International Journal of Automation and Computing

DOI: 10.1007/s11633-016-0993-3

Energy Efficient Scheduler of Aperiodic Jobs for Real-time Embedded Systems

Hussein El Ghor[1] E. M. Aggoune[2]

[1]University Institute of Technology, Lebanese University, Lebanon Sensor Networks and Cellular Systems (SNCS) Research Center UT, Saudi Arabia

[2]Electrical Engineering Department, University of Tabuk Sensor Networks and Cellular Systems (SNCS) Research Center 71491 Tabuk, Saudi Arabia

Abstract: Energy consumption has become a key metric for evaluating how good an embedded system is, alongside more performance metrics like respecting operation deadlines and speed of execution. Schedulability improvement is no longer the only metric by which optimality is judged. In fact, energy efficiency is becoming a preferred choice with a fundamental objective to optimize the system′s lifetime. In this work, we propose an optimal energy efficient scheduling algorithm for aperiodic real-time jobs to reduce CPU energy consumption. Specifically, we apply the concept of real-time process scheduling to a dynamic voltage and frequency scaling (DVFS) technique. We address a variant of earliest deadline first (EDF) scheduling algorithm called energy saving-dynamic voltage and frequency scaling (ES-DVFS) algorithm that is suited to unpredictable future energy production and irregular job arrivals. We prove that ES-DVFS cannot attain a total value greater than $C/\widehat{S}^{\alpha}$, where $\widehat{S}$ is the minimum speed of any job and C is the available energy capacity. We also investigate the implications of having in advance, information about the largest job size and the minimum speed used for the competitive factor of ES-DVFS. We show that such advance knowledge makes possible the design of semi-on-line algorithm, ES-DVFS**, that achieved a constant competitive factor of 0.5 which is proved as an optimal competitive factor. The experimental study demonstrates that substantial energy savings and highest percentage of feasible job sets can be obtained through our solution that combines EDF and DVFS optimally under the given aperiodic jobs and energy models.

Keywords: Real-time systems, energy efficiency, aperiodic jobs, scheduling, dynamic voltage scaling, low-power systems, embedded systems.

1 Introduction

The number of embedded systems operated by batteries is constantly increasing with applications in autonomous robots, battlefields, industrial process monitoring, mobile communication systems, and environmental monitoring, to name but a few. In these systems, time is no longer the only metric by which performance is judged. In fact, reducing the energy consumption is of primary importance to prolong the battery life. Hence, the stringent timing constraints as well as reducing energy consumption are highly desirable and sometimes critical features of any embedded computing system.

Due to increased computational demands, the obvious target for energy reduction became the processor. Recent reports show that the processor consumes more than 50% of the total energy[1].

Dynamic voltage and frequency scaling (DVFS) is a popular and widely used technique for power management in real-time embedded systems[2]. With DVFS, one can dynamically adjust the voltage and frequency of the CPU to reduce energy consumption. As we throttle the processor, we inevitably sacrifice the execution speed, and consequently an utmost care must be exercised to find the convenient slowdown factor that saves the most energy while avoiding deadline misses.

This paper investigates the on-line job scheduling problem for real-time embedded systems implemented on a variable voltage processor. The real-time system consists of aperiodic jobs where we have no prior knowledge about the characteristics of the future jobs until their release. The idea behind an on-line scheduling problem is that the proposed algorithm must decide at each time t the job to be executed but it is not aware of the existence of a job until its release. As soon as a job is released, the scheduler must be able to learn all its characteristics including the processing time, deadline and energy consumption.

To measure the overall system performance, we add a value for each job if and only if it meets its deadline without depleting the energy reservoir. It is important to note that no value is accrued for partial executions of any job. Hence, if a job set is 100% feasible, then the total system value is equal to the sum of the worst case execution times (WCET).

The worst-case performance guarantees of on-line algorithms are often derived by comparing it to that of an optimal algorithm that knows the entire input in advance. This framework, known as competitive analysis, is considered as

Research Article

Manuscript received December 11, 2014; accepted June 3, 2015

This work was fully supported by the SNCS Research Center, the University of Tabuk, and the Ministry of Higher Education in Saudi Arabia.

Recommended by Associate Editor Jinhua She

© Institute of Automation, Chinese Academy of Sciences and Springer-Verlag Berlin Heidelberg 2016

a standard analysis and well-known technique in theoretical computer science[3]. Over the past decade, the research community has made significant progress in analyzing the competitive analysis framework[4, 5]. However, the competitive analysis of energy-constrained on-line real-time scheduling has remained an open problem.

The introduction of dynamic voltage and frequency scaling (DVFS) capabilities into embedded systems posits a lot of questions. Firstly, how to dynamically adapt the concept of real-time process scheduling to a DVFS technique? Secondly, how to dynamically adjust the voltage and frequency of the processor to reduce energy consumption? Thirdly, how to guarantee a high performance under unpredictable future energy production and irregular job arrivals?

The remaining of this paper is organized as follows: In the next section, we summarize the related work. The system and energy model are introduced in Section 3. Section 4 formulates the problem and gives some definitions. In Section 5, an optimal scheduling algorithm is presented with illustrative example and some notations. A semi-on-line algorithm is introduced in Section 6. Experimental results are discussed in Sections 7 and 8 concludes the paper.

2 Prior work

Aperiodic jobs are typically used to serve random processing requirements, such as operator requests or displaying activities[6]. An aperiodic job (or non-periodic job) demands running once. The activation of such a job takes place during the occurrence of an event that can be either external when issued by an environment or internal when issued from another job. The problem of scheduling aperiodic jobs has been widely considered in literature. The most well-known strategies are dynamic priority policies and fixed priority policies. With fixed priority policies, all jobs of a particular job set have the same priority level. Rate monotonic scheduling (RM)[7] and deadline monotonic scheduling (DM)[8] are the most well-known examples of such policy. On the other hand, with dynamic priority policies, different jobs of the same job set may have different priority levels at runtime. The most known algorithm among such scheduling approaches is the earliest deadline first (EDF) algorithm[9]. Scheduling aperiodic jobs under the EDF algorithm was first investigated in [10, 11]. Our work is based on dynamic priority scheduling, preemptive and without resource and precedence constraints.

Recently, the research community posited that power management is both crucial and necessary for real-time embedded systems. Among the earliest works, Yao et al.[12] addressed the real-time dynamic voltage scaling (RT-DVS) problem. That means the problem of minimizing the dynamic energy consumption of the processor without violating timing constraints. For aperiodic jobs, a polynomial optimal static off-line solution based on speed assignments that minimizes the energy consumption for the worst-case workload is evaluated. However, RT-DVS is limited by the static solution. This drawback was later solved in [13]. Authors proposed heuristics for on-line scheduling of aperiodic jobs that takes into account the feasibility of periodic requests. Another choice was suggested in [13]. Based on the absence of aperiodic jobs, the CPU speed must be set to the processor utilization of the periodic jobs. Still, the optimality of this choice is not demonstrated. Same authors investigated the non-preemptive power-aware scheduling problem in [14].

