive analysis is to explore the impact of giving some (but still, limited) information to the online algorithm about the actual input sequence (actual job set) in an effort to improve its competitiveness. Such algorithms are known as *semi-online* algorithms [30]. For example, a priori information about the largest job size that will appear in the *actual* input sequence typically makes possible design of semi-online algorithms with improved competitive ratio [30].

A significant body of research has been devoted to the analysis of *online* scheduling algorithms for overloaded real-time systems, where the goal is to maximize the total system value [8, 9, 11, 23]. In a seminal paper, Baruah et al. showed that no online algorithm can achieve a competitive factor greater than 0.25 [8], in an overloaded real-time system. This result holds for task systems with *uniform density* settings, where the value

[*]Supported, in part, by US National Science Foundation grants CNS-072047 and CNS-546244 (CAREER Award)

of a job is proportional to its execution time. For more general, *non-uniform value density* settings where different tasks may contribute different values per execution time, the bound is much smaller. Consider a firm real-time task system where the job J_i accrues k_i units of value per execution time (*value density*), if it completes by the deadline. Then, no online algorithm can achieve a competitive factor greater than $\frac{1}{(1+\sqrt{k})^2}$, where $k = \frac{max(k_i)}{min(k_i)}$ is the *importance ratio* obtained through the largest and smallest value densities in the task set [8].

Remarkably, Koren and Shasha provided an optimal online algorithm D^{over} that achieves this upper bound [23]. The same authors also considered extensions to multiprocessor settings [24]. Several other studies addressed competitive on - line real-time scheduling for imprecise computation tasks [12], tasks with bounded slack factors [11], and tasks with given stretch metrics [27] among others.

Recently, energy management has moved to the forefront of research in real-time systems. This is primarily due to the emergence of devices that rely on battery power, which is fairly limited. Literally hundreds of research papers were published, each extending the real-time scheduling results to energy-aware settings for various task/system models. These studies can be broadly classified in two categories. The first line targets minimizing the total energy consumption while meeting the timing constraints [3, 5, 29]. The second line addresses the settings where the system has to operate within a *given and fixed energy budget* and explores ways to maximize the performance of an underloaded real-time system, when the energy is not sufficient to meet all the deadlines [1, 2, 16, 31, 32, 33]. In the latter formulation, energy is effectively *the* hard constraint and the system is said to be *energy-constrained* [1, 2].

Contributions of this paper. The primary objective of this paper is to undertake a preliminary competitive analysis of *energy-constrained online* real-time scheduling problem. As mentioned above, most of the online real-time scheduling studies focus on *overloaded* settings, where *time* is the main (and only) limiting factor. While recognizing the fundamental importance of these pioneering studies, we posit that there is a need to consider also the *online* real-time scheduling frameworks with *hard energy constraint* (where energy is the main scarce resource). As a first step, we analyze the energy-constrained settings for *underloaded* uni-processor systems (i.e. where *energy* is the main *and only* limiting factor), derive inherent performance bounds, and prove the existence of online real-time scheduling algorithms that achieve these bounds for a few fundamental task models. Specifically:

- By extending the conventional uniform value density online real-time system settings to energy-constrained envi - ronments, we show that no online algorithm can achieve a competitive factor better than $1 - \frac{e_{max}}{E}$, where e_{max} is the upper bound on the size of any job and E is the available energy budget of the system (Section 3).
- We develop a variant of EDF algorithm, called *EC-EDF*, that is provably optimal in online energy-constrained settings, in the sense that it achieves the best possible competitive ratio of $1 - \frac{e_{max}}{E}$ (Section 3.1).
- We show that if the online algorithm has a priori information about the largest job size in the *actual* input instance, then there is a *semi-online* algorithm ($EC\text{-}EDF^*$) with a competitive factor of 0.5. We further show that this is the theoretical upper bound that can be achieved by *any* semi-online algorithm with such additional information (Section 3.2).
- We extend our competitive analysis to more general settings with non-uniform value densities (Section 4.1) and those where the processor has the *Dynamic Voltage Scaling* capability (Section 4.2).
- Finally, we also analyze the *EC-EDF* algorithm within the context of the *resource augmentation* technique [28, 22] and characterize the conditions under which *EC-EDF* can achieve a competitive factor of 1, with *extra* energy or DVS feature (Section 5).

A table with a partial list of the basic results of this paper, with the performance bounds that we establish along with the competitive factors of our solutions, is presented below.

Settings	Bound	Our Solution
Online	$1 - \frac{e_{max}}{E}$	$1 - \frac{e_{max}}{E}$
Semi-online	0.5	0.5
Non-uniform value density	$\frac{1}{(k_{max})^{\frac{e_{max}}{E}}}$	$\frac{1}{2k_{max}}$

Before proceeding with the details of our solutions, we underline that competitive analysis is certainly not the only way to analyze the performance of a scheduling algorithm with uncertain job arrival patterns. For example, if we have a reasonable approximation of the input probability distribution, *average-case analysis* can be undertaken either analytically or experimentally (through simulations). However, when such information is not available or reliable and/or when *analytical worst-case performance guarantees* are sought, competitive analysis is of fundamental importance.

2 System Model, Assumptions, and Terminology

We consider a uni-processor system with a limited energy budget of E energy units. The system's total energy consumption during its operation cannot exceed this allowance. We assume that executing a job consumes one unit of energy per time unit. The system's energy consumption in idle state (i.e. when it is not executing any job) is negligible.

Real-time jobs enter and leave the system during its operation. We use ψ to denote the finite input sequence of jobs that arrive to the system during its operation. Preemptive scheduling is assumed. Each job J_i is represented by the tuple

(r_i, e_i, d_i). Here r_i and d_i are the release time and deadline of the job J_i, respectively. For the sake of clarity, we use e_i to denote the *size* of the job, indicating both its *execution time* and *energy* requirements (since execution per unit time requires unit energy). The *laxity* of the job J_i is given by $d_i - (r_i + e_i)$ Since ψ is finite and one unit of execution requires one unit of energy, similar to [1, 2], we define $E_b = \sum_{J_i \in \psi} e_i$ to be the minimum energy needed to complete *all* the jobs in ψ.

As mentioned in Section 1, as opposed to the well-studied online *overloaded* real-time scheduling settings (e.g. [8, 23]) where there is sufficient energy but insufficient time; our objective is to study the impact of the energy constraint on the competitiveness of online real-time scheduling in settings where there is sufficient time but insufficient energy to complete all the workload. Hence, we make the following assumption throughout the paper.

Underloaded energy-constrained system assumption: We assume that the input instance ψ is feasible in real-time sense; that is, there exists a feasible schedule where all the jobs in ψ meet their deadlines *if the energy constraint is not taken into account*.

