Real-Time Syst (2010) 46: 88–120
DOI 10.1007/s11241-010-9100-y

Competitive analysis of online real-time scheduling algorithms under hard energy constraint

Vinay Devadas · Fei Li · Hakan Aydin

Published online: 14 July 2010

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-EDF** which achieves a constant competitive factor of 0.5. This turns out to be the best achievable competitive factor in these settings. In non-uniform value density settings, we derive an upper bound on the competitive factor achievable by any online algorithm, as well as the competitive factors of *EC-EDF* and *EC-EDF** algorithms. We also investigate the implications of having additional constraints on the online scheduling algorithm, including *non-idling* and *non-preemptive* execution constraints. Moreover, we analyze the case where the processor can adjust its speed at run-time through Dynamic Voltage Scaling (DVS) capability.

Keywords Real-time systems · Energy management · Competitive analysis

V. Devadas · F. Li · H. Aydin (✉)
Department of Computer Science, George Mason University, Fairfax, VA 22030, USA
e-mail: aydin@cs.gmu.edu

V. Devadas
e-mail: vdevadas@cs.gmu.edu

F. Li
e-mail: lifei@cs.gmu.edu

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 (Borodin and El-Yavin 1998). 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 Borodin and El-Yavin (1998), Pruhs et al. (2004) 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 (Dertouzos 1974). 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 (Dertouzos and Mok 1989).

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 (Baruah et al. 1991a, 1991b; Buttazzo 2005). 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* (Baruah et al. 1991a; Buttazzo 2005; Koren and Shasha 1992). 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 factor, 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 (Borodin and El-Yavin 1998; Pruhs et al. 2004).

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 (Pruhs et al. 2004). 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 factor (Pruhs et al. 2004).

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 (Baruah et al. 1991a, 1991b; Baruah and Haritsa 1997; Koren and Shasha 1992). In a seminal paper, Baruah et al. showed that no online algorithm can achieve a competitive factor greater than 0.25 (Baruah et al. 1991a), in an overloaded real-time system. This result holds for task systems with *uniform density* settings, where the value 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 time 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 (Baruah et al. 1991a).

Remarkably, Koren and Shasha provided an optimal online algorithm D^{over} which achieved this upper bound (Koren and Shasha 1992). The same authors also considered extensions to multiprocessor settings (Koren et al. 1993). Several other studies addressed competitive online real-time scheduling for imprecise computation tasks (Baruah and Hickey 1998), tasks with bounded slack factors (Baruah and Haritsa 1997), and tasks with given stretch metrics (Palis 2004) 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 (Aydin et al. 2004, 2006; Pillai and Shin 2001). 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 (AlEnawy and Aydin 2004, 2005; Chen and Kuo 2005; Rusu et al. 2002, 2003; Wu et al. 2007). In the latter formulation, energy is effectively *the* hard constraint and the system is said to be *energy-constrained* (AlEnawy and Aydin 2004, 2005).

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 online real-time system settings with uniform value density to energy-constrained environments, 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 (Sect. 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 a competitive factor equal to the upper bound of $1 - \frac{e_{\max}}{E}$ (Sect. 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-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 (Sect. 3.2).
- We analyze the performance of online algorithms in non-idling and non-preemptive execution settings, separately (Sect. 4). We show that *EC-EDF* remains optimal in non-idling settings (Sect. 4.1), even when preemption is not allowed. However, in Sect. 4.2, we demonstrate that in non-preemptive execution settings, there cannot exist a competitive algorithm regardless of the $\frac{e_{\max}}{E}$ ratio if the online algorithm cannot idle.
- We extend our competitive analysis to more general settings with non-uniform value densities where the *importance ratio* $k > 1$ (Sect. 5). We show that, in this case, no online algorithm can achieve a competitive factor greater than $\frac{1}{k^{\frac{e_{\max}}{E}}}$. We also derive the competitive factors of *EC-EDF* and *EC-EDF** algorithms in non-uniform value density settings.
- We consider the settings where the processor has the *Dynamic Voltage Scaling* (*DVS*) capability (Sect. 6). We provide a semi-online algorithm *EC-DVS** which uses the additional knowledge of the largest job size in the *actual* input instance and the *maximum loading factor* β (Baruah et al. 1990; Jeffay and Stone 1993). It is shown that the competitive factor of *EC-DVS** is 0.5 times that of the optimal semi-online algorithm in these settings.
- We also analyze the *EC-EDF* algorithm within the context of the *resource augmentation* technique (Phillips et al. 2002; Kalyanasundaram and Pruhs 1995) and characterize the conditions under which *EC-EDF* can achieve a competitive factor of 1, with *extra* energy or DVS feature (Sect. 7). Finally, we comment on the relationship between the problem considered in this paper and the well-known *knapsack* problem (Garey and Johnson 1990) (Sect. 8).

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 Table 1.

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

Table 1

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	$\frac{1}{(k_{\max})^{\frac{e_{\max}}{E}}}$	$\frac{1}{2k_{\max}}$

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 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 considered negligible (recent processors indeed have very low energy consumption figures in *Deep Sleep* or *Halt* states).

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, as in AlEnawy and Aydin (2004, 2005) 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 Sect. 1, as opposed to the well-studied online *overloaded* real-time scheduling settings (e.g. Baruah et al. 1991a; Koren and Shasha 1992) 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 when there is sufficient time but no sufficient 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 (Baruah et al. 1991a; Koren and Shasha 1992). We consider the *uniform value density* model (Baruah et al. 1991a, 1991b; Baruah and Haritsa 1997; Koren and Shasha 1992), where the job's value is equal to its size (e_i). We also extend our results (in Sect. 5) 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 of uniform value density settings where all jobs have the same release times and deadlines by a reduction through the subset-sum problem which is known to be NP-Hard (Garey and Johnson 1990).

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 a job 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. The information about $e_{\max}$ can be derived from the workload characteristics or from the addressing capabilities of the underlying hardware.

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] Similar to $e_{\max}$, the exact value of δ can be derived from the workload and/or system characteristics. 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 introduce a specific job arrival pattern that is proven very useful in deriving competitive factors in online real-time scheduling (Baruah et al. 1991a; Koren and Shasha 1992). 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* was crucial in obtaining the well-known competitive factor bound of 0.25 for overloaded real-time systems (Baruah et al. 1991a; Koren and Shasha 1992).

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.

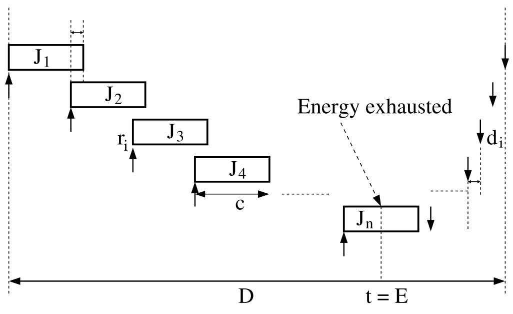
Proof Follows from the optimality of preemptive EDF in underloaded real-time environments (Dertouzos 1974). □

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 is non-competitive* (*i.e. it cannot guarantee a non-zero competitive factor*).

[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.

Fig. 1 The worst-case instance for preemptive *EDF*



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 $c \le e_{\max}$ 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 $n \ge 1$, 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 of δ, as can be seen 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 it 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 (Baruah et al. 1991b; Borodin and El-Yavin 1998; Koren and Shasha 1992; Pruhs et al. 2004). 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 that of 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 the maximum 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 1 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 Sect. 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 the 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 each. 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 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 $\mathcal{C}$ as 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 Fig. 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. □

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

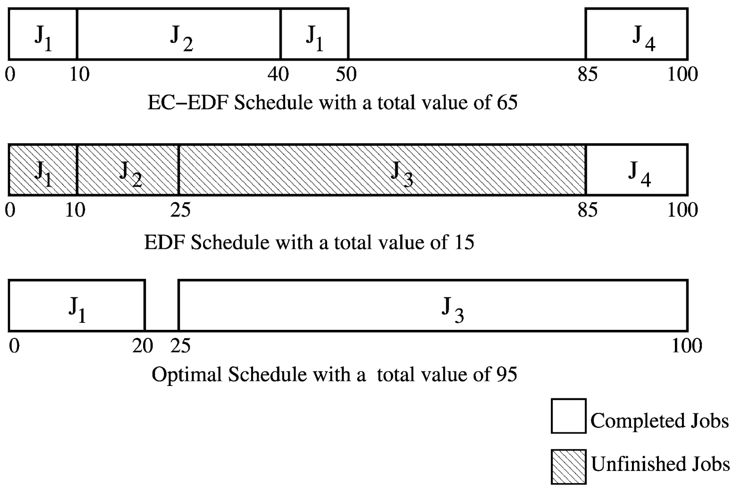

J1
J2
J1
J4
0
10
40
50
85
100
EC–EDF Schedule with a total value of 65
J1
J2
J3
J4
0
10
25
85
100
EDF Schedule with a total value of 15
J1
J3
0
20
25
100
Optimal Schedule with a total value of 95
Completed Jobs
Unfinished Jobs

Fig. 2 Schedules Generated by *EC-EDF*, *EDF* and *Optimal*

Proof First, note that if $E_b \le E$, *EC-EDF* reduces to the conventional 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, let ψ' denote the set of jobs admitted by *EC-EDF* before discarding J_d and let E_d denote the total workload (and energy demand) of the jobs in ψ'. From Proposition 2, *EC-EDF* is guaranteed to complete the job set ψ' in a timely manner without violating the system energy budget limits. Thus, *EC-EDF* guarantees a total value of E_d.