An early technique based on slowing down the processor whenever there is only one job in the ready queue eligible for execution was proposed in [15]. A static solution for periodic jobs with different power characteristics was given in [16]. The problem of computing job slowdown factors for periodic jobs where job deadlines are different from the job period was addressed in [17]. Research efforts in [18, 19] aimed to benefit from the unused CPU time due to early completions of some jobs so as to further reduce the CPU energy by applying the DVS policy.

In [18], authors suggested a cycle-conserving algorithm that relies on the concept of dynamic utilization. That means that the processor utilization is updated dynamically relative to the actual execution times of the ready jobs. Authors also presented a look-ahead algorithm that uses low frequencies in presence of slack time and postpone the use of high frequencies as much as possible. Based on the same concept, Aydin et al.[19] proposed the generic dynamic reclaiming Algorithm (GDRA). The GDRA algorithm reduces the CPU frequency by transferring the slack of high priority jobs to low-priority ones.

Another solution to the power management problem in real-time systems is based on prediction mechanisms. In [19], authors suggested the aggressive speed adjustment (AGR) policies. AGR relies on the execution history of jobs to set the CPU frequency. Later, an on-line scheduling algorithm that makes scheduling decisions based on the history of already arrived events and the predicted future arrivals was proposed in [20]. However, using predictions may lead to potential deadline misses and inefficient energy management.

With processors bounded between a minimum and maximum speeds, authors in [21] presented an on-line DVS scheduling algorithm that aims to maximize the throughput and optimize the trade-off between the total flow time incurred and the energy consumed by jobs. Unfortunately, this algorithm is dedicated to soft real-time systems and cannot be adopted for systems with hard real-time constraints.

Aiming to reduce the energy consumption of cloud computing platforms while respecting the QoS requirements of tasks, authors in [22] presented a novel scheduling algorithm, named proactive and reactive scheduling (PRS), that dynamically exploits proactive and reactive scheduling methods, for scheduling real-time, aperiodic, independent tasks. Unlike other approaches that consider deterministic cloud computing environments, authors addressed a scheduling architecture to mitigate the impact of uncer-

tainty on the task scheduling quality for a cloud data center. Three strategies were then proposed to scale up and down the system′s computing resources according to workload to reduce energy consumption.

Very recently[23], we presented an energy saving EDF (ES-EDF) algorithm that is dedicated to independent and preemptive periodic tasks. ES-EDF is capable of stretching the worst case execution time of tasks as much as possible without violating deadlines. We demonstrated that ES-EDF is optimal in minimizing the energy consumption and the maximum lateness of the processor for which an upper bound on the processor energy saving is derived.

3 Model and terminology

3.1 System model

Hereafter, we describe our model that comprises three components: a processing element with N discrete frequencies, a job model and an energy storage unit.

3.1.1 Job model

We consider a uniprocessor system that executes aperiodic jobs. Each one is known by the system at the time of its arrival. Preemptive scheduling is assumed. We use Ψ to denote the finite input sequence of jobs that arrive to the system during its operation. Ψ is denoted as follows: $\Psi = \{J_i \mid 1 \leq i \leq n\}$. Every job J_i is characterized by (r_i, d_i, e_i) where r_i and d_i are the arrival time and the deadlines of job J_i, respectively. We consider in this work that the execution per unit time requires unit energy. For this sake, we use e_i to denote the size of the job J_i. This means e_i indicates both the worst case execution time (WCET) and energy requirement of J_i. We denote the laxity of the job J_i by $d_i - (r_i + e_i)$.

We assume that Ψ is feasible in the real-time sense. That means that when the energy constraints are not taken into consideration, there exists a feasible schedule where all deadlines in Ψ are respected.

3.1.2 DVFS system configuration

The system is equipped with a DVFS-enabled CPU where the processing frequency is assumed to be working with N discrete frequencies f ranging from $f_{\min} = f_1 \leq f_2 \leq \cdots \leq f_n = f_{\max}$[23]. The power consumption of the jobs running on the processor and frequency levels are in a way coupled together. When we change the speed of a processor, its operating frequency is changed and hence the power consumption of jobs is proportionately changed to a value which is supported at that operating frequency.

We use the term slowdown factor S_n as the ratio of the scheduled speed to the maximum processor speed. S_n ranges from $S_{\min}$ to 1:

$$S_n = \frac{f_n}{f_{\max}}. \tag{1}$$

We consider in our work that each job has different power consumption that varies according to its frequencies. Consequently, a job will have maximum power consumption at its maximum frequency and this power consumption decreases as the frequency decreases. Consequently, the power consumption of a job must be defined as function of the job index and its corresponding slowdown factor $P_i(J_i, S_i)$.

Moreover, the power consumption of the processor at a slowdown factor S is modeled as a convex function $P(S)$ where $P(S) = aS^{\alpha}$. a is considered as a constant characterized by the processor parameters and $2 \leq \alpha \leq 3$. We assume in this work that $a = 1$.

By considering DVFS policy, the energy required to execute a job depends on the slowdown factor at which the job is executed. As such, the assumption that the execution per unit time requires unit energy is no longer valid. For this sake, a job J_i is now represented as $J_i(r_i, C_i, d_i, e_i)$ where C_i is the worst case execution time at maximum slowdown factor ($S_{\max} = 1$). e_i is the minimum energy requirement relative to the minimum slowdown factor that would allow the job J_i to timely finish its execution without violating its deadline d_i.

On DVS-enabled systems, it is assumed that job execution times scale linearly with CPU frequency. Thus, when a job J_i is stretched by a slowdown factor S_i, its actual execution time $C_i(a)$ and its energy required will be C_i/S_i and $e_i(J_i, S_i) = P_i(J_i, S_i).\frac{C_i}{S_i} = S_i^{\alpha-1}C_i$, respectively.

We assume that preemption overheads are negligible. Otherwise, they can be incorporated into the jobs′ worst-case execution times[24].

3.1.3 Energy storage

Our system relies on an ideal energy storage unit, battery for example, that has a nominal capacity, namely C, corresponding to a maximum energy (expressed in Joules or Watts-hour). The energy level has to remain between two boundaries $C_{\min}$ and $C_{\max}$ with $C = C_{\max} - C_{\min}$. We denote by $C(t)$, the energy stored in the battery at time t. At any time, the stored energy is no more than the storage capacity, that is

$$C(t) \leq C \ \ \forall\, t. \tag{2}$$

3.2 Terminology

We now recall some definitions for real-time scheduling concepts that we need throughout the remainder of the paper.

Definition 1. A schedule Γ for Ψ is said to be valid if the deadlines of all jobs of Ψ are met in Γ, starting with a storage fully charged.

Definition 2. A system is feasible if there exists at least one valid schedule for Ψ with the given energy storage unit. Otherwise, it is infeasible.

Definition 3. A scheduling algorithm A is optimal if it finds a valid schedule whenever one exists.

Definition 4. A scheduling algorithm A is on-line if it makes its decisions at run-time.

Definition 5. A scheduling algorithm A is semi on-line if it is on-line where partial information about the input is given to the scheduler in advance.

We now introduce a novel terminology which is exclu-

sively related to energy constrained real-time computing systems.

Definition 6. A schedule Γ for Ψ is said to be time-valid if the deadlines of all jobs of Ψ are met in Γ, considering that $\forall\, j \in \{1, \cdots, n\}$, $E_j = 0$.