Each job is associated with a *value* that is proportional to its execution time. We consider *firm real-time systems* where a job that successfully completes execution by its deadline contributes its value to the system; otherwise no value is obtained from this job [8, 23]. We consider the *uniform value density* model [8, 9, 11, 23], where the job's value is equal to its size (e_i). We also extend our results (in Section 4.1) to the more general non-uniform value density model. Note that the *off-line* problem of maximizing the total system value can be easily shown to be *NP-Hard* even for the simple case where all jobs have the same deadline, by slightly modifying the intractability proof given in [32].

We define e_{max} as the upper bound on the size of any job that can be introduced to the system. We assume $e_{max} \leq E$ since no algorithm can process jobs with energy requirement greater than E. Observe that this definition implies that a job with size exactly e_{max} may or may not appear in the *actual* input instance.

We denote the minimum processing time (and energy requirement) of any job in the system by δ. That is, the size of any job is no smaller than δ units[1]. E and e_i (for each job J_i) are assumed to be expressed as exact multiples of δ. Observe that this assumption guarantees that the number of jobs introduced to the system is polynomial with respect to the energy constraint E. However, the ratio $\frac{\delta}{E}$ may be arbitrarily low.

Before proceeding with our formal analysis, we would like to introduce a specific job arrival pattern that is proven very useful in deriving competitive factors in online real-time scheduling [8, 23]. Specifically, we say that *a set of* **sequential jobs** *with size 'x' are introduced to the system at time t*, if they have the following characteristics:
(1) all jobs in the set have size x,
(2) the first job is released at time t with deadline $t + x$, and,
(3) each subsequent job is released at the deadline of the previous job in the set.
Hence, all the jobs in the sequence have zero laxity. Note that a similar pattern of *sequential jobs* were crucial in obtaining the well-known competitive factor bound of 0.25 for overloaded real-time systems [8, 23].

[1]As energy and execution requirements (especially with DVS) are expressed as real numbers, δ is assumed to be a real number strictly greater than zero.

3 Basic Results

We start by re-iterating a basic proposition, which states that preemptive EDF achieves the best possible performance without the knowledge of future job release times, *if* the system has sufficient energy to execute the entire workload (which is assumed to be feasible in real-time sense).

Proposition 1 *If $E \geq E_b$ then preemptive EDF has a competitive factor of* 1.

Proof: Follows from the optimality of preemptive EDF in underloaded real-time environments [18]. □

However, in underloaded environments with scarce energy, preemptive EDF turns out to be a poor online algorithm:

Lemma 1 *If $E < E_b$, then preemptive EDF cannot guarantee a non-zero total value (hence, it cannot guarantee a non-zero competitive factor).*

Proof: Figure 1 shows how to construct an instance using n jobs $(J_1 \ldots J_n)$, with decreasing deadlines and increasing release times, for which preemptive EDF cannot guarantee a non-zero value. Let c and D be two positive numbers such that $c \leq e_{max}$ and $D \gg c$. All jobs have size of c units. Also, $r_1 = 0$ and $d_1 = D$. Given this, the release times and deadlines of any two successive jobs are related by the following equations:

$$r_i = r_{i-1} + c - \delta$$

$$d_i = d_{i-1} - \delta$$

Observe that when EDF is about to finish executing a job, another job with a shorter deadline is released. EDF preempts the currently running job in favor of the job with earlier deadline. Continuing this way, while executing J_n, for some $n \geq 1$, EDF depletes its entire energy budget at time $t = E$. Notice that at time $t = E$, EDF has n pending jobs and zero remaining energy. Since preemptive EDF does not execute any job to completion in this instance, its total value is zero. □

One can attempt to modify EDF with the following simple rule: Preempt J_l is favor of J_h with an earlier deadline, only if there is enough energy to execute J_h. Even with this augmented rule, EDF cannot provide more than a total value

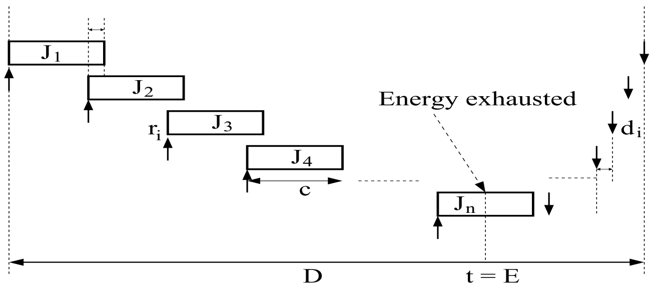

Figure 1: The worst-case instance for preemptive *EDF*

of δ by slightly modifying the scenario given in the proof of Lemma 1: at the very end when the system has δ units of remaining energy, one can release a job with size δ and zero laxity. This way, one would get a value of δ, which can be still arbitrarily low compared to E, making the algorithm still technically *non-competitive*.

An interesting question involves the *upper bound* on the performance of *any* online algorithm in energy-constrained settings, in *worst-case scenarios*. In establishing such upper bounds, the competitive analysis technique typically makes use of the so-called *adversary* method ([13, 9, 23, 30]). In this proof technique, the adversary generates an initial input job sequence, observes the online algorithms' behavior, and then decides on what further jobs should be released. This process is repeated a finite number of times. At some point (which must comply with the problem specification), the adversary announces the *optimal* schedule that it would generate for that input sequence – and a bound on the competitive factor is established by comparing it to the schedule selected by the online algorithm. Theorem 1 establishes this upper bound as a function of energy budget E and the upper bound on possible job size e_{max}.

Theorem 1 *In energy-constrained underloaded settings, no online algorithm can achieve a competitive factor greater than* $\frac{E-e_{max}}{E}$.

Proof: See Appendix for a full proof.

The proof of Theorem 1 is slightly involved, mainly because the input instance created by the adversary must comply with the system model and settings as described in Section 2. Specifically,

- The input sequence ψ created by the adversary *must be feasible* in real-time sense. Recall that our purpose is to analyze the online energy-constrained real-time scheduling for *underloaded* settings.
- The jobs released by the adversary in the input instance should not violate the upper bound on possible job size (e_{max}). For example, a job of size $E - e_{max}$ cannot be released when $e_{max} < \frac{E}{2}$.
- Energy budget and execution requirements are expressed as positive real numbers.

Theorem 1 implies that, the upper bound on the competitive factor depends heavily on the ratio $\frac{e_{max}}{E}$: the higher this ratio, the lower the best achievable competitive factor. For example, if $\frac{e_{max}}{E} = 1/3$, then, no algorithm can achieve more than $2/3$ of the total value of a clairvoyant algorithm. However, as $\frac{e_{max}}{E} \to 1$, the upper bound approaches zero; implying that no online algorithm can be competitive.

Further, for a given workload, as the system's initial energy budget E is reduced, the best achievable competitive factor quickly decreases. Similarly, for a given energy budget E, as the upper bound on job size e_{max} for the workload increases, the best achievable competitive factor quickly decreases. Nevertheless, we show that an online algorithm that is able to achieve this upper bound indeed exists.

