The CSI Journal on
Computer Science and Engineering
Vol. 7, No. 2 & 4 (b), 2009
Pages 34-47
Regular Paper

Energy-Aware Scheduling of Execution-Instant Sensitive Real-Time Systems

Leili Farzinvash[1] Mehdi Kargahi[1,2]

[1]School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran
[2]School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

Abstract

Time constraints in real-time systems are traditionally identified based on the job completion-times, i.e. using classic deadlines or time/utility functions (TUFs). In this paper, we consider a new model of time constraints, namely Instant-Value-Function (IVF) which specifies the most appropriate instants of time to execute a job. More precisely, we see that in many applications in the embedded world, the exact instants of time within which a job is executed affect the value that the job accrues. This property cannot be expressed by classic deadlines or TUFs. In this paper, we consider simultaneous timeliness and energy optimization of battery-operated IVF-constrained real-time embedded systems. For energy optimization, we discuss the semantics of applying dynamic voltage scaling (DVS) to such energy-bounded systems. We see that, in spite of the completion-time dependent time constraint models, the shape of job IVF is affected by changing the speed of the system processor. Accordingly, we present a DVS-based scheduling algorithm to maximize the accrued value of the energy-bounded real-time systems. The optimality of this algorithm is established analytically, while its effectiveness is also confirmed through simulation experiments.

Keywords: Dynamic Voltage Scaling (DVS), Energy-Bounded Systems, Instant-Value-Function (IVF), Real-Time Systems, Value Accrual Scheduling.

1. Introduction

A broad range of computing and control systems such as those related to applications in critical domains (e.g., the space and defense) as well as in less critical domains (e.g., the multimedia) are time-sensitive. Also, many such systems are embedded and mobile, and thus, are very resource constrained. Traditionally, the time sensitivity of a broad category of such real-time embedded systems is defined by the classic deadline [1]. Accordingly, while a job completes before its deadline, there is no difference among the instants of time where the job is executed or even completed. However, as mentioned in a number of studies [2-4], deadline by itself cannot express the timing characteristics of many real-time systems. These studies propose a generalization of the classic deadline, i.e. time utility function (TUF), which specifies the utility that a job can accrue as a function of its completion time.

As another generalization on the time constraint models from a different aspect, we can mention the gravitational model introduced in [5-6]. According to this model, each job, in addition to its deadline, has a target point of execution at which the job execution results in the highest utility. The model defines a certain degree of flexibility showing the relatively lower accruable utility of executing the job around the target point. Examples of such applications include control and media processing. As described more precisely in Section 2, only some limited time constraint models for non-preemptive tasks, which are targeted to be executed as close as possible to some specific instants of time, can be defined in this manner.

In summary, none of the above timing models concentrates to the exact instants of time for job execution, rather they are normally sensitive to the instant of job completion. However, time constraints of some embedded applications could not precisely be expressed by the previous models; rather, a new timing model is needed. In [7], the same authors proposed a different model of timing constraints to consider the importance of instants of time that the job executes.

More precisely, the proposed model in the study, namely Instant Value Function (IVF), specifies the value that each job obtains as a function of the exact instants of time within which the job executes. In embedded systems with sensors and actuators, IVFs can be employed to specify the most desired instants of time for gathering the sensor information, as well as the most desired instants of time for doing the respective computations based on the received information, to be able to prepare the best results through the actuators [8-9]. On the other hand, energy consumption is one of the most important design constraints of real-time embedded systems, especially for battery-operated portable devices which use limited sources of energy. In the last decade, literally many studies have been done on applying dynamic voltage scaling (DVS) to real-time systems. Most of them have been focused on the deadline time constraint [10-15], and a few on the interplay of TUF and DVS [16-19] (More descriptions on the matter are provided in Section 7). However, in this paper, we study the issues related to the semantics of applying DVS to jobs with the IVF time constraints.