As a consequence of J_d being discarded at time t, we have $E_d + e_d > E$. Since $e_d \le e_{\max}$, this implies $E_d > E - e_{\max}$. Thus, the total value obtained by *EC-EDF* is at least $E - e_{\max}$, while the adversary can make at most a value of E. We conclude that *EC-EDF* has a competitive factor of $\frac{E-e_{\max}}{E}$. □

We also underline that in settings where $E \ge E_b$, *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 Sect. 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 (Ebenlendr and Sgall 2009; Pruhs et al. 2004). 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 (Ebenlendr and Sgall 2009; Pruhs et al. 2004).

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-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-EDF**, demonstrating its optimality.

Let $e_l \le E$ represent the largest job size in the actual input sequence. We first give the rules for *EC-EDF**. *EC-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-EDF** compares e_l to the available energy budget E. If $e_l > \frac{E}{2}$, *EC-EDF** simply waits for the largest size job, e_l, and executes it. Since $E \ge e_l > \frac{E}{2}$, the value of *EC-EDF** is no less than $\frac{E}{2} + \delta$. The adversary can gather a value of at most E. On the other hand, if $e_l \le \frac{E}{2}$, *EC-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 $e_l \le \frac{E}{2}$, the lower bound on competitive factor of *EC-EDF* and hence that of *EC-EDF** reduces to 0.5. Thus, in either case, *EC-EDF** makes at least half of the value of the adversary.

Lemma 2 *EC-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. Let $E = k_1 \cdot \delta$ and $e_l = k_2 \cdot \delta$. Where, k_1 and k_2 are positive integers such that $k_1 \ge k_2$. The proof is similar to that of Theorem 1. The adversary first releases $(k_1 - k_2 + 1)$ *sequential jobs* of size δ (note that here $k_2 = \frac{e_l}{\delta}$ and not $\frac{e_{\max}}{\delta}$ as in Theorem 1). As mentioned in proof of Theorem 1, let m be the number of jobs *not executed* by the online algorithm $\mathcal{A}$. We again have 3 cases.

Cases 1, 2 and 3B are similar to those in Theorem 1, except that at the very end the adversary now releases a job of size e_l instead of $e_{\max}$. Therefore, repeating the corresponding arguments from Theorem 1, the competitive factor of $\mathcal{A}$ for these cases is given by $\frac{E-e_l}{E}$.

Case 3A corresponds to k_2 jobs of size δ having been released *and* $\mathcal{A}$ having executed $0 \le m' < m$ of these jobs. Repeating the corresponding arguments of Theorem 1, we note that $\mathcal{A}$ has accrued a value of $(k_1 - k_2 + 1 + (m' - m)) \cdot \delta$ and has a remaining energy budget of $(k_2 - (m' - m + 1)) \cdot \delta$, while the adversary has accrued a total value of E and has no energy budget left. Also, observe that in the input sequence leading to and terminating in Case 3A, a job of size e_l has not been released by the adversary. Thus, at the end of Case 3A, the adversary releases a job of size e_l which $\mathcal{A}$ can execute provided it has enough energy budget left to execute it (i.e. if $(m' - m + 1) \leq 0$). Thus, the *maximum* value that $\mathcal{A}$ can accrue is $e_l + (k_1 - k_2 + 1 + (m' - m)) \cdot \delta$ and its competitive factor is no better than

$$\frac{e_l + (k_1 - k_2 + 1 + (m' - m)) \cdot \delta}{E} \geq \frac{e_l}{E}$$

From the above 3 cases the competitive factor of $\mathcal{A}$ can be expressed as $\max(\frac{E-e_l}{E}, \frac{e_l}{E})$. Since, $0 \leq e_l \leq E$, we have $\max(\frac{E-e_l}{E}, \frac{e_l}{E}) \geq 0.5$. 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 (Baruah et al. 1991a) and further, there exists an algorithm D^{over} with this competitive factor (Koren and Shasha 1992). 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-EDF** achieves this competitive factor.

4 Competitive analysis for non-idling and non-preemptive scheduling algorithms

Our main results in preceding sections were derived under the assumption that preemption was allowed and that, if needed, the online algorithm could leave the CPU idle in the presence of ready jobs. In this section, we investigate the implications of relaxing these assumptions, first separately and then simultaneously. Note that, in the analysis of each of these sub-models, we still have the *underloaded* energy-constrained real-time system assumption. In other words, we assume that there exists a feasible schedule for the input task set, on the target (i.e. non-idling and/or non-preemptive) execution settings.

4.1 Non-idling execution settings

In real-time scheduling theory, a scheduling algorithm is said to be idling at time t, if there is a pending job and the algorithm is not executing any job. The algorithms that never idle (i.e. *non-idling* algorithms, or, sometimes called *work-conserving* algorithms) have practical importance and were studied in the literature (Jeffay et al. 1991; Liu 2000). We now undertake a basic competitive analysis of *non-idling* online real-time scheduling algorithms in energy-constrained settings.

Note that the traditional definition of *idling*, as given above, is not quite adequate for our energy-constrained settings, as an algorithm should not be forced to execute a pending job requiring more than the remaining energy, or waste energy by executing a job that is guaranteed to miss its deadline. Thus, in an effort to address non-idling scheduling issues, we first formally re-define the *idling* concept in energy-constrained settings.

Definition 1 In energy constrained settings, an algorithm is considered to be idling if and only if all of the following four conditions hold at a specific time instant t:

(1) The processor is not executing any job,
(2) There is a pending (ready) job J_i with remaining execution time/energy requirement $e_i^r > 0$,
(3) $e_i^r \le d_i - t$,
(4) $e_i^r \le E^r$, where E^r is the system's remaining energy budget.

Given this definition, a non-idling algorithm is one that does not idle at any time t. Observe that, according to these definitions, *EC-EDF* is a non-idling algorithm, while *EC-EDF** is an idling algorithm. Section 3.1 already shows that *EC-EDF* is the best non-idling algorithm that can be developed against a potentially idling adversary.

The power of idling for the adversary was crucial in deriving the competitive factor bound in Theorem 1. The main motivation behind this subsection is to investigate if there can exist an non-idling algorithm better than *EC-EDF* against an adversary that cannot idle either. Hence, as opposed to the previous sections, the adversary cannot idle in the analysis presented in this subsection.

Lemma 4 *No non-idling online algorithm can achieve a competitive factor greater than $\frac{E-e_{\max}}{E}$ against a non-idling adversary.*

Proof Let $E = k_1 \cdot \delta$ and $e_{\max} = k_2 \cdot \delta$. Where, k_1 and k_2 are positive integers such that $k_1 \ge k_2$. In the execution scenario that will establish the bound, the adversary first introduces $(k_1 - k_2)$ *sequential jobs* each of size δ. By executing these jobs, at time $t = (k_1 - k_2) \cdot \delta$, both the online algorithm $\mathcal{A}$ and the adversary have accrued a total value of $(k_1 - k_2) \cdot \delta = E - e_{\max}$ and have a remaining energy budget of $e_{\max}$ units. At time t, the adversary introduces the following 2 jobs: $J_1(t, \delta, t + \delta)$ and $J_2(t, 2\delta, t + 5\delta)$. $\mathcal{A}$ has two choices.

Case 1: $\mathcal{A}$ executes J_1. In this case, the adversary executes J_2 and waits until $t_1 = t + 5\delta$. Notice that since $\mathcal{A}$ scheduled J_1, it will have to also schedule J_2 after finishing J_1, because it cannot idle. On the other hand by executing J_2, the adversary has effectively created an idle interval for itself by missing the deadline of J_1. At t_1, $\mathcal{A}$ has accrued a value of $E - e_{\max} + 3\delta$ and has a remaining energy budget of $e_{\max} - 3\delta$ units. On the other hand, at time t_1, the adversary has accrued a value of $E - e_{\max} + 2\delta$ and has a remaining energy budget of $e_{\max} - 2\delta$ units. At t_1, the adversary introduces a job with size $e_{\max} - 2\delta$ that $\mathcal{A}$ cannot execute. By executing this job the adversary accrues a total value of E. The competitive factor is $\frac{E-e_{\max}}{E}$.