3.1 Algorithm *EC-EDF*

In this subsection, we develop and show the optimality of an online algorithm called *EC-EDF* for energy-constrained real-time scheduling in underloaded settings. *EC-EDF* uses an *admission test* to admit new jobs that arrive at run-time. Specifically, if the newly-arriving job J fails to pass the admission test, it is discarded (i.e. never executed). In case that it passes the test, the new job J is added to the set $\mathcal{C}$ of *admitted jobs*. When a job completes execution, it is removed from set $\mathcal{C}$.

EC-EDF effectively *commits* to *all* the admitted jobs, in the sense that, as formally shown below, it guarantees their timely completion without violating the system's energy budget limits. Further, all admitted jobs are scheduled according to the well-known preemptive EDF policy.

The admission test uses the following relatively simple rule to decide whether the new job J arriving at time t can be admitted or not: J **is admitted if and only if the system's remaining energy budget at time** t**,** E^r**, is sufficient to fully execute** J **and the *remaining* workload of *all* the pending admitted jobs (i.e. the remaining workload in set** $\mathcal{C}$**).**

Let E^r and e_i^r represent the remaining energy with the system and the remaining execution time of job J_i at time t respectively. We assume both E^r and e_i^r are properly updated at run-time. Hence, formally, a job J with size e arriving at time t is admitted to the system *if and only if*

$$E^r \geq e + \sum_{J_i \in \mathcal{C}} e_i^r$$

We now give an example illustrating the behavior of *EC-EDF*. Consider a system with $E = 100$. The following four jobs constitute the input sequence: $J_1(0, 20, 200)$, $J_2(10, 30, 190)$, $J_3(25, 75, 150)$ and $J_4(85, 15, 120)$. Figure 2 shows the schedules generated by *EC-EDF*, *EDF* and the clairvoyant algorithm along with the total values obtained by them. In each schedule, the unshaded jobs are those that complete before the corresponding algorithm depletes its energy budget, while the shaded jobs are those that fail to do so.

At time $t = 0$, *EC-EDF* admits J_1 and dispatches it ($\mathcal{C} = \{J_1\}$). At $t = 10$, J_2 with higher priority than J_1 arrives. *EC-EDF* admits J_2 as there is enough remaining energy to execute

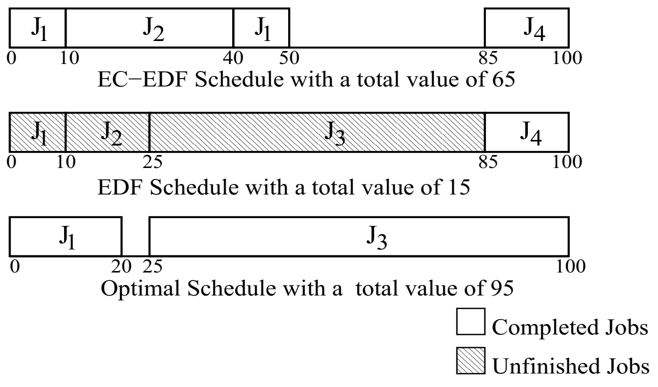
J1
J2
J1
J4
0 10 40 50 85 100
EC−EDF Schedule with a total value of 65
J1
J2
J3
J4
0 10 25 85 100
EDF Schedule with a total value of 15
J1
J3
0 20 25 100
Optimal Schedule with a total value of 95
Completed Jobs
Unfinished Jobs

Figure 2: Schedules Generated by *EC-EDF*, *EDF* and *Optimal*

both J_2 and the pending workload of admitted jobs, $\mathcal{C} = \{J_1\}$ (i.e. $E^r \geq e_2 + e_1^r$). *EC-EDF* updates set $\mathcal{C}$ to $\{J_1, J_2\}$.

Notice that at $t = 25$ when the largest job J_3 in the set arrives, the remaining energy is sufficient to execute it. However *EC-EDF* does not admit J_3 as with the remaining energy of 75 units, the system cannot execute J_3 *and* the pending workload of admitted jobs $\mathcal{C} = \{J_1, J_2\}$ (i.e. $E^r < e_3 + e_1^r + e_2^r$).

At $t = 85$, *EC-EDF* admits and executes J_4 to completion. Thus, by executing jobs J_1, J_2 and J_4 to completion, *EC-EDF* gathers a total value of 65. It is straightforward to verify that EDF gets a total value of only 15. The clairvoyant algorithm knowing the future job sizes and arrival patterns, skips certain jobs and idles as necessary, making a total value of 95 as shown in Figure 2.

Proposition 2 EC-EDF *guarantees the completion of* all *the admitted jobs before their deadlines, without violating the system's energy budget limits.*

Proof: Assume there exists a non-empty subset $\mathcal{C}'$ of the admitted jobs that cannot be completed in a timely manner without violating the energy budget limits. We will show by contradiction that $\mathcal{C}'$ cannot exist.

Recall that by assumption, the input sequence ψ is guaranteed to be feasible from timing constraints point of view. Thus, $\mathcal{C}' \subset \psi$ is also feasible. Further, *EC-EDF* schedules the admitted jobs using preemptive EDF which is known to be optimal in underloaded conditions. Hence, if $\mathcal{C}'$ exists then *EC-EDF* must run out of energy before completing the jobs it admitted. Since the admission rule of *EC-EDF* ensures that there is always enough remaining energy in the system to meet the computational demand of all pending admitted jobs along with the newly admitted job, we reach a contradiction. Hence, $\mathcal{C}'$ cannot exist. $\square$

Theorem 2 EC-EDF *has a competitive factor of* $\frac{E-e_{max}}{E}$.

Proof: First, note that if $E_b \leq E$, *EC-EDF* reduces to traditional EDF and can execute all the jobs in the input sequence ψ (which is assumed to be feasible), achieving a competitive factor of one. Thus, the adversary must create a *feasible* input sequence ψ such that $E_b > E$. Under this condition, eventually *EC-EDF* will be forced to discard a job due to energy limitations. Let J_d be the first job discarded by *EC-EDF* at time t. At time t, the system has the remaining energy of E^r units and let the amount of energy required to complete all *pending admitted* jobs be R units. Given this, the following condition must hold at time t: $E^r < e_d + R$.