1.1. Contributions

The main contributions of this paper are as follows: 1) It presents the precise definition of a new type of timing constraints called Instant Value Function (IVF) (A preliminary version of the current study [7] introduced the concept of IVF for the first time); 2) This paper explains the semantics of applying DVS to IVF-constrained real-time embedded systems. Past efforts on energy-efficient real-time scheduling algorithms consider the category of real-time systems which are sensitive to the job completion times, whereas IVF is concerned with the execution instants of jobs. To the best of the authors' knowledge, we are unaware of any previous study for applying power saving issues to execution-instant sensitive models; 3).

The paper provides a DVS-based scheduling method to maximize the accrued value of the IVF-constrained real-time embedded systems regarding their bounded sources of energy. Using this model, we also present a method to produce an efficient schedule for a set of IVF-constrained tasks to maximize the accrued value in the system with a bounded source of energy; 4). The optimality of the proposed algorithm is also proved analytically.

The rest of this paper is organized as follows. The IVF time constraint is explained in Section 2. In Section 3, the system model and the respective problem description are presented. The behavior of IVF with respect to frequency changes is discussed in Section 4. We explain our scheduling algorithm for energy-constrained IVF-based systems in Section 5. Experimental results are presented in Section 6. In Section 7, we discuss some related works. Finally, we conclude the paper in Section 8.

2. Instant Value Functions

In this section, we propose IVF as a different model of timing constraints which is concerned with the instants of job execution rather than the instant of job completion. IVF specifies the value that each job can obtain as a function of the exact instants of time at which the job executes. Accordingly, the total accrued value of a job is equal to the area under its IVF, corresponding to the instants of time allocated to that job execution.

Figure 1 shows a few samples: The step IVF presented in Figure 1(a) specifies that as long as the deadline can be met, there is no difference among the instants of time allocated to the job execution. According to Figure 1(b), it is desirable to postpone the execution of a job insofar as possible. Based on Figure 1(c), early execution of the job yields in gaining more value. The most desirable instants of job execution in Figure 1(d) are around the middle of the job's life-time.

More precisely, the semantics of the latter IVF can better be clarified by Figure 2, where a sample job is split into two segments. The job starts its execution at t_s, is preempted at t_p, resumes its execution at t_r, and finally is completed at t_c. The accrued value by this job is equal to the summation of the gray areas, i.e., the total area under the respective IVF at the exact instants of the job execution.

To have a better insight in the semantics of IVF, we consider an operational supervisory system [20] as an example. This system is designed to address the problem of interception of some moving objects. In [20], two objectives are defined for this system: 1) intercepting a set of objects which pass a boundary and prior to their crossing to another specific boundary, 2) the interceptions should occur as far away from the boundaries as possible. A number of remote sensors track the position of the moving objects and report the updated coordinates to the interceptors.

In a system that the objects do not necessarily follow a predictable moving pattern (e.g. the speed and acceleration of them change rapidly), it is desirable to obtain the most updated sensor information to determine the optimal intercepting instants. The most desirable instants of information gathering for optimal interception can effectively be specified by the respective IVF.