Case 2: $\mathcal{A}$ executes J_2. In this case, the adversary executes J_1 followed by J_2. When $\mathcal{A}$ finishes J_2 at time $t + 2\delta$, the adversary has finished executing only half of J_2. At this stage the adversary introduces job $J_3(t + 2\delta, 2\delta, t + 4\delta)$. Note that, $\mathcal{A}$ is forced to execute J_3 (as it cannot idle). The adversary, by continuing with execution of J_2, misses J_3 and again creates an idle period for itself from $t + 3\delta$ to $t + 5\delta$.

At time $t_1 = t + 5\delta$, $\mathcal{A}$ has accrued a value of $E - e_{\max} + 4\delta$ and has a remaining energy budget of $e_{\max} - 4\delta$ units. On the other hand, at time t_1, the adversary has accrued a value of $E - e_{\max} + 3\delta$ and has a remaining energy budget of $e_{\max} - 3\delta$ units. At t_1, the adversary introduces a job with size $e_{\max} - 3\delta$ that $\mathcal{A}$ cannot execute. By executing this job the adversary accrues a total value of E. The competitive factor is $\frac{E-e_{\max}}{E}$. □

In the non-idling model, the knowledge of the largest job size in the actual input sequence does not help as the algorithm $\mathcal{A}$ cannot idle and wait for it, which was crucial for achieving a competitive idling semi-online algorithm *EC-EDF**. In the non-idling model, the adversary can introduce jobs such that when the largest size job is released, the remaining energy budget of $\mathcal{A}$ is not sufficient to execute it.

Corollary 2 EC-EDF *is also optimal against an adversary that cannot idle.*

4.2 Non-preemptive execution settings

Non-preemptive execution has several attractive features, such as ease of implementation, low run-time overhead, implicit exclusive access to shared resources, among others (Liu 2000). As such, non-preemptive scheduling algorithms have been studied in literature and in the realm of real-time scheduling (Jeffay et al. 1991; Liu 2000). In general, non-preemptive hard real-time scheduling is known to be NP-Hard in the strong sense, even in *off-line* settings (Jeffay et al. 1991). In this subsection, we consider non-preemptive *online* scheduling in energy-constrained settings and we derive a basic result for settings where *both* the online algorithm $\mathcal{A}$ and the adversary schedule jobs through non-preemptive algorithms. In accordance with the underloaded real-time system assumption, we assume that there exists a feasible non-preemptive scheduling for the input task set, if the energy constraint is not considered.

Lemma 5 *There does not exist a competitive non-preemptive online algorithm, irrespective of the ratio* $\frac{e_{\max}}{E}$.

Proof In the scenario that enforces this upper bound, the adversary introduces a job of size 2δ and deadline $e_{\max} + 2\delta$. If the online algorithm $\mathcal{A}$ idles until $t = e_{\max} + 2\delta$ and skips this job, the adversary continues the pattern. $\mathcal{A}$ will have to execute one of these jobs before the cumulative workload released by the adversary equals or exceeds E. When $\mathcal{A}$ executes a job (say at time t), the adversary releases a job of size $e_{\max}$ with zero laxity at time $t + \delta$. The non-preemptive algorithm $\mathcal{A}$ gathers a value of 2δ, while the adversary gathers a value of $e_{\max}$. By repeating the same pattern a finite number of times, the algorithm $\mathcal{A}$ can be made technically non-competitive, regardless of the ratio $\frac{e_{\max}}{E}$. □

Notice that with non-preemption, while the online algorithm is executing a job of size X for execution, the adversary can release a job of size $Y \gg X$ and no laxity, causing excessive value loss for the algorithm. This was the key to the instance given in the proof of Lemma 5. As a consequence, with the knowledge of the largest job size (say e_l) in the input instance, no (potentially) idling non-preemptive semi-online algorithm can achieve a competitive factor greater than $\frac{e_l}{E}$ and the algorithm that achieves this bound simply waits for the largest job with size e_l. The ratio $\frac{e_l}{E}$ may be arbitrarily small if $e_l < \frac{E}{2}$. However, if $e_l \geq \frac{E}{2}$, this semi-online algorithm has a competitive factor of 0.5.

4.3 Non-idling and non-preemptive execution settings

In this section, we consider the settings where the scheduling must be both non-preemptive and non-idling. In one of the most cited studies (Jeffay et al. 1991), non-idling EDF is shown to be optimal among all non-idling non-preemptive scheduling policies, in the sense that it can generate a feasible schedule whenever it is possible to do so. Lemma 6 below establishes a fundamental result for non-preemptive and non-idling settings. Once again, the underloaded system assumption implies that the workload under consideration can be scheduled in feasible manner on non-preemptive and non-idling settings. The optimality of the non-preemptive EDF (Jeffay et al. 1991) (which is non-idling by definition) implies that it can schedule the input job set (or, any subset thereof) in feasible manner when the energy constraint is not considered.

Lemma 6 *No non-preemptive non-idling online algorithm can achieve a competitive factor greater than* $\frac{E-e_{\max}}{E}$ *and non-preemptive* EC-EDF *achieves this competitive factor.*

Proof The proof is very similar to that of Lemma 4. However, in Case 2 where $\mathcal{A}$ executes J_2, introduce the third job J_3 at time $t+\delta$ (i.e. $J_3(t+\delta, 2\delta, t+4\delta)$). Observe that this allows feasibility of J_1, J_2 and J_3 under non-preemptive non-idling scheduling. Now, by repeating the arguments of Lemma 4 the upper bound on performance of any online non-preemptive and non-idling algorithm can be found as $\frac{E-e_{\max}}{E}$. Since non-preemptive non-idling EDF is an optimal real-time scheduling algorithm in the sense that it can generate a feasible schedule whenever another non-preemptive non-idling algorithm can do so (Jeffay et al. 1991), the non-preemptive *EC-EDF* is found as the optimal online non-idling non-preemptive algorithm (in terms of the competitive factor it achieves), in energy-constrained settings. □

It is also straightforward to verify that the additional knowledge of the largest job size in the actual input sequence does not help design better non-preemptive non-idling online algorithms.

5 Non-uniform value densities

In the settings of this paper, 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* (Baruah et al. 1991a; Koren and Shasha 1992). In uniform density settings (Sect. 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 $\frac{1}{(k_{\max})^{\frac{e_{\max}}{E}}}$.*

Proof See Appendix 2 for a full proof. □

We note that the upper bound established by Theorem 3 is not tight when the ratio $\frac{k_{\max}}{k_{\min}}$ is close to one. This can be verified by setting $k_{\max} = 1 + \Delta \approx 1$, where Δ is a positive number arbitrarily close to zero, and repeating the input instance given in the proof of Theorem 1 with the following two changes: (1) All jobs released have value density $k_{\max}$. (2) At the very end, in every execution scenario, the adversary releases a single job of size δ with value density $k_{\min} = 1$. In this specific modified sequence the upper bound can easily be verified to be $\frac{k_{\max}(E-e_{\max})}{k_{\max}E} = \frac{E-e_{\max}}{E}$, while for the given parameters the upper bound suggested by Theorem 3 is close to 1, irrespective of the ratio $\frac{e_{\max}}{E}$. However, as $k_{\max}$ increases the upper bound established by Theorem 3 decreases in exponential fashion. Investigating the tightness of the upper bound established by Theorem 3 when the ratio $\frac{k_{\max}}{k_{\min}}$ is large is an open problem.

We also 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. This can be verified by the job release pattern given in the proof of Theorem 3. Thus, the a priori knowledge of the largest job size does not help design better online algorithms in non-uniform value density settings. Lemma 7 establishes the competitive factor of the semi-online algorithm, *EC-EDF**, which uses the additional knowledge of largest job size.

Lemma 7 *Algorithm EC-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-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-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 $\frac{1}{2k_{\max}}$. □

It is also interesting to consider the upper bound on the competitive factor of any non-idling algorithm (compared against an idling adversary) in these settings. Theorem 4 below establishes this bound.

Theorem 4 *No non-idling online algorithm can achieve a competitive factor greater than* $\min(\frac{E-e_{\max}}{E+(k_{\max}-1)e_{\max}}, \frac{1}{2k_{\max}})$ *against a potentially idling adversary.*

Proof We define two constants $c_1 = \lfloor \frac{E}{e_{\max}} \rfloor$ and $c_2 = E - (c_1 \cdot e_{\max})$. First, the adversary introduces a workload W_1 of $E - e_{\max} + \delta$ units using $n \geq 1$ jobs, each with value density $k_{\min} = 1$. All jobs in W_1 have zero laxity and are released back to back, one at a time (i.e. the release time of the ith job coincides with the deadline of the $(i-1)$th job and the deadline of the ith job is its release time plus execution time). At least one job in W_1 will be of size exactly c_2. At the end of the workload W_1 the non-idling algorithm $\mathcal{A}$ has exhausted $E - e_{\max} + \delta$ units of energy and gathered a value of $E - e_{\max} + \delta$. Now, we distinguish two cases.