Notice that by time t, *EC-EDF* has depleted $E - E^r$ units of energy in processing jobs. The pending workload of the admitted jobs is R. Therefore, the total workload size introduced by the adversary up to time t (including e_d) is $(E - E^r) + (R) + (e_d) > E$ (since $e_d + R - E^r > 0$). Also, since *EC-EDF* discards J_d, it has so far admitted a workload of at least $\psi' = E - e_d + \delta$ units. From Proposition 2, *EC-EDF* is guaranteed to complete ψ' in a timely manner without violating the system energy budget limits. Thus, *EC-EDF* guarantees a total value of $E - e_d + \delta$. Since $e_d \leq e_{max}$ by definition, the total value obtained by *EC-EDF* is no less than $E - e_{max} + \delta$, while the adversary can make at most a value of E. Since $\frac{\delta}{E}$ may be arbitrarily low, we conclude that *EC-EDF* has a competitive factor of $\frac{E-e_{max}}{E}$. $\square$

We also underline that in settings where $E \geq E_b$, *EC-EDF* is able to finish all the jobs in the input sequence thanks to the optimality of *EDF*, achieving a competitive factor of 1 under that condition. That is, **regardless of the relationship between E and E_b, *EC-EDF* yields the best competitive factor**.

3.2 A Semi-online Algorithm with a Constant Competitive Factor

As mentioned in Section 1, it is occasionally possible to improve the competitive ratio by providing some limited information about the *actual* input sequence. For example, online scheduling algorithms typically benefit from information about the maximum job size, sum of job processing times or job size patterns in the actual input [4, 30]. The online algorithms that exploit this type of limited information about the actual input are called *semi-online* and their design and analysis have been recently attracting increasing interest [4, 30].

In our problem as well, the knowledge of the largest job size in the actual input instance turns out to be very helpful. For the uniform value density model where the value of a job is equal to its execution time, the largest size job *in the input instance* is also the most valuable job in the set. Below, we describe an algorithm $EC\text{-}EDF^*$ that achieves a competitive factor of 0.5, using this information. Further, we show that with only the additional knowledge of the largest job size, no semi-online algorithm can perform better than $EC\text{-}EDF^*$, demonstrating its optimality.

Let $e_l \leq E$ represent the largest job size in the actual input sequence. We first give the rules for $EC\text{-}EDF^*$. $EC\text{-}EDF^*$ exploits the information about e_l: in semi-online settings, the fact that at least one job of size e_l will be part of the input sequence is *guaranteed* by definition. $EC\text{-}EDF^*$ compares e_l to the available energy budget E. If $e_l > \frac{E}{2}$, $EC\text{-}EDF^*$ simply waits for the largest size job, e_l, and executes it. Since $E \geq e_l > \frac{E}{2}$, the value of $EC\text{-}EDF^*$ is no less than $\frac{E}{2} + \delta$. The adversary can gather a value of at most E. On the other hand, if $e_l \leq \frac{E}{2}$, $EC\text{-}EDF^*$ schedules jobs using *EC-EDF* . By definition, no job size in the actual input sequence can be larger than e_l. Thus, from Theorem 2 the competitive factor for *EC-EDF* becomes $\frac{E-e_l}{E}$. Further, given the constraint $e_l \leq \frac{E}{2}$, the lower bound on competitive factor of *EC-EDF* and hence that of $EC\text{-}EDF^*$ reduces to 0.5. Thus, in either case, $EC\text{-}EDF^*$ makes at least half of the value of the adversary.

Lemma 2 *$EC\text{-}EDF^*$ has a competitive factor of* 0.5.

Lemma 3 *With only the additional knowledge of the largest job size, no semi-online algorithm can achieve a competitive factor greater than* 0.5.

Proof: We describe a scenario through which the adversary effectively limits the total value of *any* semi-online algorithm to half of its value. As mentioned in the proof of Theorem 1, E can be expressed as $E = k \cdot e_l + k' \cdot \delta$, where k and k' are integers, $k \geq 1$, $k' \geq 0$. The construction is similar to that given in the proof of Theorem 1. The adversary releases *sequential* jobs in three stages. Jobs of size $k' \cdot \delta$ are introduced in the first stage, while jobs of size δ appear in the second stage. In the third stage the adversary now releases jobs of size e_l. Again, if $k' = 0$, the first stage can be skipped. We distinguish two cases.

Case 1: The semi-online algorithm $\mathcal{A}$ skips all these jobs until time $t = E$. Observe that in the pattern of jobs released in the proof of Theorem 1, the third stage cannot have started at time $t < E$. Thus, in this scenario, the adversary introduces a single job of size e_l at time $t = E$ and then stops releasing jobs. Since $\mathcal{A}$ has skipped all sequential jobs, it will be able to execute the job of size e_l gathering a value of e_l, while the adversary gathers a value of E by executing all the sequential jobs up to time $t = E$. The competitive factor is $\frac{e_l}{E}$.

Case 2: $\mathcal{A}$ executes one of the sequential jobs at $t < E$. The scenario can now be mapped to that in the proof of Theorem 1. Therefore, by repeating the arguments of Theorem 1, the upper bound on the competitive factor can be evaluated as $\frac{E-e_l}{E}$.

By choosing $e_l = \frac{E}{2}$ one can restrict the total value of $\mathcal{A}$ to $\frac{E}{2}$ for both Cases 1 and 2. Thus, no semi-online algorithm can achieve a competitive factor greater than 0.5. □

Corollary 1 *Among semi-online algorithms that have only the additional knowledge of the largest job size, EC-EDF* is optimal.*

We remark that, in online real-time scheduling in *overload* conditions, the best achievable competitive factor was shown to be 0.25 [8] and further, there exists an algorithm D^{over}, with this competitive factor [23]. In contrast, in online real-time scheduling with hard energy constraints for underloaded settings, with the knowledge of the largest job size in the actual input sequence, the best achievable competitive factor is 0.5 which is twice as large as that in overloaded settings. Further, the algorithm $EC\text{-}EDF^*$ achieves this competitive factor.

4 Extensions to more general settings

In this section, we extend our analysis to more general settings involving non-uniform value density and those with the advanced power management features (DVS).

4.1 Non-Uniform Value Densities

As mentioned previously, each job is associated with a value proportional to its execution time. The *value density* of a job is its value divided by its execution time. The ratio of the largest value density to the smallest value density is called *importance ratio* [8, 23]. In uniform density settings (Section 3), the *importance ratio* is one and hence the value of the job is equal to its size. In this section, we extend our competitive analysis to the more general non-uniform density settings.

In these settings, the value of a job J_i with value density k_i is given by $k_i e_i$. Let k_{min} and k_{max} denote the smallest and largest value densities that can be associated with any job. Without loss of generality, we assume that $k_{min} = 1$ and thus the ratio of $\frac{k_{max}}{k_{min}}$ is simply k_{max}. Observe that in comparison to uniform density settings, the largest size job is no longer guaranteed to be the most valuable job.

Theorem 3 *In non-uniform value settings where $k_{max} > 1$, no online algorithm can achieve a competitive factor greater than* $\frac{1}{(k_{max})^{\frac{e_{max}}{E}}}$.