Definition 7. A system is said to be time-feasible if there exists at least one time-valid schedule for Ψ.

Definition 8. A schedule Γ for Ψ is said to be energy-valid if the deadlines of all jobs of Ψ are met in Γ, considering that $\forall\, j \in \{1, \cdots, n\}$, $C_j = 0$.

Definition 9. A system is said to be energy-feasible if there exists at least one energy-valid schedule for Ψ.

4 Problem formulation

Given an aperiodic real-time application of n independent aperiodic jobs $\Psi = \{J_1, J_2, \cdots, J_n\}$ with a release time, worst-case execution time and deadline, running on a DVS-enabled platform, and using a single processor during execution, the problem of determining the CPU slowdown decisions so as to minimize the overall system energy is considered. we refer to a set of speed values during the whole time interval where Ψ is executed as a speed schedule.

Before proceeding, let us state some basic definitions and existing results that will be instrumental in our analysis.

Definition 10. Let $\Psi'(t_1, t_2)$ denote the set of jobs ready to be processed at time t_1 and with deadlines at or earlier than t_2 and let $W(t_1, t_2)$ denote the total amount of workload of jobs in $\Psi'(t_1, t_2)$. The effective loading factor $h(t_1, t_2)$ over an interval $[t_1, t_2]$ is defined as $h(t_1, t_2) = \frac{W(t_1, t_2)}{t_2 - t_1}$

Definition 11. Let $\Psi'(t_1, t_2)$ denote the set of jobs ready to be processed at time t_1 and with deadlines at or earlier than t_2 and let $I(t_1, t_2)$ denote the intensity of jobs in $\Psi'(t_1, t_2)$. $I(t_1, t_2)$ is computed as

$$I(t_1, t_2) = \max_{j \in \Psi'} \left(\frac{\sum\limits_{d_i \le d_j} C_i}{d_j - (t_2 - t_1)} \right). \tag{3}$$

The most helpful observation in formulating our problem was presented in [12]. Authors stated the best speed for a given set of jobs within a defined interval of time.

Theorem 1. Let $\Psi'(t_1, t_2)$ denote the set of jobs ready to be processed at time t_1 and with deadlines at or earlier than t_2. The speed schedule that employs a constant speed in $[t_1, t_2]$ is necessarily an optimal schedule in the sense that no other schedule consumes less energy to complete the jobs in time.

Based on the above theorem, we can now prove the following lemma that describes the optimal speed schedule for a job set Ψ.

Lemma 1. An optimal speed schedule for a job set Ψ is defined on a set of time intervals in which the processor maintains a constant speed $S_i = \max(I_j, h_k)$ where h_k and I_j are respectively the workload and intensity of jobs in $[t_i, t_j]$ and each of these intervals $[t_i, t_j]$ must start at t_i and with deadlines at or earlier than t_j.

Proof. Let $\Psi'(t_i, t_j)$ denote the set of ready jobs in the interval of time $[t_i, t_j]$ and $t_0, t_1, \cdots, t_n$ as the release times or deadlines of the jobs. Let $S_1, S_2, \cdots, S_N$ be the constant speeds such that

$$S_i = \max(I_j, h_k) \tag{4}$$

where

$$h_k = \frac{\sum\limits_{\Psi'(t_i, t_j)} C_i}{d^{\max}} \tag{5}$$

and $d^{\max}$ is the maximum deadline in the job set $\Psi'(t_i, t_j)$.

If the processor executes all the jobs in $\Psi'(t_i, t_j)$ according to S_i, then no job will violate its deadline. According to Theorem 1, the processor will consume less energy to complete the jobs at speeds designated by $S_1, S_2, \cdots, S_N$ where the optimal speed schedule will only change when passing to other interval. If the processor operates accordingly, all the jobs in Ψ must be completed by their deadlines and no other voltage schedules can consume lesser energy.

5 Optimal scheduling algorithm

5.1 Presentation of the scheduler

The intuition behind energy saving-dynamic voltage and frequency (ES-DVFS) algorithm is to schedule aperiodic jobs as soon as possible according to earliest deadline first (EDF) in the presence of variability in dynamic execution behavior. We use a modified EDF strategy to reduce the CPU energy consumption by using the dynamic voltage and frequency selection. ES-DVFS uses an on-line speed reduction mechanism to minimize the system-wide energy consumption by adapting to the actual workload. ES-DVFS still guarantees that all deadlines are met.

5.2 Computing the minimum constant speed for each job

The ES-DVFS scheduler maintains a priority job queue in which jobs are ordered by the EDF basis. In the beginning, the job queue is empty. Scheduling decisions are only applied when any of the following events really arrive: 1) Event 1: a new aperiodic job is ready and is added to the job queue. 2) Event 2: the current job completes its execution.

We use a dynamic-priority assignment approach where jobs are executed by a variable speed processor. Hence, the worst case execution time (WCET) of each job varies depending on the processor slowdown factor under different speed levels. The challenge is how to essentially build an optimal speed schedule which leads to the maximum energy saving. Our technique is based on the assumption that the parameters of each job are only known when it is released.

ES-DVFS attempts to allocate the maximum possible amount of slack time based on jobs presented in the ready job list. When Event 1 occurs, the ES-DVFS updates the optimal speed schedule of the processor for all the jobs including the new one in the job queue and consequently the

slack time is updated. Further, when an Event 2 occurs, the completed job is removed from the job queue and we execute the job with the highest priority following a new calculated slowdown factor.

When preemption occurs, we implicitly calculate the new processor speed to the favor of the newly dispatched job. It is formally proved that the jobs will still meet their deadlines if the speed is changed according to the jobs found in the job queue. This is due to the fact that we keep track of the remaining execution times of the preempted job in the queue, and consequently the worst-case workload is still the same.

In aperiodic real-time system, we are not able to reveal how jobs actually arrive. Thus, we cannot know the maximum deadline or the worst case execution time of the jobs before beginning the schedule. Instead, we would like to apply on-line DVS scheduling algorithm to make scheduling decisions only when jobs really arrive, i.e., only for jobs in the job queue.

Based on ES-DVFS, scheduling decisions at time t are as follows: ES-DVFS selects the job with the earliest deadline J_i and then adopt the speed such that the job has to be finished exactly at the release time of the next job. This means that J_i is executed at speed $S_i = \frac{C_i}{d_i - t}$ where a job J_i is associated with worst-case execution time (at $S_{\max}$) C_i and absolute deadline d_i.

However, if we do not consider the maximum intensity of all the ready jobs, it is possible that the required execution speed, at some moment t, of the resulting schedule might miss future deadlines. Therefore, in order to guarantee that we still meet all the jobs$'$ deadlines in the ready queue $\wp$, the workload h_k and the intensity I_j must be verified in advance by considering the highest speed between h_k and I_j.

In other words, J_i is executed at speed $S = \max(I_j, h_k)$. h_k is the workload of jobs in the ready queue p. This means $h_k = \sum_{J_i \in p} \frac{C_i}{d_{\max}}$ where $d_{\max}$ is the maximum deadline in p.