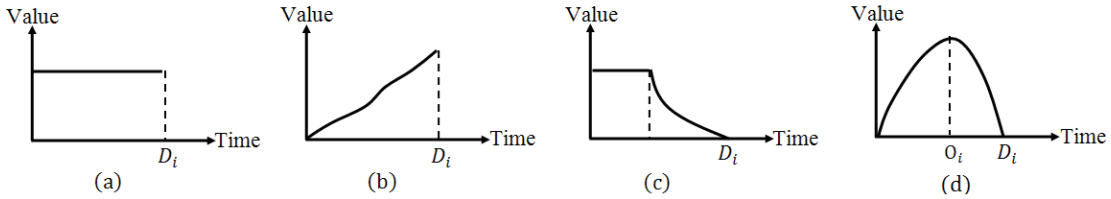

Figure 1. Sample IVFs

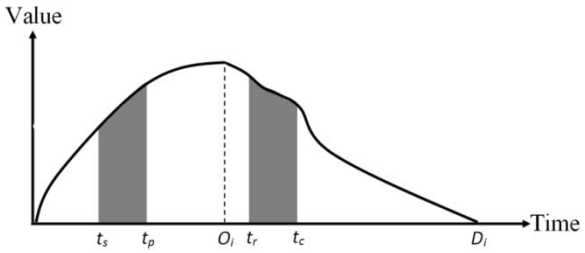

Figure 2. A schematic view of accrued value by a job

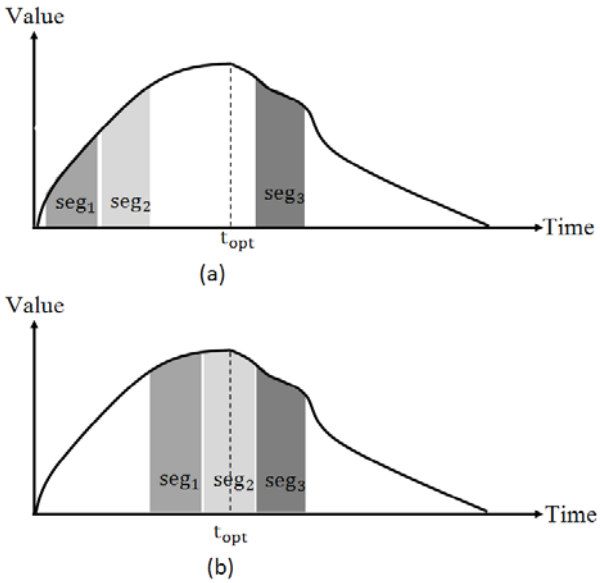

Figure 3. Two schedules with different accruing values for a job

Assuming the IVF shown in Figure 2, two schedules shown in Figure 3 could be compared. In Figure 3(a), the information has been gathered so early by the first two segments of the job, such that intercepting suffers from the lack of enough precision due to inappropriate and/or non-updated positioning data. On the other hand, if the execution of different segments of the job is done according to Figure 3(b), because of more updated sensor data, considerable improvements may be achieved in the result of the intercepting operation.

Therefore, according to the IVFs of the system jobs, the exact instants of job execution have considerable effects on the accuracy (value) of the results. Based on the instants, the IVF model can be used for describing the resulting behavior of embedded systems that interact to the environment through sensors/actuators. In such systems, the exact instants of gathering sensor data, doing the respective processing/computation, and actuation, all are of high importance and directly affect the accuracy of the system service.

3. System Model

3.1. Processor and Application Models

We consider a single DVS-enabled processor whose frequency fr can vary continuously between a lower bound fr_{min} and an upper bound fr_{max}. The ratio of fr_{max} to fr_{min} is shown by *max-ratio*. The operating frequency at t is represented by $fr(t)$.

The intended system is composed of a set of periodic preemptive real-time tasks denoted by $T = \{T_1, T_2, \ldots, T_n\}$. Furthermore, tasks are independent of each other, and do not share any non-CPU resources. The task T_i is characterized by a pair (p_i, e_i), where p_i represents its period and e_i denotes the worst case execution time of T_i at fr_{max}. Each task has a sequence of jobs. The j^{th} job of T_i is shown by $J_{i,j}$.

The arrival time and deadline of $J_{i,j}$ are given by $r_{i,j}$ and $d_{i,j}$, respectively, where $r_{i,j}$ is equal to $(j-1).p_i$, and $d_{i,j}$ is equal to $j.p_i$. The worst case execution time of $J_{i,j}$ is equal to e_i, and is shown by $e_{i,j}$. The hyper-period of the tasks is shown by H. There exist l jobs in H, where $l = \sum_{i=1}^{n} H/p_i$.