The full proof of Theorem 3 is given in [19], due to space constraints. We add that, the upper bound on performance established by Theorem 3 holds with or without the knowledge of the largest job size in the input sequence. Thus, the a priori knowledge of the largest job size does not help design better online algorithms in non-uniform value density settings. Lemma 4 establishes the competitive factor of the semi-online algorithm, $EC\text{-}EDF^*$, which uses the additional knowledge of largest job size.

Lemma 4 *Algorithm $EC\text{-}EDF^*$ has a competitive factor of* $\frac{1}{2k_{max}}$.

Proof: Let $e_l \leq E$ represent the largest job size in the input sequence. If $e_l > \frac{E}{2}$, $EC\text{-}EDF^*$ waits for the largest job of size e_l and is guaranteed a value of $k_{min}(\frac{E}{2} + \delta) = \frac{E}{2} + \delta$. The adversary can make at most $k_{max}E$. The competitive factor is $\frac{1}{2k_{max}}$. On the other hand, if $e_l \leq \frac{E}{2}$, $EC\text{-}EDF^*$ follows *EC-EDF* which would guarantee a value of at least $k_{min}(E - e_l + \delta)$ even when executed on jobs with the minimum value density $k_{min} = 1$. Again, the adversary can make a total value of at most $k_{max}\ E$. Thus, the competitive factor is found as the minimum value of $\frac{E-e_l+\delta}{k_{max}\ E}$ which is $\frac{1}{2k_{max}}$ (obtained when $e_l = \frac{E}{2}$). As a result, in both cases, the competitive factor is $\frac{1}{2k_{max}}$. □

We conclude this subsection with the following remarks.

- The competitive factor of the algorithm heavily depends on the ratio $\frac{e_{max}}{E}$. When $e_{max} \leq \frac{E}{2}$, the best achievable competitive factor is $\frac{1}{\sqrt{k_{max}}}$. As $\frac{e_{max}}{E} \to 1$, the best achievable competitive factor reduces to $\frac{1}{k_{max}}$.
- The tightness of the upper bound established in Theorem 3 is an interesting open problem that we are currently investigating.

4.2 Extensions to DVS settings

Dynamic Voltage Scaling (DVS) is a popular energy management technique through which the processor frequency and voltage can be varied at run-time. Since processor power consumption increases in convex manner with processor clock frequency, DVS significantly reduces processor dynamic energy consumption. DVS has been subject to extensive research in the past decade [5, 25, 29]. There have also been a few research research efforts on competitive analysis of DVS based systems, mostly relating to energy minimization of sporadic jobs [6, 15, 34] (without the hard energy constraint). DVS-capable processors can play a critical role in energy-constrained real-time systems. Using effective DVS policies, which help minimize the energy spent on processing a workload, the energy-constrained system can successfully process additional workload with a given energy budget, thus giving higher overall system value compared to non-DVS systems. In this section, we derive an upper bound on the competitive factor of DVS-enabled online algorithms.

We consider a DVS capable processor whose frequency can be varied in the range $[f_{min}, f_{max}]$. Without loss of generality, we assume $f_{max} = 1$. Power consumption of the processor at frequency f is modeled as a convex function $P(f) = af^{\alpha}$, where a is a constant characterized by the processor parameters[2] and $2 \leq \alpha \leq 3$. Thus, at frequency f, the time and energy required to execute a job with processing time x are given by $\frac{x}{f}$ and $E(f) = P(f) \cdot \frac{x}{f} = f^{\alpha-1}x$, respectively.

Notice that with DVS, the energy required to execute a job depends on *the frequency* at which the job is executed. As such, there is no longer one-to-one correspondence between job execution times and energy requirements. By taking this into account, in this section, we represent a job J_i as $J_i(r_i, c_i, D_i, e_i)$, where r_i is its release time, c_i is its *execution time at frequency* f_{max}, D_i is its *relative deadline* and e_i is its *minimum energy requirement*. The minimum energy requirement e_i is computed by assuming the *minimum frequency* $\frac{c_i}{D_i}$ that would allow the timely completion of J_i before its deadline. That is, $e_i = E(\frac{c_i}{D_i}) = (\frac{c_i}{D_i})^{\alpha-1} \cdot c_i$.

Note that we assume the uniform density model throughout this section and the value of a job J_i is assumed to be equal to c_i (execution time at f_{max}). That is, executing a job at a low frequency reduces the energy consumption but does not affect its value. Before proceeding, we give some basic definitions and existing results that will be instrumental in our derivation.

Definition 1 *Let $l(t_1, t_2)$ denote the total amount of workload of jobs with release times at or later than t_1 and deadlines at or earlier than t_2. The effective loading factor $h(t_1, t_2)$ over an interval $[t_1, t_2]$ is defined as $h(t_1, t_2) = \frac{l(t_1,t_2)}{t_2-t_1}$.*

Definition 2 *The absolute effective loading factor (or simply the loading factor) β is the maximum effective loading factor over all intervals $[t_1, t_2]$: $\beta = max(h(t_1, t_2)),\ \ 0 \leq t_1 < t_2$.*

Theorem 4 (from [7, 20]) *A set of real-time jobs can be scheduled in feasible manner (by preemptive EDF) if and only if $\beta \leq 1$.*

Given the loading factor β, if the processor executes all jobs at constant frequency $f = max(f_{min}, \beta)$, then the new loading factor β' (increased due to the reduced frequency) would be $\frac{\beta}{f}$. Further, one can easily verify that β' would still not exceed 1.0 under that condition [34]. Thus, running jobs at frequency $f = max(f_{min}, \beta)$ preserves the system feasibility (without the energy constraint).

Proposition 3 *A DVS algorithm that runs at constant speed $f \geq max(f_{min}, \beta)$ cannot make a total value $> \frac{E}{f^{\alpha-1}}$.*

Proposition 3 can be justified by observing that with DVS, to deplete e units of energy, the system will have to execute a workload of $\frac{e}{f^{\alpha-1}}$ at frequency f. Thus, with e units of energy, a maximum value of $\frac{e}{f^{\alpha-1}}$ can be made by running the processor at constant speed f. Note that this implies that with DVS the system is able to achieve a value greater than E. In settings where both the online algorithm and adversary have DVS, the pre-knowledge of β can potentially provide some advantage to the online algorithm. Theorem 5 characterizes this result.

Theorem 5 *Assuming $\beta > f_{min}$, where $0 < \beta \leq 1$,*
(i) Without the knowledge of β, there is no online DVS algorithm with a competitive factor greater than $f_{min}^{\alpha-1}$.
(ii) With the knowledge of β, there is no online DVS algorithm with a competitive factor greater than $(\frac{f_{min}}{\beta})^{\alpha-1}$.