5.3 ES-DVFS algorithm

The ES-DVFS algorithm provides sound dynamic speed reduction mechanisms. We integrate DVFS techniques with EDF scheduler for aperiodic real-time applications that potentially use uniprocessor devices during execution. ES-DVFS provides an exact energy management technique as function of the CPU frequency in such a way that time constraints are still met. Using this framework, the speed of the jobs ready to be executed is dynamically adjusted on the fly.

As mentioned above, when a job arrives, it is added to so called job ready queue p. At any time t, there is a single job J_i eligible for execution. Thus, before executing this job, we should use the minimum CPU speed available to stretch out the WCET as much as possible without violating deadlines. Therefore, job J_i must be executed with a speed S equal to the total workload h_k in p.

Note that stretching out a job J_i at a time t with speed h_k may lead to deadline violations. In this case, using the speed $S = \max(I_j, h_k)$ will result in a total effective workload which is equal to 1. Hence, ES-DVFS can achieve up to 100 percent CPU utilization where all the jobs are completed before their deadlines.

The major components of ES-DVFS are: $C(t)$, $h(t)$ and $I(t)$ where t is the current time, $C(t)$ is the amount of energy that is currently stored at time t that means the remaining amount of energy in the energy storage at time t. $h(t)$ and $I(t)$ are respectively the workload and the intensity of a job at current time t. Moreover, we use the function $execute()$ to put the processor to run the ready job with the earliest deadline.

We describe in Algorithm 1 the pseudo code of the ES-DVFS scheduler:

Algorithm 1. Energy saving-earliest deadline first (ES-EDF) algorithm

Require: A Set of N aperiodic jobs $\Psi = \{J_i | J_i = (r_i, C_i, d_i, e_i), \quad i = 1, \cdots, N\}$ according to EDF, current time t, battery with capacity ranging from $C_{\max}$ to $C_{\min}$, energy level of the battery $C(t)$.

Require: A processor working with DVFS policy. Power consumption of the processor at a speed S is $P(S) = S^{\alpha}$, $2 \leq \alpha \leq 3$.

Ensure: $ES - EDF$ Schedule.

1) Initially, the ready job queue p is empty.
2) At t, all ready jobs are added to p.
3) **while** $p \neq \emptyset$ **do**
4) **while** $C(t) \geq 0$ **do**
5) Select job J_i in p with the highest priority.
6) Calculate the workload $h_k = \sum_{J_i \in p} \frac{C_i}{d_{\max}}$
7) Calculate the intensity $I_j = \max_{j \in \Psi'}(\frac{\sum_{d_i \leq d_j} C_i}{d_j - t})$
8) $S_i = \max(I_j, h_k)$
9) Actual Execution Time $C_i(a) = \frac{C_i}{S_i}$
10) Energy Consumption $e_i = C_i \times S_i^{\alpha - 1}$
11) Calculate the remaining energy in the battery at the end of the execution.
12) $C(t + C_i(a)) = C(t) - e_i$
13) $execute()$
14) **if** Job J_j becomes ready **then**
15) Add J_j to p.
16) **if** J_j becomes the highest priority job **then**
17) Update the speed S_i.
18) Preempt J_i
19) **else**
20) Complete execution of J_i
21) Remove job J_i from p.
22) **end if**
23) **end if**
24) **end while**
25) **end while**

ES-DVFS helps to significantly reduce the processor dynamic energy consumption at the cost of increasing job execution times.

5.4 Illustrative example

Consider a job set $\Psi = \{J_i \mid 1 \leq i \leq 5\}$ with $J_i = (r_i, C_i, d_i)$. Let $J_1 = (0, 4, 16)$, $J_2 = (4, 3, 12)$, $J_3 = (4, 3, 24)$, $J_4 = (0, 4, 14)$ and $J_5 = (9, 1, 20)$.

In line with the previous sections, e_i is considered as the minimum energy consumption relative to the minimum slowdown factor S_i. The power consumption of the processor at slowdown factor S is modeled as $P(S) = aS^{\alpha}$. For sake of simplicity, we assume that $a = 1$ and $\alpha = 2$. Consequently, the time and energy needed to execute the job J_i with processing time C_i at slowdown factor S_i are denoted by $\frac{C_i}{S_i}$ and $E(S_i) = P(S_i) \times \frac{C_i}{S_i} = C_i \times S_i$, respectively. We assume that the battery capacity is $C = 11$ energy units at $t = 0$.

First of all, we have to schedule Ψ according to EDF. We verify that it is not schedulable since the battery capacity is equal to zero at $t = 11$ and consequently the deadline miss rate is about 40%. In detail: At time $t = 0$, J_1 and J_4 are ready. J_4 is the highest priority job and is executed until $t = 4$ where $C(4) = 7$ energy units. At time $t = 4$, J_2 and J_3 are released. J_2 is the highest priority job and is executed until $t = 7$ where $C(7) = 4$ energy units. J_1 is now the highest priority job and is executed until $t = 11$ where battery is fully discharged and consequently the scheduling is terminated where the deadline miss rate is 40% (Fig. 1). To increase the efficiency of the processor and increase energy saving, we have to schedule the same job set Ψ but with ES-DVFS. We find that Ψ is schedulable since all jobs are executed without violating deadlines and without getting out of energy and with energy saving of about 32%.

At time $t = 0$, J_1 and J_4 are ready. They are added to the job queue $\wp$. J_4 is the highest priority job. $h_4 = \sum_{i \epsilon \Psi'} \frac{C_i}{d_{\max}} = \frac{1}{2}$ and $I_4 = \max(\frac{1}{2}, \frac{2}{7}) = \frac{1}{2}$. Thus J_4 is executed at speed $S_4 = \max(I_4, h_4) = \frac{1}{2}$. Consequently, the actual execution time for J_4 is equal to $\frac{C_4}{S_4} = 8$ and its required energy is equal to $C_4 \times S_4 = 2$.

J_4 is now executed until $t = 4$ where J_2 and J_3 are released and added to the queue p. J_4 stops its execution and is preempted by J_2 that is executed with speed $S_2 = \max(\frac{1}{2}, \frac{3}{4}) = \frac{3}{4}$ till $t = 8$ where the energy storage capacity $C(t) = 6.75$ energy units.

At $t = 8$, J_4 must now resume its execution with speed $S_4 = \max(\frac{3}{8}, \frac{3}{4}) = \frac{3}{4}$. Thus $C_4(a) = 3$ and the required energy is equal to $\frac{3}{2}$. J_4 finishes its execution at time $t = 11$ and $C(t) = 5.25$ energy units. It is important to note that J_5 is released at time $t = 9$ and added to the job queue, but it does not preempt J_4 since its deadline is greater than that of J_4.

J_1 is now the highest priority job with a loading factor $h_1 = \frac{1}{3}$. But, its execution time cannot be stretched until $t = 23$ or else the deadline will be violated. Therefore, in order to guarantee that J_1 will still meet its deadlines in the ready queue p, the workload h_1 must be verified in advance by considering the highest speed between h_1 and I_1. As a matter of fact, J_1 is executed with speed $S_1 = \max(\frac{1}{3}, \frac{4}{5}) = \frac{4}{5}$ until $t = 16$ where $C(t) = 2.05$.