Case 1: Assume $e_{\max} \leq \frac{E}{2}$
At this stage, the adversary introduces another workload W_2 of $c_1 e_{\max}$ units using c_1 jobs of size $e_{\max}$ and value density $k_{\max}$ (as with W_1 all jobs in W_2 have zero laxity and are released back to back). $\mathcal{A}$ cannot execute any job in W_2 irrespective of the job deadlines. The adversary gathers a value of $c_1 k_{\max} e_{\max} + c_2$ by executing all jobs in workload W_2 and the job of size c_2 in workload W_1. The value of $\mathcal{A}$ is $E - e_{\max} + \delta$. If $c_2 = 0$, the competitive factor is $\frac{E-e_{\max}}{k_{\max}E}$. If $c_2 \neq 0$, by setting $c_2 = \delta$ the competitive factor is minimized and in that case, is bounded by $\frac{E-e_{\max}}{k_{\max}E}$. Thus, either way the competitive factor is no more than $\frac{1}{2k_{\max}}$ (obtained by setting $e_{\max} = \frac{E}{2}$).

Case 2: Assume $e_{\max} > \frac{E}{2}$
In this case $c_1 = 1$ and $c_2 = E - e_{\max}$. The adversary has two different strategies to reduce the competitive factor as much as possible depending on the value of $k_{\max}$ (similar to Case 1 above). First, the adversary can follow the *exact* same sequence given in Case 1 above by restricting the largest job size in the actual instance to $\frac{E}{2}$. In this case the competitive factor as explained in Case 1 would be $\frac{1}{2k_{\max}}$.

Second, the adversary can introduce another workload, W_2, of $c_1 e_{\max}$ units using c_1 jobs of size $e_{\max}$ and value density $k_{\max}$. $\mathcal{A}$ cannot execute any job in W_2 irrespective of their deadlines. The adversary gathers a value of $c_1 k_{\max} e_{\max} + c_2$ by executing all jobs in workload W_2 and the job of size c_2 in workload W_1. The competitive factor would then be $\frac{E-e_{\max}}{k_{\max}e_{\max}+E-e_{\max}}$.

From Cases 1 and 2 one can conclude that the competitive factor is constrained either by $\frac{1}{2k_{\max}}$ or by $\frac{E-e_{\max}}{k_{\max}e_{\max}+E-e_{\max}}$. Further, since the ordering between these two terms is dependent on the values of $e_{\max}$ and $k_{\max}$, no non-idling algorithm can achieve a competitive factor greater than $\min(\frac{E-e_{\max}}{E+(k_{\max}-1)e_{\max}}, \frac{1}{2k_{\max}})$. □

We now show that *EC-EDF* is the optimal non-idling algorithm in non-uniform value density settings as well.

Lemma 8 EC-EDF *has a competitive factor of* $\min(\frac{E-e_{\max}}{E+(k_{\max}-1)e_{\max}}, \frac{1}{2k_{\max}})$.

Proof Let t represent the time instance at which *EC-EDF* discards a job for the first time. Let the execution requirement of this job discarded by *EC-EDF* be e_d. Since $e_d \leq e_{\max}$, at time t, *EC-EDF* has accepted a workload of at least $E - e_d + \delta$ and from Proposition 2, *EC-EDF* is guaranteed to make a value of at least:

$$V = k_{\min}(E - e_d + \delta) = E - e_d + \delta$$

Thus, the total value accrued by *EC-EDF* is $V + \Delta V$, where $\Delta V \geq 0$.

If V^* represents the total value accrued by the adversary then $0 \leq V^* \leq k_{\max} E$. Thus the competitive factor can be defined as $\frac{V+\Delta V}{V^*}$. However, in non-uniform value density settings, it is not necessary that ΔV is minimized (i.e. $\Delta V = 0$) when V^* is maximized (i.e. $V^* = k_{\max} E$); which was the case in uniform value density settings. Thus, when jobs have different value densities, there is a non-trivial relation between V^* and ΔV. We will distinguish two cases.

Case 1: $0 < e_d \leq \frac{E}{2}$
In this case the maximum value the adversary can make without increasing the value accrued by *EC-EDF* beyond V is $k_{\max} E$; which is also the maximum value the adversary can make in non-uniform value density settings. This can be achieved by releasing a workload consisting of jobs with execution requirement e_d and value density $k_{\max}$ after time t. While *EC-EDF* has not enough energy left to execute any of these jobs, the adversary can execute all these jobs making a value of at most $k_{\max} E$. Thus, in this case $\Delta V = 0$ and $V^* = k_{\max} E$ giving a competitive factor of $\frac{E-e_d}{k_{\max} E}$ which is minimized at $e_d = \frac{E}{2}$ to $\frac{1}{2k_{\max}}$.

Case 2: $e_{\max} \geq e_d > \frac{E}{2}$
In this case, the maximum value the adversary can make *without increasing* the value accrued by *EC-EDF* beyond V is:

$$V_1^* = E - e_d + \delta + k_{\max} e_d < k_{\max} E$$

In this case, the competitive factor is minimized at $e_d = e_{\max}$ and is given by

$$\frac{E - e_{\max}}{E + (k_{\max} - 1) e_{\max}}$$

On the other hand, if the adversary intends to make a value $V_2^* > V_1^*$ then $\Delta V > 0$ and the following observation holds.

Observe that, if V_2^* is maximized (i.e. $V_2^* = k_{\max} E$) then $\Delta V \geq (2e_d - E)$. As a consequence, if $V_2^* = k_{\max} E$ then $V + \Delta V \geq e_d + \delta$. In this case, the competitive factor is minimized at $e_d = \frac{E}{2} + \delta$ and is given by $\frac{1}{2k_{\max}}$.

From Cases 1 and 2, the competitive factor of *EC-EDF* is:

$$\min\left(\frac{E - e_{\max}}{E + (k_{\max} - 1) e_{\max}}, \frac{1}{2k_{\max}}\right) \qquad \square$$

Corollary 3 *EC-EDF is an optimal non-idling algorithm in non-uniform value settings.*

We conclude this subsection with the following remarks.

- The bound of $(k_{\max})^{-\frac{e_{\max}}{E}}$ holds in general for any online algorithm (idling or non-idling). The knowledge of the largest job size does not help improve the upper bound on competitive factor achievable by any online algorithm.
- Again, 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 upper bound established in Theorem 3 is not tight when the ratio $\frac{k_{\max}}{k_{\min}}$ is close to one. For larger values of $\frac{k_{\max}}{k_{\min}}$, investigating the tightness of this upper bound is an interesting open problem that deserves further research.

6 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 (Aydin et al. 2004; Lee and Shin 2004; Pillai and Shin 2001). DVS-capable processors can play a critical role also in energy-constrained real-time systems. Using effective DVS policies that 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.

There have been a few research efforts on competitive analysis of DVS-based systems (Bansal et al. 2004; Chan et al. 2007; Yao et al. 1995). These efforts deal with the energy minimization problem of hard real-time sporadic jobs (without the hard energy constraint) where the objective is to minimize energy consumption subject to meeting the temporal constraints. Same studies assume no upper bound on the maximum CPU clock frequency. In contrast, our work considers soft real-time jobs running on a DVS-enabled system (with minimum and maximum frequency limits) and a fixed energy budget. Our objective is to maximize the total value of jobs that meet their deadlines, subject to the system's energy budget constraint.

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.

[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 (Aydin et al. 2006). Our upper bound results can easily be adapted to these system-level energy models by enforcing arbitrarily low energy-efficient speed levels.

In line with the previous sections we still make the *underloaded energy-constrained system assumption* according to which, in DVS settings, there exists a feasible schedule for all jobs in ψ, when the processor executes all jobs at frequency $f_{\max}$ and the energy constraints are not taken into account.

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* $f_{\max}$ (also referred to as *workload* of J_i), 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 analysis.

Definition 2 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 3 The absolute effective loading factor (or simply the loading factor) β is the maximum effective loading factor over all intervals $[t_1, t_2]$: $\beta = \max(h(t_1, t_2))$, $0 \le t_1 < t_2$.

Theorem 5 (from Baruah et al. 1990; Jeffay and Stone 1993) *A set of real-time jobs can be scheduled in feasible manner* (*by preemptive EDF*) *if and only if* $\beta \le 1$.