Proof: Case 1: Assume β is unknown to the algorithm. Consider the following instance. The adversary sets $\beta = 1$ and introduces a job $J_1(0, E, E, E)$ (Notice the minimum energy requirement for J_1 is $e_1 = E$). Clearly, if the online algorithm

[2]In this section we assume $a = 1$. We also point out that recently there have been research efforts addressing *system energy* models, where it was shown that lowering frequency below a certain threshold (called *energy-efficient speed*) may adversely affect overall system energy consumption [3]. Our upper bound results can easily be adapted to these system-level energy models by enforcing arbitrarily low energy-efficient speed levels.

$\mathcal{A}$ does not execute J_1, it will miss its deadline. The adversary executes J_1 and releases no more jobs. The value of $\mathcal{A}$ is zero, while that of the adversary is E.

If $\mathcal{A}$ executes J_1, then, at time $t = E$, the adversary introduces $J_2(E, \frac{E}{k^{\alpha-1}}, \frac{E}{k^{\alpha}}, E)$. Observe that J_2 can be executed at frequency $\frac{\frac{E}{k^{\alpha-1}}}{\frac{E}{k^{\alpha}}} = k$. By skipping J_1, the adversary executes J_2 gathering a reward of $\frac{E}{k^{\alpha-1}}$. Thus, the competitive factor is $\frac{E}{\frac{E}{k^{\alpha-1}}} = k^{\alpha-1}$. Since the minimum possible frequency is f_{min}, by setting $k = f_{min}$, the adversary can force an upper bound of $(f_{min})^{\alpha-1}$ for the competitive factor.

Case 2: Assume β is known to the algorithm.
The pattern in Case 1 can be repeated with a slight modification. J_1 is given with parameters $J_1(0, \frac{E}{\beta^{\alpha-1}}, \frac{E}{\beta^{\alpha}}, E)$. $\mathcal{A}$ is forced to execute J_1 at frequency β to guarantee a non-zero total value. At that point, the adversary introduces J_2 with the following parameters $J_2(E, \frac{E}{f_{min}^{\alpha-1}}, \frac{E}{f_{min}^{\alpha}}, E)$. Observe that J_2 can be executed by the adversary at frequency f_{min}, yielding a value of $\frac{E}{f_{min}^{\alpha-1}}$. Further, since $f_{min} < \beta$, the processor loading factor can be easily shown to be β, satisfying the assumption. $\mathcal{A}$ has a value of $\frac{E}{\beta^{\alpha-1}}$ and thus the competitive factor is bounded by $(\frac{f_{min}}{\beta})^{\alpha-1}$. □

The case where $\beta \leq f_{min}$ is relatively simple: consider the algorithm *EC-EDF using a constant speed* f_{min}. Notice that this algorithm does *never* spend more energy than required while processing jobs, as the system limitations do not allow processing below f_{min}. Also, due to the same constraint, f_{min} is the least possible frequency with which the adversary can process jobs. As such, this case can be shown to be equivalent to the non-DVS case (Section 3), where both the online algorithm and the adversary had to process jobs at the same (constant) speed. Thus, all the results of Section 3 apply.

As a consequence of Theorem 5, the upper bound on competitive factor approaches zero as $f_{min} \to 0$ and $\beta \to 1$. An open problem under investigation is the performance of semi-online algorithms with the knowledge of both the largest job size in the actual input sequence and β in DVS settings. The results in this section mostly indicate increased difficulties for online algorithms operating on DVS-enabled settings. However, in the next section, we will show that the DVS feature can help an online algorithm to compete with a clairvoyant (but non-DVS-enabled) adversary.

5 Resource Augmentation

Noting that the competitive analysis characterizes performance guarantees in the worst-case scenarios, some recent efforts exploited alternative means to quantify the performance of online algorithms. *Resource augmentation* technique, introduced by [28] and popularized by [22], is such a framework. With resource augmentation, the online algorithm is given *additional* resources compared to the adversary in an effort to compensate for the lack of knowledge about the future. For example, the online algorithm may run on a faster processor [22], or it may have access to additional CPUs [10]. In the following, we show how resource augmentation can help significantly improve the performance of *EC-EDF*, especially when $\frac{e_{max}}{E}$ is close to one.

First, we explore the implications of providing the online algorithm *EC-EDF* with additional energy. Specifically, if the adversary possesses an energy budget of E units, then *EC-EDF* is assumed to have an initial energy of $(1 + x)E$ units, where $x > 0$. We know from Theorem 2 that given an initial energy of E, *EC-EDF* guarantees a value of at least $E - e_{max}$. Thus, with $(1 + x)E$ units of initial energy, *EC-EDF* guarantees a value of at least $(1 + x)E - e_{max}$. The competitive factor is $1 + x - \frac{e_{max}}{E}$. Hence, if $x = \frac{e_{max}}{E}$ then *EC-EDF* has a competitive factor of 1.

Proposition 4 *The online algorithm* EC-EDF *achieves a competitive factor of* 1 *compared to an adversary with E units of energy budget, if it is allocated* $(1 + \frac{e_{max}}{E})$ *units of energy.*

Further, since $\frac{e_{max}}{E} \leq 1$, we have:

Corollary 2 *If* EC-EDF *is provided twice as much energy as the clairvoyant adversary $\mathcal{C}_{adv}$, it becomes at least as powerful as $\mathcal{C}_{adv}$.*

Using the terminology made popular by the seminal resource augmentation analysis paper [22], we can state the following thanks to Proposition 4: *energy is as powerful as clairvoyance*.

In the following, we describe a practical way to effectively give more energy to *EC-EDF*. Specifically, we augment the *EC-EDF* scheduler with the knowledge of the absolute loading factor β, and a DVS-capable processor. We will show that *EC-EDF* can successfully compete with a clairvoyant adversary without DVS feature. With this resource augmentation, *EC-EDF* always executes all jobs at speed $f = \beta$. We refer to this modified *EC-EDF* as *β-EC-EDF*. Observe that since the processor always runs at constant speed β, to deplete e units of energy, the processor must execute $\frac{e}{\beta^{\alpha-1}}$ units of workload. Thus, the initial energy budget of *β-EC-EDF* is *effectively* $\frac{1}{\beta^{\alpha-1}}$ times that of the adversary. Following Theorem 2, *β-EC-EDF* guarantees a value of $\frac{E}{\beta^{\alpha-1}} - e_{max}$.

Proposition 5 β-EC-EDF *has a competitive factor of* $\frac{E-\beta^{\alpha-1}e_{max}}{\beta^{\alpha-1}E}$.

As a consequence of Proposition 5 and since $e_{max} \leq E$, we have the following:

Corollary 3 *If* $\beta \leq (\frac{1}{2})^{\frac{1}{\alpha-1}}$, β-EC-EDF *is as powerful as a clairvoyant adversary without DVS.*

6 Conclusion

In this paper, we undertook a preliminary study of competitive analysis for energy-constrained online real-time scheduling in underloaded settings. We proposed an optimal algorithm $EC\text{-}EDF$ which achieves the best possible performance guarantee obtainable by any online algorithm. Further, by assuming the knowledge of the largest job size, we proposed an optimal semi-online algorithm $EC\text{-}EDF^*$, which has a competitive factor of 0.5. We extended our analysis and provided fundamental results in various models and settings including those of non-uniform value density and DVS. To the best of our knowledge, this is the first theoretical investigation of the problem of online real-time scheduling in underloaded but energy-constrained environments. We hope that this research effort will trigger further research in this direction.