At $t = 16$, J_5 is the highest priority job and is executed with speed $S_5 = \frac{1}{2}$ till $t = 18$ where the battery capacity becomes equal to 1.55 energy units.

Now, J_3 is the only job in p. It completes its execution at time $t = 24$ where 0.05 energy units are left in the battery.

5.5 Competitive analysis of ES-DVFS

In general, an on-line algorithm is said to be competitive, if it has a constant competitive factor strictly greater than zero. In this section, we undertake a preliminary competitive analysis of ES-DVFS that is provably optimal in on-line energy-constrained settings. The challenge is to investigate the impact of energy constraints on the competitiveness of on-line real-time scheduling of aperiodic jobs when there is sufficient time but no sufficient energy to complete all the workload.

Before characterizing the competitive analysis theorem, it is important to start with the proposition below.

Proposition 1. ES-DVFS cannot make a total value greater than $C/\widehat{S}^{\alpha}$ where $\widehat{S}$ is the minimum used speed.

Proof. The proof of this proposition can be done by two ways.

First way: Proposition 1 can easily be proved by observing that under ES-DVFS, the energy consumption of each time unit in the schedule assuming minimum speed factor will be S^{α}, so if the amount of energy is C, the maximum total value will be $C/\widehat{S}^{\alpha}$ where $\widehat{S}$ is the minimum speed.

Second way: Let Ψ'_{ij} denote the set of ready jobs in the

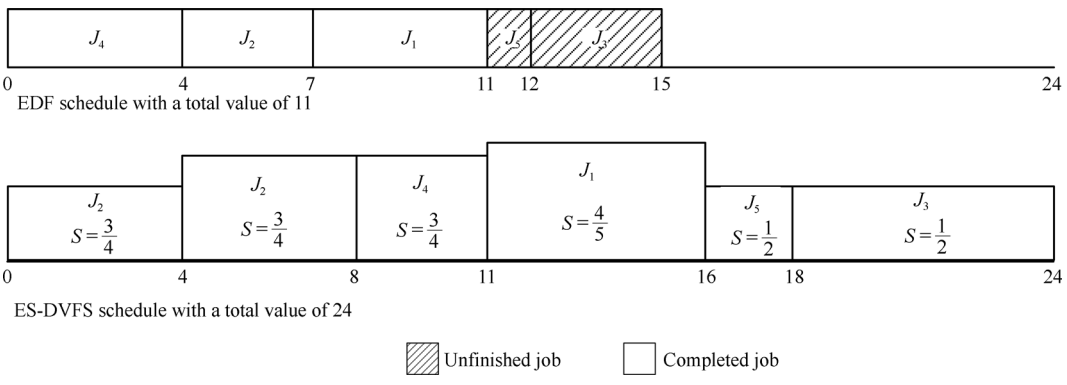

Fig. 1 Schedules generated by EDF and ES-DVFS

interval of time $[t_i, t_j]$ and $t_0, t_1, \cdots, t_n$ as the release times or deadlines of the jobs. Let $S_1, S_2, \cdots, S_N$ be the slowdown factors associated with the execution of jobs in each interval of time $[t_i, t_j]$. Proposition 1 can be proved by observing that under ES-DVFS, a job J_i depletes e_i energy units when executing a workload of $\frac{C_i}{aS_i^{\alpha-1}}$. Thus, with M jobs executed, each with its corresponding slowdown factor S_i, we get $\sum_{i=1}^{M} C_i(S_i)^{\alpha-1} \leq C$.

Now, let us consider the following inequality that can be verified using the Cauchy-Schwartz inequality

$$\sum_{i=1}^{M} C_i(S_i)^{\alpha-1} \leq \sum_{i=1}^{M} \frac{C_i}{S_i} \times \sum_{i=1}^{M} (S_i)^{\alpha}. \tag{6}$$

By making some arrangement and distribution of the terms, we get

$$C \leq \sum_{i=1}^{M} \frac{C_i}{S_i} \times \sum_{i=1}^{M} (S_i)^{\alpha}. \tag{7}$$

Consequently, the total value must be less than or equal to $\frac{C}{\sum_{i=1}^{M}(S_i)^{\alpha}}$.

Now, let us denote by $\widehat{S}$ the minimum speed used during the execution of jobs according to ES-DVFS. This implies that, the upper bound on the total value heavily depends on $\widehat{S}$ and C.

Consequently, a maximum total value of $\frac{C}{\widehat{S}^{\alpha}}$ can be made under ES-DVFS. □

Under ES-DVFS, the pre-knowledge of $\widehat{S}$ is not possible before the end of the schedule. Hence the competitive factor must be computed under this assumption.

Theorem 2. ES-DVFS algorithm has competitive factor of $(S_{\min})^{\alpha}$

Proof. This proof is based on the work done in [25]. Since $\widehat{S}$ is not known in advance, let us consider that a job $J_1(0, \frac{C}{k_1^{\alpha-1}}, \frac{C}{k_1^{\alpha}}, C)$ is released at time $t = 0$. Since J_1 is the only job in the ready queue, it can be executed at frequency S_1 equal to the intensity and workload. Thus $S_1 = \frac{C_1}{d_1} = k_1$. ES-DVFS is forced to execute J_1 gathering a total value of $(\frac{C}{k_1^{\alpha-1}}) \times (\frac{1}{k_1}) = \frac{C}{k_1^{\alpha}}$. Consequently, the competitive factor is equal to $C \times \frac{k_1^{\alpha}}{C} = k_1^{\alpha}$. But k_1 must be bounded between $S_{\min}$ and 1. Hence, ES-EDF can force an upper bound of $(S_{\min})^{\alpha}$ when $k_1 = S_{\min}$ and consequently the competitive factor of ES-DVFS is $(S_{\min})^{\alpha}$. □

6 Semi-on-line algorithm ES-DVFS*

On-line algorithms, where partial information about the input is given to the scheduler in advance, are called semi-on-line algorithms. In our work, the pre-knowledge of $\widehat{S}$ can potentially improve the competitiveness of the ES-DVFS algorithm.

In the semi-on-line version of ES-DVFS algorithm, same scheduling steps are performed except that ES-DVFS* uses the knowledge of the minimum speed used, namely $\widehat{S}$, so as to enhance the competitive factor.

Theorem 3. ES-DVFS* algorithm has competitive factor not greater than $(\widehat{S})^{\alpha}$.

Proof. We repeat the proof of theorem 2, but with slight modifications. Le us consider that a job $J_1(0, \frac{C}{k_2^{\alpha-1}}, \frac{C}{k_2^{\alpha}}, C)$ is released at time $t = 0$. J_1 is the only job in the ready queue, it can be executed at frequency S_1 equal to the intensity and workload. Thus $S_1 = \frac{C_1}{d_1} = k_2$. ES-DVFS** is forced to execute J_1 gathering a total value of $\frac{C}{k_2^{\alpha-1}}$. Since $\widehat{S}$ is known in advance then k_2 must be bounded between $\widehat{S}$ and 1. Hence, ES-DVFS* can force an upper bound of $(\widehat{S})^{\alpha}$ when $k_2 = \widehat{S}$. Consequently the competitive factor of ES-DVFS* is $(\widehat{S})^{\alpha}$. □

ES-DVFS* can be further enhanced by considering that it uses the knowledge of the maximum job size in addition to $\widehat{S}$. Let us denote it by ES-DVFS**. ES-DVFS** is a variant of ES-DVFS where the maximum job size and $\widehat{S}$ are known in advance by the input.