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 (Yao et al. 1995). 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* $f \ge \max(f_{\min}, \beta)$ *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 constant 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 6 characterizes this result.

Theorem 6 *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 $\mathcal{A}$ 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 $J_2(E, \frac{E}{k^{\alpha-1}}, \frac{E}{k^{\alpha}}, E)$, where $f_{\min} \leq k \leq 1$. 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 value 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 $(f_{\min})^{\alpha-1}$ on 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 (Sect. 3), where both the online algorithm and the adversary had to process jobs at the same (constant) speed. Thus, all the results of Sect. 3 apply.

As a consequence of Theorem 6, the upper bound on competitive factor approaches zero as $f_{\min} \to 0$ and $\beta \to 1$.

6.1 Semi-online algorithm *EC-DVS**

In this subsection we present a semi-online algorithm *EC-DVS** that uses the knowledge of both β and the maximum job size in the input instance. *EC-DVS** is a variant of *EC-EDF** for DVS settings where $\beta > f_{\min}$. First, we will describe a variant of *EC-EDF* called *EC-DVS* which will later be used by *EC-DVS**.

The *EC-DVS* algorithm uses an admission test similar to that of *EC-EDF* and admits a new job to the system *if and only if* there is enough remaining energy to completely execute *both* the new job and the *remaining* workload from all pending admitted jobs. All admitted jobs are executed in EDF order and at a constant speed β.

Observe that in executing a workload of W units, *EC-DVS* spends $\beta^{\alpha-1} \cdot W$ units of energy. Thus, formally, at time t, when the system has remaining energy budget of E^r units and remaining workload of R units from *all* pending admitted jobs, *EC-DVS* admits a newly arriving job $J_i(t, c_i, D_i, e_i)$ *if and only if*

$$E^r \geq \beta^{\alpha-1} \cdot (c_i + R)$$

Lemma 9 *If the largest job size in the input instance c_l is such that $c_l \leq \frac{1}{2}(\frac{E}{\beta^{\alpha-1}})$ then* EC-DVS *has a competitive factor of $\frac{1}{2}(\frac{f_{\min}}{\beta})^{\alpha-1}$.*

Proof If *EC-DVS* rejects a job J_i with execution time c_i, then it implies that *EC-DVS* has depleted (or will eventually deplete upon completely executing the remaining workload of pending admitted jobs) at least $E' = E - \beta^{\alpha-1}c_i + \delta$ units of energy. In other words, *EC-DVS* has admitted a workload of at least $\frac{E'}{\beta^{\alpha-1}}$ units.

The following property of *EC-DVS* can be easily verified in the lines of Proposition 2: *EC-DVS* executes all admitted jobs to completion before their respective deadlines without violating the system's energy constraints. Thus, *EC-DVS* guarantees a value of at least $\frac{E'}{\beta^{\alpha-1}}$. Since $c_i \leq c_l \leq \frac{1}{2}(\frac{E}{\beta^{\alpha-1}})$, we have:

$$\frac{E'}{\beta^{\alpha-1}} = \frac{E - \beta^{\alpha-1}c_i + \delta}{\beta^{\alpha-1}} \geq \frac{E - \beta^{\alpha-1}c_l + \delta}{\beta^{\alpha-1}}$$

$$\frac{E'}{\beta^{\alpha-1}} \geq \frac{E - (\beta^{\alpha-1} \cdot \frac{1}{2}(\frac{E}{\beta^{\alpha-1}})) + \delta}{\beta^{\alpha-1}}$$

$$\frac{E'}{\beta^{\alpha-1}} \geq \frac{1}{2}\left(\frac{E}{\beta^{\alpha-1}}\right)$$

Since the optimal algorithm can make a value of at most $\frac{E}{f_{\min}^{\alpha-1}}$, the competitive factor of *EC-DVS* is $\frac{1}{2}(\frac{f_{\min}}{\beta})^{\alpha-1}$. □

The *EC-DVS** algorithm is based on *EC-DVS* and has the following rules. If the maximum job size in the input instance c_l is such that $c_l > \frac{1}{2}(\frac{E}{\beta^{\alpha-1}})$ then *EC-DVS** just waits for the job J_l with execution time c_l and executes it at speed $f = \frac{c_l}{D_l}$, where D_l is the deadline of J_l. Since the processor cannot run at a frequency lower than $f_{\min}$ and the minimum execution requirement of any job cannot exceed the system energy budget E, f is guaranteed to be in the range $[f_{\min}, \beta]$. On the other hand, if $c_l \leq \frac{1}{2}(\frac{E}{\beta^{\alpha-1}})$ then *EC-DVS** follows the rules of *EC-DVS*.

Lemma 10 *EC-DVS* has a competitive factor of $\frac{1}{2}(\frac{f_{\min}}{\beta})^{\alpha-1}$.*

Proof Let c_l denote the maximum job size in the input instance.

Case 1: If $c_l > \frac{1}{2}(\frac{E}{\beta^{\alpha-1}})$ then *EC-DVS** makes a value of at least c_l while the optimal online algorithm can make at most $\frac{E}{f_{\min}^{\alpha-1}}$. Thus, the competitive factor of *EC-DVS** is $\frac{1}{2}(\frac{f_{\min}}{\beta})^{\alpha-1}$.

Case 2: If $c_l \le \frac{1}{2}(\frac{E}{\beta^{\alpha-1}})$ then *EC-DVS** follows *EC-DVS* and thus has a competitive factor of $\frac{1}{2}(\frac{f_{\min}}{\beta})^{\alpha-1}$ (Lemma 9).

From Cases 1 and 2, *EC-DVS** has a competitive factor of $\frac{1}{2}(\frac{f_{\min}}{\beta})^{\alpha-1}$. □

Lemma 11 below establishes the upper bound on the best achievable competitive factor by any semi-online algorithm that uses only the knowledge of β and the largest job size in the input instance.

Lemma 11 *No semi-online algorithm with the knowledge of β and the largest job size in the input instance can achieve a competitive factor greater than $(\frac{f_{\min}}{\beta})^{\alpha-1}$.*

Proof The adversary constructs an instance of back-to-back jobs (J_{i+1} released at the deadline of J_i) with the following parameters: $c = \delta$, $D = \frac{\delta}{f_{\min}^{\alpha-1}}$ and $e = f_{\min}^{\alpha-1} \cdot \delta$. This sequence of jobs ends either when the total workload released by the adversary reaches $\frac{E}{f_{\min}^{\alpha-1}}$ or $\mathcal{A}$ executes one of these jobs at time $t < \frac{E}{f_{\min}^{\alpha-1}}$. We will consider both of these cases separately. Let c_l denote the maximum job size in the input instance.

- Case A: The total workload released by the adversary reaches $\frac{E}{f_{\min}^{\alpha-1}}$. In this case, the adversary introduces job $J(\frac{E}{f_{\min}^{\alpha-1}}, c_l, \frac{c_l}{\beta}, \beta^{\alpha-1}c_l)$. Observe that job J ensures the loading factor of the input instance created by the adversary is β. Since the online algorithm $\mathcal{A}$ has not executed any job (and hence not depleted any energy budget) until the release time of job J, it can execute job J and accrue a value no more than c_l. On the other hand, executing all jobs except job J, the adversary makes a total value of $\frac{E}{f_{\min}^{\alpha-1}}$. Thus, the competitive factor is $f_{\min}^{\alpha-1} \cdot \frac{c_l}{E}$. Since the minimum energy requirement of c_l must not exceed E we have: $\beta^{\alpha-1}c_l \le E$. Thus, setting $c_l = \frac{E}{\beta^{\alpha-1}}$ the competitive factor is bounded by $(\frac{f_{\min}}{\beta})^{\alpha-1}$.
- Case B: $\mathcal{A}$ executes one of the jobs at time $t < \frac{E}{f_{\min}^{\alpha-1}}$. In this case, $\mathcal{A}$ has depleted $f_{\min}^{\alpha-1} \cdot \delta$ units of energy and accrued a value of δ. At this stage the adversary introduces job $J(t, c_l, \frac{c_l}{\beta}, E)$. Observe again that job J ensures the instance created by the adversary has a loading factor β. By executing J at frequency β the adversary makes a value of c_l. The competitive factor is $\frac{\delta}{c_l}$, where $c_l \le \frac{E}{\beta^{\alpha-1}}$. Thus the competitive factor in this case can be arbitrarily small.