References

[1] T. A. AlEnawy and H. Aydin. Energy-Constrained Scheduling for Weakly-Hard Real-Time Systems. In *Proc. of the Real-time Systems Symposium (RTSS'05)*, 2005.

[2] T. A. AlEnawy and H. Aydin. On Energy-Constrained Real-Time Scheduling. In *Proc. of the European Conference on Real-Time Systems (ECRTS'04)*, 2004.

[3] H. Aydin, V. Devadas, and D. Zhu. System-level Energy Management for Periodic Real-Time Tasks. In *Proc. of Real-Time Systems Symposium (RTSS'06)*, 2006.

[4] T. Ebenlendr and J. Sgall. Semi Online Preemptive Scheduling: One Algorithm for All Variants. In *Proc. International Symposium on Theoretical Aspects of Computer Science (STACS'09)*, 2009.

[5] H. Aydin, R. Melhem, D. Mosse, and P. Mejia-Alvarez. Power-aware Scheduling for Periodic Real-time Tasks. *IEEE Transactions on Computers*, Vol. 53, No. 10, 2004.

[6] N. Bansal, T. Kimbrel, and K.Pruhs. Dynamic speed scaling to manage energy and temperature. In *Symposium on Foundations of Computer Science (FOCS'04)*, 2004.

[7] S. Baruah, L. Rosier, and R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. In *Real Time Systems(2)*. 1990

[8] S. Baruah, G. Koren, B. Mishra, A. Raghunathan, L. Rosier, and D. Shasha. Online Scheduling in the presence of Overload. In *Proc. of the Symposium on Foundations of Computer Science (FOCS'91)*, 1991.

[9] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, and F. Wang. On the Competitiveness of Online Real-Time Task Scheduling. In *Proc. of the Real-Time Systems Symposium (RTSS'91)*, 1991.

[10] S. Baruah. Overload Tolerance for Single-Processor Workloads. In *Proc. of the Real-Time Technology and Application Symposium (RTAS'98)*, 1998.

[11] S. Baruah and J. Haritsa. Scheduling for Overload in Real-Time Systems. *IEEE Transactions on Computers*, Vol. 46, No. 9, 1997.

[12] S. Baruah and M. E. Hickey. Competitive Online Scheduling of Imprecise Computations. *IEEE Transactions on Computers*, Vol. 47, No. 9, 1998.

[13] A. Borodin and R. El-Yavin *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[14] G. Buttazzo. *Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications*, Second Edition, Springer, 2005

[15] H. L. Chan, W. T. Chan, T. W. Lam, L. K. Lee, K. S. Mak, and P. Wong. Energy Efficient Online Deadline Scheduling. In *Proc. of the Symposium On Discrete Algorithms (SODA'07)*, 2007.

[16] J. J. Chen and T. W. Kuo. Voltage Scaling Scheduling for Periodic Real-Time Tasks in Reward Maximization. In *Proc. of the Real-Time System Symposium (RTSS'05)*, 2005.

[17] M. Dertouzos and A. K. Mok. Multiprocessor Online Scheduling for Hard Real-Time Tasks. *IEEE Transactions on Software Engineering*, Vol 15, No. 12, 1989.

[18] M. Dertouzos. Control Robotics: the procedural control of physical processes. In *Proc. IFIP Congress*, 1974.

[19] V. Devadas, F. Li, and H. Aydin. Competitive Analysis of Energy-Constrained Online Real-Time Scheduling. *Technical Report.* Computer Science Department, George Mason University, January 2009.

[20] K. Jeffay and D. L. Stone. Accounting for Interrupt Handling Costs in Dynamic Priority Task Systems. In *Proc. of the Real-Time Systems Symposium (RTSS'93)*, 1993.

[21] K. Jeffay, D. F. Stanat, and C. U Martel. On Non-Preemptive Scheduling of Periodic and Sporadic Tasks. In *Proc. of the Real-Time Systems Symposium (RTSS'91)*, 1991.

[22] B. Kalyanasundaram and K. Pruhs. Speed is as Powerful as Clairvoyance. In *Proc. of the Symposium on Foundations of Computer Science (FOCS'95)*, 1995.

[23] G. Koren and D. Shasha. D-over: An Optimal Online Scheduling Algorithm for Overloaded Real-Time Systems. In *Proc. of the Real-Time Systems Symposium (RTSS'92)*, 1992.

[24] G. Koren, D. Shasha and S. C. Huang. MOCA: A Multiprocessor Online Competitive Algorithm for Real-Time Scheduling. In *Proc. of the Real-Time Systems Symposium (RTSS'93)*, 1993.

[25] C. H. Lee and K. G. Shin. Online Dynamic Voltage Scaling for Hard Real-Time Systems Using the EDF Algorithm. In *Proc. of the Real-Time Systems Symposium (RTSS'04)*, 2004.

[26] J. Liu. *Real Time Systems* Prentice Hall, NJ, 2000.

[27] M. A. Palis. Competitive Algorithms for Fine-Grain Real-Time Scheduling. In *Proc. of the Real-Time Systems Symposium (RTSS'04)*, 2004.

[28] C. Phillips, C. Stein, E. Torng, and J. Wein. Optimal Time-Critical Scheduling via Resource Augmentation. *Algorithmica*, pp. 163-200, 2002.

[29] P. Pillai and K. G. Shin. Real-time Dynamic Voltage Scaling for Low-power Embedded Operating Systems. In *Proc. of the Symposium on Operating Systems Principles (SOSP'01)*, 2001.

[30] K. Pruhs, J. Sgall, and E. Torng. In *the Handbook of Scheduling, Algorithms, Models and Performance Analysis*. CRC press, FL, USA, 2004, edited by J.Y.T. Leung.

[31] C. Rusu, R. Melhem, and D. Mosse. Maximizing Rewards for Real-Time Applications with Energy Constraints. In *ACM Transactions of Embedded Computing Systems*, Vol 2, No. 4, 2003.

[32] C. Rusu, R. Melhem, and D. Mosse. Maximizing the System Value while Satisfying Time and Energy Constraints. In *Proc. of the Real-Time System Symposium (RTSS'02)*, 2002.

[33] H. Wu, B. Ravindran, and E. D. Jensen. Utility Accrual Real-Time Scheduling Under the Unimodal Arbitrary Arrival Model with Energy Bounds. In *IEEE Transactions on Computers* , Vol. 56, No. 10, 2007.