First of all we will consider the largest job size for ES-DVFS that will be later used by ES-DVFS**.

Lemma 2 establishes the competitive factor of ES-DVFS as a function of the maximum speed used by the scheduler $\vec{S}$. □

Lemma 2. ES-DVFS has a competitive factor of 0.5 if the largest job size $C_L \leq \frac{1}{2}(\frac{C}{(\vec{S})^{\alpha-1}})$.

Proof. In ES-DVFS, and before executing a job, we should use the minimum CPU speed available to stretch out the WCET as much as possible without violating deadlines. Therefore, job J_i is executed with a speed $S_i = \max(\frac{C_i}{d_i - t}, h_k)$ if the energy storage unit has sufficient energy for this execution. Thus, at time t, EC-DVFS executes a job $J_k(t, C_k, d_k, e_k)$ if and only if

$$C_r \geq (S_k)^{\alpha-1} C_k \tag{8}$$

where C_r is the remaining energy in the energy storage unit.

Now, let us consider that ES-DVFS schedules n jobs where the energy storage unit is not fully discharged and all deadlines are met. At time t, a job J_k is released and cannot be executed because of energy starvation. This implies that $C_r < C_k(S_k)^{\alpha-1}$. Denote the minimum size of any job by ξ units. This means that $C_r = C_k(S_k)^{\alpha-1} - \xi$. Thus, ES-DVFS consumed at least C' energy units, where

$$C' = C - C_r = C - C_k(S_k)^{\alpha-1} + \xi. \tag{9}$$

The workload admitted till time t cannot exceed $\frac{C'}{(\widehat{S})^{\alpha}}$ and consequently, ES-DVFS guarantees a value not greater than $\frac{C'}{(\widehat{S})^{\alpha}}$.

But since C_L is the largest job size, then $C_k \leq C_L \leq \frac{1}{2}(\frac{C}{(\vec{S})^{\alpha-1}})$. This implies that

$$\frac{C'}{(\widehat{S})^{\alpha}} = \frac{C - C_k(S_k)^{\alpha-1} + \xi}{(\widehat{S})^{\alpha}} \geq \frac{C - C_L(S_k)^{\alpha-1} + \xi}{(\widehat{S})^{\alpha}}$$

$$\frac{C'}{(\widehat{S})^{\alpha}} \geq \frac{C - (\frac{1}{2}(\frac{C}{(\vec{S})^{\alpha-1}})(S_k)^{\alpha-1}) + \xi}{(\widehat{S})^{\alpha}}$$

$$\frac{C'}{(\widehat{S})^{\alpha}} \geq \frac{C - \frac{1}{2}(\frac{S_k}{\widehat{S}})^{\alpha-1}C + \xi}{(\widehat{S})^{\alpha}}.$$

In addition, it is not difficult to see that $\vec{S} \geq S_k$. This means $\frac{S_k}{\vec{S}} \leq 1$. By making some arrangement and distribution of the terms, we get

$$\frac{C'}{(\widehat{S})^{\alpha}} \geq \frac{1}{2}\left(\frac{C}{(\widehat{S})^{\alpha}}\right).$$

Further, ES-DVFS can make a value not greater than $\frac{C}{(\widehat{S})^{\alpha}}$. Hence, the competitive factor of ES-DVFS is equal to

$$\frac{\frac{1}{2}\frac{C}{(\widehat{S})^{\alpha}}}{\frac{C}{(\widehat{S})^{\alpha}}} = \frac{1}{2}.$$

□

Lemma 3. ES-DVFS** has a competitive factor of 0.5.

Proof. ES-DVFS** is the extension of ES-DVFS where it knows in advance the minimum used speed during job execution ($\widehat{S}$) and the largest job size (S_L). We point out the competitive analysis of ES-DVFS** by comparing S_L to $\frac{1}{2}(\frac{C}{(\widehat{S})^{\alpha-1}})$.

Case 1. If $C_L \leq \frac{1}{2}\frac{C}{(\widehat{S})^{\alpha-1}}$, then we underline that ES-DVFS** follows the rules of ES-DVFS as stated in Lemma 2. Consequently, the competitive of ES-DVFS** is 0.5.

Case 2. If $C_L > \frac{1}{2}\frac{C}{(\widehat{S})^{\alpha-1}}$, then we should be ready to execute the job J_L with speed $S_L = \max(\frac{C_L}{d_L - t}, h_k)$ that is beyond $\widehat{S}$ to guarantee that the energy requirement cannot exceed the energy storage capacity. This means that ES-DVFS** attains at least a value of $\frac{1}{2}\frac{C}{(\widehat{S})^{\alpha}}$. Further, the optimal on-line algorithm can attain a value not greater than $\frac{C}{(\widehat{S})^{\alpha}}$. Consequently, ES-DVFS** has a competitive factor of 0.5.

From the above steps, we can deduce that the competitive factor of ES-DVFS** is equal to 0.5. □

7 Performance evaluation

In this section, we apply ES-DVFS to several job sets and compare the performance of our research by two simulation experiments: percentage of feasible job sets and ratio of energy savings. For brevity, we implement ES-DVFS, EDF and EDF*. EDF* is an enhanced version of EDF in a way that jobs are slacked by the same slowdown factor $S = \sum_i \frac{C_i}{d_{\max}}$, where $d_{\max}$ is the largest deadline.

7.1 Experimental setup

The simulation environment consists of a simulation kernel (scheduler) with a number of components involved in the management and analysis of simulations. The main components are: task generator, scheduler and CPU.

We implemented the proposed scheduling techniques in a discrete event simulator using C/C++. To evaluate the effectiveness of the ES-DVFS algorithm, we consider a job generator of aperiodic jobs. It accepts as input several parameters: the number of desired jobs n and processor load L_p. At the output, we obtain a jobs configuration $\Psi = \{J_i(r_i, C_i, d_i, e_i) \mid 1 \leq i \leq n\}$. Jobs are released at random times that is not necessarily at $t = 0$. The execution times of jobs are randomly generated such that $L_p = \sum_{i=1}^{n} \frac{C_i}{d_{\max}} \leq 1$.

The results shown are from 100 randomly generated synthetic job sets, each containing 30 jobs with maximum deadline equal to 3 360. Deadlines are greater than or equal to the computation times ($C_i \leq d_i$). We assume that the power consumption of a processor is a quadratic function of the processor speed. $P(S) = aS^{\alpha}$ where a is considered as a constant characterized by the processor parameters and equal to 1 and α is equal to 2.

After the generation of jobs, the simulator order them on-line for each scheduling algorithm. Simulation results are then ordered to excel files to be stored and analyzed.

7.2 Percentage of feasible job sets by varying the processor load

As mentioned above, when we attempt to dynamically reduce the CPU speed, we risk exceeding worst-case completion times of the scheduled jobs and thus violating deadlines. In this study, we adopt an approach that takes interest in the percentage of job sets which are feasible with ES-DVFS, EDF and EDF*. We consider the settings where jobs exhibit their worst case workload. The impact of processor load on the schemes are shown. We report the results of this simulation study where the processor load L_p is scaled from 0.1 till 1.