From Cases A and B above we conclude that no semi-online algorithm with the knowledge of β and the largest job size in the input instance can achieve a competitive factor greater than $(\frac{f_{\min}}{\beta})^{\alpha-1}$. □

Corollary 4 *The competitive factor of EC-DVS* is* 0.5 *times that of the optimal semi-online algorithm in these settings.*

7 Resource augmentation

While 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 Phillips et al. (2002) and popularized by Kalyanasundaram and Pruhs (1995), 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 (Kalyanasundaram and Pruhs 1995), or it may have access to additional CPUs (Baruah 1998). 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* $(E + e_{\max})$ *units of energy.*

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

Corollary 5 *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 (Kalyanasundaram and Pruhs 1995), 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 6 *If* $\beta \leq (\frac{1}{2})^{\frac{1}{\alpha-1}}$, β-EC-EDF *is as powerful as a clairvoyant adversary without DVS.*

8 Relationship to the knapsack problem

While inspired by existing articles on the competitive analysis of real-time scheduling algorithms under *overload*, our study has introduced the novel dimension of hard energy constraint for competitive analysis of *underloaded* real-time systems. However, a careful look suggests some interesting similarities between energy-constrained real-time scheduling (ECRTS) and the well-known knapsack (KNP) problems. In this section, we investigate these similarities and clarify the unique characteristics of our problem. We first formally define the knapsack problem:

Knapsack Problem (KNP) Given a knapsack of capacity W, a set K of n items each with weight w_i and value v_i, select a subset $S \subseteq K$ such that $\sum_{i \in S} v_i$ is maximized subject to the constraint $\sum_{i \in S} w_i \leq W$.

The *offline* version of the knapsack problem is known to be NP-Hard. If we consider the *offline* version of our ECRTS problem for non-uniform density settings, we can see some parallelism. Consider the special case of the offline ECRTS problem where all jobs have the same release time, deadline, but possibly different values and execution times (non-uniform value density settings). In this special case, the dimension of job scheduling is trivial (as the system is underloaded and there is a common deadline, *any* scheduling order is valid). Thus, the only dimension that needs to be addressed is that of selecting a subset of jobs to maximize the system value subject to the energy budget constraint. As such, this special case maps directly to the offline KNP problem.

On the other hand, in the more general offline ECRTS problem settings where jobs may have arbitrary release times and deadlines, *scheduling* also becomes an issue in addition to the *selection of the jobs*. Now, since we assume an underloaded system, the general offline ECRTS problem can be mapped to the offline knapsack in two steps: first, selecting the set of jobs to be executed using the offline knapsack solution, and second, using preemptive EDF to schedule the selected set of jobs in feasible manner. After this transformation, standard heuristics and approximation algorithms developed for the offline KNP problem (Martello and Toth 1990) could be used to address the offline ECRTS problem.

However, the focus of this paper is *online* settings where parameters of jobs are not available in advance. As a result, one needs to compare the online KNP and online ECRTS problems. The online version of the knapsack problem occurs when the items are presented one after the other, and the algorithm has to make decisions about accepting or rejecting an item without seeing future items. Research efforts have studied the online knapsack problem using the competitive analysis framework and there are two variants of the online problem (Iwama and Taketomi 2002).

- *Irrevokable Knapsack (IR-KNP)*: In this version once the algorithm accepts an item, it cannot revoke or undo its decision at a later point of time. Thus, the online algorithm cannot replace an item it accepted in the past with a currently available and more valuable item. For the IR-KNP problem, there does not exist a competitive algorithm (i.e. an algorithm with a non-zero competitive factor) (Iwama and Taketomi 2002).
- *Revokable Knapsack (R-KNP)*: In this version, the algorithm is allowed to drop items it accepted in the past, in favor of more valuable items. However, the items that are rejected or dropped in favor of more valuable items cannot be later accepted. In other words *only* the decision of accepting an item can be revoked (notice that the problem becomes offline if both accept and reject decisions can be revoked). For the IR-KNP problem there exists a constant competitive optimal algorithm with a competitive factor of $\frac{2}{\sqrt{5}+1}$ (Iwama and Taketomi 2002).

To start with, in online ECRTS, the concept of revoking a decision is not applicable. This is because with the knapsack problem, when an accepted item is later removed (as in R-KNP), the space (or resource) it held becomes available. However, in ECRTS, the resource (or energy) spent in executing a job cannot be reclaimed at a later time. Thus, the results of R-KNP do not apply to our settings.

For IR-KNP, only a limited mapping is possible. Specifically, consider a special instance of the online ECRTS where jobs are released back-to-back and one at a time (i.e. the release time of the ith job coincides with the deadline of the $(i-1)$th job). Assume non-uniform value densities. Since, at any given time the scheduler has only one job to execute and further a new job is released only when an existing job expires, this specific instance of ECRTS maps to IR-KNP. On the other hand, for the general online ECRTS instance, where job release times and deadlines can be arbitrary, execution intervals of jobs will overlap. As a result, scheduling will become a major dimension of the problem, while it is non-existent in KNP problem. Moreover, the scheduler can be idling or non-idling, preemptive or non-preemptive; jobs can be executed partially and later abandoned, and a proper *admission test* must be designed. Obviously, in such settings, there does not exist a clear one-to-one mapping between online ECRTS and IR-KNP problems, as the latter is subsumed by the former.

The main result available in the competitive analysis of the IR-KNP problem is the non-existence of a constant-competitive algorithm (which does not imply much for the ECRTS settings). In this paper, we formally derived and quantified the upper bound on best achievable competitive factor in terms of maximum job size ($e_{\max}$) and energy budget (E). Also, we provide an algorithm *EC-EDF* that uses a non-trivial admission test rule and scheduling policy in order to achieve this upper bound. Finally, the considerations of DVS and resource augmentation dimensions do not have any immediate mapping in the KNP problems.

9 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-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-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. In particular, obtaining competitive factors for more general power models (that explicitly consider, for example, off-chip and on-chip workload components as done in Aydin et al. 2006) is an interesting research avenue.

Acknowledgements This work has been supported, in part, by US National Science Foundation grants CCF-0915681, CNS-072047 and CNS-546244 (CAREER Award). We would like to also extend our sincere thanks to the anonymous reviewers who helped us to improve our paper through their suggestions.

Appendix 1: 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 $E = k_1 \cdot \delta$ and $e_{\max} = k_2 \cdot \delta$. Where, k_1 and k_2 are integers such that $k_1 \geq k_2$, $k_1 \geq 1$ and $k_2 \geq 1$. 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.

We start with time $t = 0$. The adversary introduces $(k_1 - k_2 + 1)$ *sequential jobs* with size δ (recall from Sect. 2, that sequential jobs have zero laxity and are released back to back one at a time). At time $t_1 = (k_1 - k_2 + 1) \cdot \delta$, let $m \geq 0$ denote the number of jobs that are *not executed* by $\mathcal{A}$. We have the following three cases.

- Case 1: $m = 0$. In this case $\mathcal{A}$ has executed to completion a total workload of $W = (k_1 - k_2 + 1) \cdot \delta$, accruing a value of W units while also depleting W units of the initial energy budget E. At time t_1, the adversary introduces a single job with size $e_{\max} = k_2 \cdot \delta$. With only $(k_2 - 1) \cdot \delta$ units of energy left, $\mathcal{A}$ cannot execute this job. On the other hand, the adversary by executing $(k_1 - k_2)$ jobs of size δ and the job with size $e_{\max}$ accrues a value of $(k_1 - k_2) \cdot \delta + e_{\max}$. Thus, the competitive factor is given by:

$$\frac{(k_1 - k_2 + 1) \cdot \delta}{(k_1 - k_2) \cdot \delta + e_{\max}} = \frac{E - e_{\max} + \delta}{E}$$

- Case 2: $m \geq k_2$. In this case $\mathcal{A}$ has skipped execution of at least k_2 jobs. Thus, at time t_1, the value accrued by $\mathcal{A}$ is no more than $(k_1 - 2k_2 + 1) \cdot \delta$ and its remaining energy budget is no more than $(2k_2 - 1) \cdot \delta$. At t_1, the adversary introduces a single job of size $e_{\max}$. By executing this job, $\mathcal{A}$ can increase its value to at most $(k_1 - 2k_2 + 1) \cdot \delta + e_{\max}$. Similar to Case 1 above, the adversary makes a value of E by executing $(k_1 - k_2)$ jobs of size δ along with the job of size $e_{\max}$. The competitive factor is given by:

$$\frac{(k_1 - 2k_2 + 1) \cdot \delta + e_{\max}}{E} = \frac{E - e_{\max} + \delta}{E}$$