[34] F. Yao, A. Demers, and S. Shenker. A scheduling model for reduced CPU energy. In *Proc. of the Symposium on Foundations of Computer Science (FOCS'95)*, 1995.

APPENDIX: Proof of Theorem 1

To prove Theorem 1, we are going to create an input sequence ψ such that for any given positive small value δ and for any online algorithm $\mathcal{A}$, $\mathcal{A}$ accrues a value no more than $E - e_{max} + \delta$ and the adversary gains a total value of E. Thus, the competitive factor of $\mathcal{A}$ will be shown to be no better than $\frac{E-e_{max}}{E}$. Recall that E and e_i for any job J_i are exact multiples of δ. This implies e_{max}, the upper bound on the size of any job, is also an exact multiple of δ.

Let $E = k \cdot e_{max} + e'$, where k is an integer, $k \geq 1$, and $0 \leq e' < e_{max}$. Furthermore, $e_{max} = k_x \cdot \delta$, where k_x is an integer. Also, note that $e' = k' \cdot \delta$, where k' is an integer. Then, we can write:

$$E = k \cdot k_x \cdot \delta + k' \cdot \delta,$$

where k, k_x, and k' are integers, $k_x, k \geq 1$, $k' \geq 0$.

In the following, we feed the algorithm $\mathcal{A}$ the input sequence ψ such that there exists a time t, where $\mathcal{A}$ obtains a total value of no more than $E - e_{max} + \delta$. Further, $\mathcal{A}$ will be unable to accrue any additional value after time t.

$\mathcal{A}$'s schedule is divided into up to three stages, based on its action over the released jobs. We start at time $t = 0$. In the first stage, the adversary releases jobs with size $k' \cdot \delta$. If $k' = 0$, the first stage is not needed and we proceed directly to the second stage. In the second stage, it releases jobs with size δ, and in the third stage, it releases jobs with size of e_{max} $(= k_x \cdot \delta)$. Note that, as we will see, the third stage may not be always needed and the number of jobs released in each stage depends on $\mathcal{A}$'s actions.

Assume $k' \neq 0$. In the first stage, the adversary keeps introducing *sequential jobs* with size of $k' \cdot \delta$ (recall from Section 2, that sequential jobs have zero laxity and are released back to back one at a time) until $\mathcal{A}$ picks up one such job to run. Let $N = \lfloor \frac{E}{k' \cdot \delta} \rfloor$. The adversary stops releasing new jobs as soon as (1.) the online algorithm $\mathcal{A}$ picks up one such job to run, or, (2.) the number of released jobs reaches N.

a. Assume $\mathcal{A}$ does not accept any job until the time when the adversary has released N such jobs. In this case, the adversary announces that it has executed all N released jobs in its optimal schedule. Observe that, at this point, the adversary has accrued a total value of $N \cdot k' \cdot \delta$ and its remaining energy is $E - N \cdot k' \cdot \delta$. $\mathcal{A}$ has E units of remaining energy, but its total value is still zero. Now, if $N \cdot k' \cdot \delta = E$, then the adversary does not introduce any new jobs and the competitive factor of $\mathcal{A}$ is zero. Otherwise, the first stage ends and we enter the second stage.

b. Assume $\mathcal{A}$ selects a job with size $k' \cdot \delta$ to run before the adversary has released N such jobs. In this case, as soon as $\mathcal{A}$ executes its first job, the first stage ends. The adversary executes the same job selected by $\mathcal{A}$. Observe that now both the adversary and $\mathcal{A}$ have obtained a value of $k' \cdot \delta$. Their remaining energy levels are the same and equal to $E - k' \cdot \delta$. Then, the second stage follows.

In the second stage, the adversary keeps releasing sequential jobs with size δ. If the algorithm does not pick up any job in the first stage to run (i.e., the above case a.), the adversary keeps releasing such jobs until the number of jobs with size δ reaches

$$\frac{E - N \cdot k' \cdot \delta}{\delta} = \frac{E - \lfloor \frac{E}{k' \cdot \delta} \rfloor \cdot k' \cdot \delta}{\delta} \leq k'.$$

Then, the adversary stops releasing any new jobs. We can see that the algorithm $\mathcal{A}$ cannot obtain a value exceeding $k' \cdot \delta$ $(\leq E - e_{max})$. But the adversary can obtain a total value of E, by executing all the jobs it has released in two stages. Thus, the competitive factor c is:

$$c \leq \frac{k' \cdot \delta}{E} \leq \frac{E - e_{max}}{E}.$$

In the following, we consider the case where the algorithm picks up one job in the first stage to run (i.e., we focus on the case b. above). The second stage ends as soon as (1.) the algorithm $\mathcal{A}$ picks up one job with size δ to run, or (2.) the adversary releases $M = \frac{E - k' \cdot \delta}{\delta} = k \cdot k_x$ such jobs.

c. Assume $\mathcal{A}$ does not accept any job until the time when the adversary has released M such jobs. In this case, the second stage ends and we do not need to go to the third stage. The adversary does not release any new job. The algorithm obtains a total value of $k' \cdot \delta$. The adversary accrues a total value of $k' \cdot \delta + k \cdot k_x \cdot \delta = E$. Thus, the competitive factor c is

$$c = \frac{k' \cdot \delta}{E} \leq \frac{E - e_{max}}{E}.$$

d. Assume $\mathcal{A}$ selects a job with size δ for execution before the adversary has released M such jobs. Again, the second stage ends at this point. The adversary does not execute any of these jobs, its accumulated value remains at $k' \cdot \delta$ and its remaining energy is still $E - k' \cdot \delta$. The algorithm has an accumulated value of $k' \cdot \delta + \delta$, its remaining energy is $E - k' \cdot \delta - \delta$, and we continue with the third stage.

In the third stage, we release a set of jobs with size e_{max}. We are going to show that at least one such job cannot be executed by the online algorithm $\mathcal{A}$. Based on the remaining energy for both algorithms ($\mathcal{A}$ and the adversary), we are going to force the algorithm $\mathcal{A}$ to accept at least one fewer job with processing time of e_{max} in the following way. The adversary releases k jobs each with size e_{max} and they share a common relative deadline of $k \cdot e_{max}$. Note that the algorithm $\mathcal{A}$ can accept at most $k - 1$ such jobs (because $k' \cdot \delta + \delta + k \cdot e_{max} > E$) while the adversary can accept all these jobs $(k' \cdot \delta + k \cdot e_{max} = E)$. The adversary obtains a total value of E (recall it had executed one job of size $k' \cdot \delta$ released in the first stage) and the algorithm $\mathcal{A}$ has a total value $\leq E - e_{max} + \delta$. Since $\frac{\delta}{E}$ can be arbitrarily low, the competitive factor c is again found as $\frac{E-e_{max}}{E}$. $\square$