As the system is preemptive, $J_{i,j}$ can be split into some arbitrary number of segments. The k^{th} segment of $J_{i,j}$ is denoted as $J_{i,j,k}$. The length of $J_{i,j,k}$ at operating frequency fr is represented by $\left|J_{i,j,k}^{fr}\right| = (fr_{\max}/fr).\left|J_{i,j,k}^{fr_{\max}}\right|$. The starting point of $J_{i,j,k}$ is shown with $s_{i,j,k}$, and the finishing point of it is denoted as $f_{i,j,k}$. The sum of the lengths of $J_{i,j}$ segments executing at fr_{max} is equal to $e_{i,j}$, i.e, if $J_{i,j}$ splits into m segments, we have $e_{i,j} = \sum_{k=1}^{m_{i,j}} \left|J_{i,j,k}^{fr_{\max}}\right|$.

3.2. Energy Consumption Model

We consider a system with a limited source of energy that relies on battery and has an energybudget bound,E_b. As recharging the battery is infeasible in the considered system, it is subject to a hard energy constraintE_b.

When the processor operates at frequencyfr, the dynamic power consumption of the processor, denoted asP_d, is given by $P_d = C_{ef}.V_{dd}^2.fr$, whereC_{ef} is the effective switch capacitance and V_{dd} is the supply voltage. On the other hand, $fr = k.(V_{dd} - V_t)^2 / V_{dd}$, where k is constant and V_t is a threshold voltage [21]. As it is done in [18], we approximate this expression by $fr = \beta.V_{dd}$, where β is a constant. Thus, we obtain $P_d = (C_{ef} / \beta^2).fr^3$. This expression is equivalent to $P_d = \alpha.fr^3$, where $\alpha = C_{ef} / \beta^2$.

Iffr is constant during the execution of$J_{i,j,k}$, the relationship between the consumed energy and P_d can be described as $E_{i,j,k} = \left| J_{i,j,k}^{fr} \right|.P_d$. Here $E_{i,j,k}$represents the consumed energy during the execution of$J_{i,j,k}$. Further, the consumed energy in$H$ is denoted asE_H, which is equal to the consumed energy by the job segments in$H$, i.e., $E_H = \sum_{i=1}^{n} \sum_{j=1}^{H/p_i} \sum_{k=1}^{m_{i,j}} E_{i,j,k}$.

As the system is operating on the bounded energyE_b, the system life-time is limited to a bounded number of hyper-periods N_H. Using E_b and E_H, N_H can be calculated as $N_H = E_b / E_H$. As $E_H \ll E_b$, N_H can be approximated by an integer.

3.3. Value Function Model

The value of instantaneous execution of taskT_i at time t at operating frequency fr is specified by an IVF such as$V_i^{fr}(t)$. This function is defined for a task; thus, all jobs of the task have the same value function. Accordingly, we use$V_{J_{i,j}}^{fr}(t)$ as the value function of$J_{i,j}$. We refer to the maximum value of $V_i^{fr_{\max}}(t)$ as $V_i^{\max}$. Usually, $V_i^{fr_{\max}}(t)$ is given as a task property. However, we can calculate $V_i^{fr}(t)$ for different frequencies based on the value of $V_i^{fr_{\max}}(t)$. More details on this subject will be presented in Section 4.

Note that $V_i^{fr}(t)$is defined for a task, but $V_{J_{i,j}}^{fr}(t)$ specifies a job time constraint. Therefore,$V_i^{fr}(t)$ is defined in [0, p_i], while$V_{J_{i,j}}^{fr}(t)$ is defined in the range of [$r_{i,j}$,$d_{i,j}$]. Thus, the relationship between$V_i^{fr}(t)$ and$V_{J_{i,j}}^{fr}(t)$ can be expressed as $V_{J_{i,j}}^{fr}(t) = V_i^{fr}(t - (j-1).p_i)$.

3.4. Problem Description

In this paper, we consider hard real-time systems, i.e, systems in which all jobs must be executed completely before their deadlines. When time constraints are described by IVFs, the scheduling objective is to maximize the total gained value of the system through executing the jobs as close as possible to their optimal instants of execution. As the system is energy bounded, the goal of maximizing accrued value in H would not necessarily lead to gain the maximum possible value in the system life-time. To get the maximum value in$H$, jobs have to be split into some segments and each job segment must be executed as close as possible to the optimal instants of execution at operating frequency offr_{max}. Doing so, the system consumes lots of energy, and$N_H$is decreased. Thus, there must be a tradeoff between the accrued value in$H$ and the value ofN_H. In this case, the problem is to execute jobs in order to maximize the total accrued value through the system life-time while meeting all the deadlines.