- Case 3: $0 < m < k_2$. In this case, first observe that in the time interval $[0, t_1]$, $\mathcal{A}$ accrues a value of $W_1 = (k_1 - k_2 + 1 - m) \cdot \delta$. Starting from time t_1, the adversary continues to incrementally release *sequential jobs* of size δ until one of the following two conditions holds: (1) k_2 jobs have been released since time t_1 *and* $\mathcal{A}$ has executed $0 \le m' < m$ of these jobs, or, (2) $\mathcal{A}$ executes m of these jobs. We have the following two sub-cases.
  - Case 3a: k_2 jobs of size δ have been released *and* $\mathcal{A}$ has executed $0 \le m' < m$ of these jobs. Thus, in the time interval $(t_1, (k_1 + 1) \cdot \delta]$, $\mathcal{A}$ accrues a value of $W_2 = m' \cdot \delta$. The adversary can accrue a total value of E by executing $(k_1 - k_2)$ jobs released in interval $[0, t_1]$ and k_2 jobs released interval $[t_1, (k_1 + 1) \cdot \delta]$. The competitive factor is given by:

    $$\frac{W_1 + W_2}{E} = \frac{(k_1 - k_2 + 1 + (m' - m)) \cdot \delta}{E}$$

    Since $m' < m$, we have

    $$\frac{(k_1 - k_2 + 1 + (m' - m)) \cdot \delta}{E} < \frac{(k_1 - k_2 + 1) \cdot \delta}{E} = \frac{E - e_{\max} + \delta}{E}$$

  - Case 3b: $\mathcal{A}$ executes m of these jobs. Let $t_2 \le (k_1 + 1) \cdot \delta$ be the time when $\mathcal{A}$ completes executing m jobs. At time t_2, $\mathcal{A}$ has accrued a value of

    $$W = W_1 + m \cdot \delta = (k_1 - k_2 + 1) \cdot \delta = E - e_{\max} + \delta$$

    Note that at time t_2 the remaining energy budget of $\mathcal{A}$ is $(k_2 - 1) \cdot \delta$ units. At time t_2, the adversary releases a single job with size $e_{\max}$ which $\mathcal{A}$ cannot execute. The adversary, by executing $(k_1 - k_2)$ jobs released in interval $[0, t_1)$ and the job with size $e_{\max}$ released at t_2, accrues a total value of E. Thus, the competitive factor is given by $\frac{W}{E} = \frac{E - e_{\max} + \delta}{E}$.

From Cases 1, 2, and 3, we can conclude that no online algorithm can achieve a competitive factor greater than $\frac{E - e_{max+\delta}}{E}$. Since $\frac{\delta}{E}$ can be arbitrarily low, the upper bound on the achievable competitive factor is given by $\frac{E - e_{\max}}{E}$.

Appendix 2: Proof of Theorem 3

In order to prove Theorem 3, we will provide an input instance for which the ratio of the total value accrued by any online algorithm $\mathcal{A}$ to that of a clairvoyant adversary cannot be more than $\frac{1}{(k_{\max})^{\frac{e_{\max}}{E}}}$.

This instance consists of a series of *periods* $P_1, P_2, \dots, P_m$ $m \geq 1$. The exact number of periods (m), as well as the exact number of jobs released in each period, depend on the actions of the algorithm $\mathcal{A}$. Our strategy will consist in showing that, *in each period*, the ratio of the value accrued by $\mathcal{A}$ to that of the adversary cannot be greater than $\frac{1}{(k_{\max})^{\frac{e_{\max}}{E}}}$. This, in turn, will establish the upper bound over all the periods, that is, the competitive factor of the entire input instance.

Let the remaining energy of $\mathcal{A}$ at the beginning of the period P_i be E_i. Clearly, $E_1 = E \geq e_{\max}$. All the jobs released within a period are released back-to-back and with laxity zero; that is, at the deadline of a job, a new job is released. We first describe the structure of the first period P_1 and then show how it can be generalized to multiple periods.

In the first period, first a job with value density $k_{\min}$ and size $X \leq e_{\max}$ is released by the adversary. The adversary keeps releasing such jobs with value density $k_{\min}$ and size X back to back (i.e. the following job is released at the deadline of the previous one) until one of the following conditions occurs:

a. Either the online algorithm $\mathcal{A}$ does not accept *any* of these jobs until the total energy requirement (the total size) of the released jobs in period P_1 reaches E_1.

   In this case, the adversary stops releasing any new jobs. No more periods are introduced and this marks the end of the input instance as well. While $\mathcal{A}$ obtained a value of zero in this period, the adversary announces that it has accrued a non-zero value by executing all $\lfloor \frac{E_1}{X} \rfloor$ jobs that it has released; yielding the total value ratio zero in this (last) period.

b. Or, the online algorithm $\mathcal{A}$ accepts one of these jobs at some point before their total energy requirement does not exceed E_1.

   At this point, the adversary releases another job of size X but with the value density $k \cdot k_{\min}$ at the deadline of the first job accepted by $\mathcal{A}$. If $\mathcal{A}$ executes this second job as well, the adversary releases a third job of size X, and value density $k^2 \cdot k_{\max}$ at the deadline of the second job and so on. This pattern continues, i.e. the adversary keeps releasing jobs where the value density of each job is equal to k times that of the previous one, until one of the following conditions is satisfied: either the remaining energy of the player becomes strictly less than $e_{\max}$, or, $\mathcal{A}$ rejects executing a job even though its remaining energy is greater than or equal to $e_{\max}$. We examine each of these cases in detail below.

b1. After executing $s+1$ jobs with value densities $k_{\min}, k \cdot k_{\min}, \dots, k^s \cdot k_{\min}$, the remaining energy of $\mathcal{A}$ becomes strictly smaller than $e_{\max}$.

   In this case, as the last job of this period (and as the last job of the entire input instance), the adversary releases a job of size $e_{\max}$, zero laxity and value density $k^{s+1} \cdot k_{\min}$. Observe that $\mathcal{A}$ cannot execute this job due to the energy deficiency. However, the adversary announces that it has skipped all the jobs with value density $k_{\min}$, but executed the s jobs with value densities $k \cdot k_{\min}, \dots, k^s \cdot k_{\min}$ and the last one with value density $k^{s+1} \cdot k_{\min}$ in this period.[3]

[3]Observe that, in this case, $E_1 = (s+1) \cdot X + Y$ where $s \geq 0$ and $0 \leq Y < e_{\max}$. Recalling that $E_1 \geq e_{\max}$ and $X \leq e_{\max}$, we can infer that $E_1 \geq (X+Y) \geq e_{\max}$; because if this was not true, there would not be a non-negative integer s for which $E_1 = (s+1) \cdot X + Y$ holds. So, we can re-write E_1 as $s \cdot X + (X+Y)$

Thus, the ratio of the values of accrued by $\mathcal{A}$ and the adversary in this last period is given by:

$$\begin{aligned} c &= \frac{(1+k+k^2+\cdots+k^s)\cdot X\cdot k_{\min}}{(k+k^2+\cdots+k^s)\cdot X\cdot k_{\min}+(k^{s+1}k_{\min})\cdot e_{\max}} \\ &\le \frac{(1+k+k^2+\cdots+k^s)\cdot X\cdot k_{\min}}{(k+k^2+\cdots+k^s)\cdot X\cdot k_{\min}+(k^{s+1}k_{\min})\cdot X} \\ &\le \frac{1}{k}. \end{aligned}$$

b2. Or, after executing $s+1$ jobs with value densities $k_{\min}, k\cdot k_{\min}, \ldots, k^s\cdot k_{\min}$, $\mathcal{A}$ rejects executing the job with value density $k^{s+1}\cdot k_{\min}$ even though its remaining energy is greater than or equal to $e_{\max}$.

In this case, in this period $\mathcal{A}$ accrues a total value of $k_{\min}\cdot X\cdot \sum_{i=0}^{s} k^i$. Then, the adversary announces that it has skipped all the jobs it has released with value density $k_{\min}$; instead, it has executed the $(s+1)$ jobs with value densities $k\cdot k_{\min}, \ldots, k^s\cdot k_{\min}, k^{s+1}\cdot k_{\min}$, making a total value of $k_{\min}\cdot X\cdot \sum_{i=1}^{s+1} k^i$. Hence, the ratio of the total value of $\mathcal{A}$ to that of the adversary in this period is:

$$c = \frac{k_{\min}\cdot X\cdot \sum_{i=0}^{s} k^i}{k_{\min}\cdot X\cdot \sum_{i=1}^{s+1} k^i} = \frac{1}{k}. \tag{1}$$

At this point, this period ends and we start a new period with the same job release pattern as that in the previous one. Observe that, up to this point, the adversary and $\mathcal{A}$ have executed exactly the same number of jobs with the same size; hence, at the beginning of the next period, their remaining energy levels are identical. Further, by the very nature of the condition that must be satisfied at the beginning of the case (b2), this energy level is definitely greater than or equal to $e_{\max}$.