Under ES-DVFS, we can assure that the completion time of the currently ready job will not extend beyond its deadline. This is because, through dynamic speeds, an utmost care is taken in order to guarantee the timely completions of all the jobs in the ready queue. However, this case is not evident in EDF*.

At lower load values, ES-DVFS effectively benefits from the long idle times to prolong the execution periods where the processor is in continuous use. However, the EDF algorithm runs at full processor speed and does not utilize DVS technique to save energy and EDF* slacks jobs with equal slowdown factor and consequently not all deadlines will be met. According to Fig. 2, ES-DVFS exceeds that of EDF* and EDF by about 31% and 44% respectively.

As the processor load increases, the percentage of feasible jobs sets in ES-DVFS decreases. This decrease is due to the fact that the likelihood of having large idle time is lower under high processor load. Since the large idle time adversely affects and limits the reclaiming opportunities, the performance is degraded, though marginally. Although ES-DVFS incurs higher deadline miss rate but still exceeds that of EDF* and EDF by about 26% and 29%.

It is important to note that when the processor load is set to one, the processor is always active and there is no processor idle time and consequently ES-DVFS, EDF* and EDF will have the same percentage of feasible job sets.

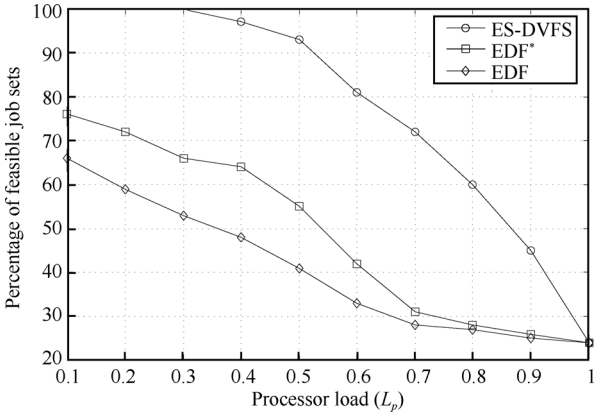

Fig. 2 Percentage of feasible job sets by varying L_p

7.3 Percentage of feasible job sets by varying the number of jobs

To examine the effect of the number of jobs on the percentage of feasible job sets, we performed simulations by varying the number of jobs from 5 to 40. The processor load considered in this simulation is $L_p = 0.5$. For brevity, we use the trends similar to that of 30-job systems. Fig. 3 depicts that the percentages do differ from one case to the other. In particular, we see that the percentage of feasible jobs sets provides a slightly larger advantage when the number of jobs increases because there are more opportunities for reclaiming unused slack-times as well as for aggressively reducing the CPU speed with increasing number of jobs.

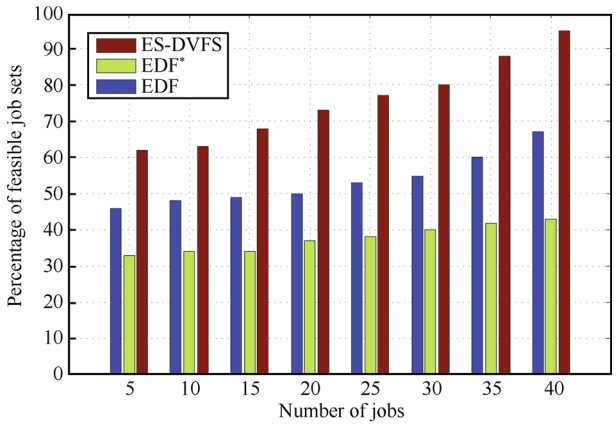

Fig. 3 Percentage of feasible job sets by varying the number of jobs

Comparing the mechanism of our algorithm to those of EDF* and EDF schemes, we can make the following observations: when increasing the number of jobs, the percentage of feasible job sets increases by about 35%, 31% and 23% respectively for ES-DVFS, EDF* and EDF. Moreover, as the number of jobs increases, the percentage of feasible job sets for ES-DVFS exceeds that of EDF* and EDF by an average of 23% and 15% respectively.

7.4 Ratio of energy saving

In this section, we measure the energy savings by comparing ES-DVFS, EDF* and EDF. The simulation methodology is parallel to the one described in Section 7.2. In this experiment, we measure the normalized energy gains, that means the quantity of energy gains relative to battery capacity. From the beginning, we observe that there is no energy saving in EDF scheme since it operates at maximum processor frequency.

Fig. 4 shows that in decreasing order of performance with respect to energy savings, the algorithms can be arranged as: ES-DVFS, EDF* and EDF.

At low processor load values, the gain in energy saving provided by ES-DVFS scheme is significant (about 65%) and that exceeds EDF* by 40%. This is due to the fact that, when ES-DVFS is run with low processor load, the CPU will operate at low speed levels, and consequently energy consumption will decrease.

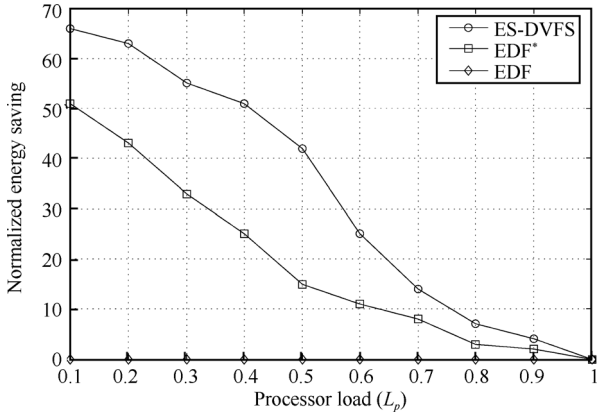

Fig. 4 Normalized energy saving by varying the number of jobs

Yet, as we increase the processor load and move toward more active processor levels, the dynamic energy consumption of the workload dominates and hence the benefits of energy saving decrease. In fact, when the processor load exceeds 60%, ES-DVFS forced to consume more energy to avoid deadline violations that can result in low system performance.

8 Conclusions

In this paper, we addressed the problem of determining the optimal speed schedule for embedded systems which are implemented on a variable speed processor and employs a dynamic priority scheme to schedule aperiodic jobs. We presented an optimal energy efficient scheduling algorithm, named ES-DVFS, to reduce the CPU energy consumption. Unlike prior studies, our algorithm provides sound dynamic speed reduction mechanisms. This means that the speed of the jobs ready to be executed is dynamically adjusted on the fly. By deriving the dynamic adjustment properties from this characterization, we demonstrated that ES-DVFS cannot attain a total value greater than $C/\widehat{S}^{\alpha}$. Further, by

assuming the knowledge of the largest job size and the minimum speed used in the schedule, we presented an optimal semi-on-line algorithm EC-DVFS**, which has a competitive factor of 0.5. To the best of our knowledge, this is the first work that investigates minimizing energy consumption under dynamic speed mechanisms.

Our simulation results show that, when the processor workload is low, ES-DVFS can save up to 65 % of the energy over a modified EDF algorithm (EDF*), which takes a constant speed all over the schedule. Our algorithm offers also a consistent advantage over EDF* and EDF in the percentage of feasible job sets. The experiments confirmed that a "dynamic" slowdown factor that aims to achieve the best energy saving would be optimal under the different processor workload and yields the best results.

We hope that this research work will trigger further research efforts. In particular, obtaining the best energy saving for homogeneous and heterogeneous multiprocessor systems will be an interesting research direction.

References

[1] H. Zeng, C. S. Ellis, A. R. Lebeck, A. Vahdat. ECOSystem: Managing energy as a first class operating system resource. In *Proceedings of the 10th International Conference on Architectural Support for Programming Languages and Operating Systems*, ACM, New York, USA, pp. 123–132, 2002.

[2] M. Weiser, B. Welch, A. Demers, S. Shenker. Scheduling for reduced CPU Energy. In *Proceedings of the 1st USENIX Conference on Operating Systems Design and Implementation*, ACM, Berkeley, USA, 1994.

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

[4] S. Baruah, G. Koren, D. Mao, B. Mishra, A. Raghunathan, L. Rosier, D. Shasha, F. Wang. On the competitiveness of on-line real-time task scheduling. In *Proceedings of the 12th IEEE Real-time Systems Symposium*, IEEE, San Antonio, USA, pp. 106–115, 1991.

[5] S. K. Baruah, J. R. Haritsa. Scheduling for overload in real-time systems. *IEEE Transactions on Computers*, vol. 46, no. 9, pp. 1034–1039, 1997.

[6] M. Spuri, G. Buttazzo. Scheduling aperiodic tasks in dynamic priority systems. *Real-time Systems*, vol. 10, no. 2, pp. 179–210, 1996.

[7] C. L. Liu, J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[8] J. Y. T. Leung, J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, vol. 2, no. 4, pp. 237–250, 1982.

[9] M. L. Dertouzos. Control robotics: The procedural control of physical processes. In *Proceedings of International Federation of Information Processing Congress*, pp. 807–813, 1974.

[10] H. Chetto, M. Chetto. Some results of the earliest deadline scheduling algorithm. *IEEE Transactions on Software Engineering*, vol. 15, no. 10, pp. 1261–1269, 1989.

[11] H. Chetto, M. Silly, T. Bouchentouf. Dynamic scheduling of real-time tasks under precedence constraints. *Real-time Systems*, vol. 2, no. 3, pp. 181–194, 1990.

[12] F. Yao, A. Demers, S. Shenker. A scheduling model for reduced CPU energy. In *Proceedings of the 36th IEEE Annual Symposium on Foundations of Computer Science*, IEEE, Milwaukee, USA, pp. 374–382, 1995.

[13] I. Hong, M. Potkonjak, M. B. Srivastava. On-line scheduling of hard real-time tasks on variable voltage processor. In *Proceedings of the 1998 IEEE/ACM International Conference on Computer-Aided Design*, IEEE, San Jose, USA, pp. 653–656, 1998.

[14] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, M. Srivastava. Power optimization of variable voltage core-based systems. In *Proceedings of IEEE Design Automation Conference*, IEEE, San Francisco, USA, pp. 176–181, 1998.

[15] Y. Shin, K. Choi. Power conscious fixed priority scheduling for hard real-time systems. In *Proceedings of the 36th IEEE Design Automation Conference*, IEEE, New Orleans, USA, pp. 134–139,1999.

[16] H. Aydin, R. Melhem, D. Mosse, P. Mejia-Alvarez. Determining optimal processor speeds for periodic real-time tasks with different power characteristics. In *Proceedings of IEEE the 13th EuroMicro Conference on Real-Time Systems*, IEEE, Delft, Netherlands, pp. 225–232, 2001.

[17] R. Jejurikar, R. Gupta. Optimized slowdown in real-time task systems. In *Proceedings of the 16th IEEE EuroMicro Conference on Real-time Systems*, IEEE, Catania, Italy, pp. 155–164, 2004.

[18] P. Pillai, K. G. Shin. Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the 8th ACM Symposium on Operating Systems Principles*, ACM, New York, USA, pp. 89–102, 2001.

[19] H. Aydin, R. Melhem, D. Mosse, P. Mejia-Alvarez. Power-aware scheduling for periodic real-time tasks. *IEEE Transactions on Computers*, vol. 53, no. 5, pp. 584–600, 2004.

[20] L. Thiele, S. Chakraborty, A. Maxiaguine. DVS for buffer-constrained architectures with predictable QoS-energy tradeoffs. In *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, IEEE, Jersey City, USA, pp. 111–116, 2005.

[21] N. Bansal, H. L. Chan, T. W. Lam, L. K. Lee. Scheduling for speed bounded processors. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming*, Part I, Springer, Reykjavik, Iceland, vol. 5125,pp. 409–420, 2008.

[22] H. K. Chen, X. M. Zhu, H. Guo, J. H. Zhu, X. Qin, J. H. Wu. Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment. *Journal of Systems and Software*, vol. 99, pp. 20–35, 2015.

[23] H. E. L. Ghor, E. H. M. Aggoune. Energy saving EDF scheduling for wireless sensors on variable voltage processors. *International Journal of Advanced Computer Science and Applications*, vol. 5 no. 2, pp. 158–167, 2014.

[24] J. W. S. W. Liu. *Real-time Systems*, NJ, USA: Prentice Hall, 2000.

[25] V. Devadas, F. Li, H. Aydin. Competitive analysis of online real-time scheduling algorithms under hard energy constraint. *Real-time Systems*, vol. 46, no. 1, pp. 88–120, 2010.

Hussein El Ghor received the engineering degree from the Lebanese University, Lebanon in 2002. He also received the Ph. D. degree in automatics and applied informatics from the University of Nantes, France in 2012. He is currently a member of the Sensor Networks and Cellular Systems Research Center, University of Tabuk, Tabuk, Saudi Arabia. He authored many papers in prestigious journals and conferences.

His research interests include real-time scheduling and partitioning with particular emphasis on energy efficiency and energy harvesting systems.

E-mail: hussein.ghor@ul.edu.lb (Corresponding author)

El-Hadi M. Aggoune received his M. Sc. and Ph. D. degrees in electrical engineering from the University of Washington, USA. He is a professional engineer registered in the State of Washington, and senior member of the Institute of the IEEE. He has taught graduate and undergraduate courses in electrical engineering at a number of universities in the US and abroad. He served at many academic ranks including endowed chair professor and vice president and provost. He was the winner of the Boeing Supplier Excellence Award. He was also the winner of the IEEE professor of the Year Award, UW Branch. He is listed as inventor in a major patent assigned to the Boeing Company. His research work is referred to in many patents including patents assigned to ABB, Switzerland and EPRI, USA. Currently he is a professor and director of the Sensor Networks and Cellular Systems Research Center, University of Tabuk, Tabuk, Saudi Arabia. He authored many papers in IEEE and other journals and conferences. He is serving on many technical committees.

His research interests include modeling and simulation of large scale networks, sensors and sensor networks, scientific visualization, and control and energy systems.

E-mail: haggoune.sncs@ut.edu.sa