Thus, the adversary starts a new period by releasing jobs of size X and value density $k_{\min}$ back to back, until $\mathcal{A}$ picks up one of these and the whole analysis above can be repeated to establish that the value ratio at the end of the second, third, and all subsequent periods cannot exceed $\frac{1}{k}$. Obviously, this sequence of periods will end after a finite number of steps, either when $\mathcal{A}$ does not pick up any job in the sequence of back-to-back released jobs with value density $k_{\min}$ (in case (a)), or by the triggering condition of the case (b1) above (as the remaining energy monotonically decreases whenever $\mathcal{A}$ executes a job). By considering the fact that the value ratio cannot exceed $\frac{1}{k}$ in any of the periods, we show that the competitive factor is indeed bounded by $\frac{1}{k}$.

We now examine how large k can be. Let P_j be the period during which the algorithm $\mathcal{A}$ executes the maximum number of jobs. The last job released by the adversary in this period P_j has value density k^{i+1}. We will fix the value density

where $(X+Y)\ge e_{\max}$; that is, the adversary has sufficient energy to execute the s jobs with size X, along with the last one of size $e_{\max}$.

of this job to $k_{\max}$. Thus, $k^{i+1} = k_{\max}$. Observe that $\mathcal{A}$ executed exactly $i + 1$ jobs in this period P_j. Note that the number of jobs executed by $\mathcal{A}$ in the period P_j puts a constraint on $i + 1$; that is, $(i + 1) \cdot X \leq E$. That is, $(i + 1)$ can be, at most, equal to $\frac{E}{X}$. As a result we obtain, $k_{\max} = k^{i+1} = k^{\frac{E}{X}}$; or equivalently, $k = (k_{\max})^{\frac{X}{E}}$. To maximize k, we need to choose the maximum possible value for X, which is equal to $e_{\max}$. In that case, the competitive factor c is bounded by at most $\frac{1}{k} = \frac{1}{(k_{\max})^{\frac{e_{\max}}{E}}}$, completing the proof.

References

AlEnawy TA, Aydin H (2005) Energy-constrained scheduling for weakly-hard real-time systems. In: Proceedings of the real-time systems symposium (RTSS'05)

AlEnawy TA, Aydin H (2004) On energy-constrained real-time scheduling. In: Proceedings of the European conference on real-time systems (ECRTS'04)

Aydin H, Devadas V, Zhu D (2006) System-level energy management for periodic real-time tasks. In: Proceedings of real-time systems symposium (RTSS'06)

Aydin H, Melhem R, Mosse D, Mejia-Alvarez P (2004) Power-aware scheduling for periodic real-time tasks. IEEE Trans Comput 53(10)

Bansal N, Kimbrel T, Pruhs K (2004) Dynamic speed scaling to manage energy and temperature. In: Symposium on foundations of computer science (FOCS'04)

Baruah S, Rosier L, Howell R (1990) Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. In: Real time systems(2)

Baruah S, Koren G, Mishra B, Raghunathan A, Rosier L, Shasha D (1991a) Online scheduling in the presence of overload. In: Proceedings of the symposium on foundations of computer science (FOCS'91)

Baruah S, Koren G, Mao D, Mishra B, Raghunathan A, Rosier L, Shasha D, Wang F (1991b) On the competitiveness of online real-time task scheduling. In: Proceedings of the real-time systems symposium (RTSS'91)

Baruah S (1998) Overload tolerance for single-processor workloads. In: Proceedings of the real-time technology and application symposium (RTAS'98)

Baruah S, Haritsa J (1997) Scheduling for overload in real-time systems. IEEE Trans Comput 46(9)

Baruah S, Hickey ME (1998) Competitive online scheduling of imprecise computations. IEEE Trans Comput 47(9)

Borodin A, El-Yavin R (1998) Online computation and competitive analysis. Cambridge University Press, Cambridge

Buttazzo G (2005) Hard real-time computing systems: predictable scheduling algorithms and applications, 2nd edn. Springer, Berlin

Chan HL, Chan WT, Lam TW, Lee LK, Mak KS, Wong P (2007) Energy efficient online deadline scheduling. In: Proceedings of the symposium on discrete algorithms (SODA'07)

Chen JJ, Kuo TW (2005) Voltage scaling scheduling for periodic real-time tasks in reward maximization. In: Proceedings of the real-time system symposium (RTSS'05)

Dertouzos M, Mok AK (1989) Multiprocessor online scheduling for hard real-time tasks. IEEE Trans Softw Eng 15(12)

Dertouzos M (1974) Control robotics: the procedural control of physical processes. In: Proceedings of IFIP congress

Ebenlendr T, Sgall J (2009) Semi online preemptive scheduling: one algorithm for all variants. In: Proceedings of international symposium on theoretical aspects of computer science (STACS'09)

Garey MR, Johnson DS (1990) Computers and intractability. A guide to the theory of Np-completeness. Freeman, New York

Iwama K, Taketomi S (2002) Removable online knapsack problems. In: Proceedings of the international colloquium on automata, languages and programming (ICALP'02)

Jeffay K, Stone DL (1993) Accounting for interrupt handling costs in dynamic priority task systems. In: Proceedings of the real-time systems symposium (RTSS'93)

Jeffay K, Stanat DF, Martel CU (1991) On non-preemptive scheduling of periodic and sporadic tasks. In: Proceedings of the real-time systems symposium (RTSS'91)

Kalyanasundaram B, Pruhs K (1995) Speed is as powerful as clairvoyance. In: Proceedings of the symposium on foundations of computer science (FOCS'95)

Koren G, Shasha D (1992) D-over: an optimal online scheduling algorithm for overloaded real-time systems. In: Proceedings of the real-time systems symposium (RTSS'92)

Koren G, Shasha D, Huang SC (1993) MOCA: a multiprocessor online competitive algorithm for real-time scheduling. In: Proceedings of the real-time systems symposium (RTSS'93)

Lee CH, Shin KG (2004) Online dynamic voltage scaling for hard real-time systems using the EDF algorithm. In: Proceedings of the real-time systems symposium (RTSS'04)

Liu J (2000) Real time systems. Prentice Hall, New York

Martello S, Toth P (1990) Knapsack problems: algorithms and computer implementation. Wiley, New York

Palis MA (2004) Competitive algorithms for fine-grain real-time scheduling. In: Proceedings of the real-time systems symposium (RTSS'04)

Phillips C, Stein C, Torng E, Wein J (2002) Optimal time-critical scheduling via resource augmentation. Algorithmica 163–200

Pillai P, Shin KG (2001) Real-time dynamic voltage scaling for low-power embedded operating systems. In: Proceedings of the symposium on operating systems principles (SOSP'01)

Pruhs K, Sgall J, Torng E (2004) Maximizing rewards for real-time applications with energy constraints. In: Leung JYT (ed) The handbook of scheduling, algorithms, models and performance analysis. CRC press, Boca Raton

Rusu C, Melhem R, Mosse D (2003) Maximizing rewards for real-time applications with energy constraints. ACM Trans Embed Comput Syst 2(4)

Rusu C, Melhem R, Mosse D (2002) Maximizing the system value while satisfying time and energy constraints. In: Proceedings of the real-time system symposium (RTSS'02)

Wu H, Ravindran B, Jensen ED (2007) Utility accrual real-time scheduling under the unimodal arbitrary arrival model with energy bounds. IEEE Trans Comput 56(10)

Yao F, Demers A, Shenker S (1995) A scheduling model for reduced CPU energy. In: Proceedings of the symposium on foundations of computer science (FOCS'95)

Vinay Devadas received his B.S. degree in Computer Science and Engineering from Visvesvaraya Technological University, Bangalore, India in 2005 and his M.S. degree in Computer Science from George Mason University, Fairfax, VA in 2007. He is currently a Ph.D. candidate at the Department of Computer Science, George Mason University. His research interests include low-power computing, real-time embedded systems and operating systems.

Fei Li received the B.S. degree in Computer Science from Jilin University, Changchun, China, in 1997, the M.S., M.Phil., and Ph.D. degrees in Computer Science from Columbia University, New York, NY, in 2002, 2007, and 2008, respectively. He joined the Department of Computer Science at George Mason University as an Assistant Professor in 2007. His research interests include online and approximation algorithm design and analysis, combinatorial optimization, and scheduling algorithms.

Hakan Aydin received the B.S. and M.S. degrees in Control and Computer Engineering from Istanbul Technical University in 1991 and 1994, respectively, and the Ph.D. degree in computer science from the University of Pittsburgh in 2001. He is currently an Associate Professor in the Computer Science Department at George Mason University, Fairfax, Virginia. He has served on the program committees of several conferences and workshops, including the IEEE Real-Time Systems Symposium and IEEE Real-time Technology and Applications Symposium. He is currently serving as the Technical Program Committee Chair of IEEE Real-time Technology and Applications Symposium (RTAS'11). He was a recipient of the US National Science Foundation (NSF) Faculty Early Career Development (CAREER) Award in 2006. His research interests include real-time systems, low-power computing, and fault tolerance.