To better formalizing the problem, we define a new metric, called EAV (Execution Accrued Value), to measure the amount of value gained in the system life-time. We refer to the accrued value of$J_{i,j}$ as $EAV_{J_{i,j}}$, which is formally defined as follows:

$$EAV_{J_{i,j}} = \sum_{k=1}^{m_{i,j}} \int_{s_{i,j,k}}^{f_{i,j,k}} V_{J_{i,j}}^{fr}(t)\,dt \tag{1}$$

By this definition, the scheduling objective is to maximize the total obtained *EAV* of the jobs throughout the system life-time. Thus, through segmenting the jobs and finding$fr(t)$, the scheduling objective can be formulated as follows:

$$\max\left(\sum_{i=1}^{n} \sum_{j=1}^{H/p_i} EAV_{J_{i,j}} \right) \times N_H$$

$$\begin{aligned}
&\textit{subject to} \\
&\sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{\max}} \right| = e_{i,j} && i = 1,\ldots,n \\
& && j = 1,\ldots,H/p_i \\
&r_{i,j} \le s_{i,j,k} \le f_{i,j,k} \le d_{i,j} && i = 1,\ldots,n \\
& && j = 1,\ldots,H/p_i \\
& && k = 1,\ldots,m_{i,j} \\
&fr_{\min} \le fr(t) \le fr_{\max}
\end{aligned} \tag{2}$$

The 2nd constraint in (2) guarantees that each job segment executes completely between its release time and deadline.

4. Speed Scaling and IVF

In the last decade, power and energy saving issues in real-time systems have been extensively studied [12-14, 16, 18]. As described in Section 7, most of the previous attempts have been done on the deadline time constraint [12-14]. There is also few studies on the interplay of TUF and DVS [16, 18].

As can be derived from the definitions of deadline and TUF, DVS has no effect on the shape and characteristics of these types of time constraints. However, changing frequency directly affects the IVF shape. Suppose that atfr_{max}, an amount of work of size$I$ work can be executed in the time interval t. Now, if the operating frequency is switched to $fr\,(fr \le fr_{\max})$, the amount of work that can be

executed in Δt is diminished to $I.(fr/fr_{\max})$. Accordingly, the accrued value in Δt also decreases with the ratio of $fr/fr_{\max}$. Because IVF is defined as the accruable value of executing a job at instant*t*, the accruable value in time*t* decreases in line with the frequency as follows:

$$V^{fr}(t)=\frac{fr}{fr_{\max}}.V^{fr_{\max}}(t) \quad (3)$$

For more illustrations, we discuss the following example.

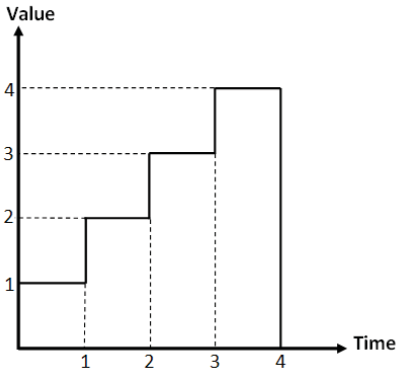

Figure 4. A sample IVF

Example: Suppose we have a job $J_{i,j}$ ($r_{i,j}$=0, $d_{i,j}$=4, and$e_{i,j}$=2), with an IVF as shown in Figure 4 Now, consider three suggestionsfor the execution of $J_{i,j}$:

1) *fr*is equal to fr_{max}, and$J_{i,j}$'s execution starts at the release time. Thus, it will be executed in the time interval [0, 2]. In this case, the accrued value is equal to 1×1+1×2=3.

2) $fr=fr_{max}$and $J_{i,j}$is executed such that it finishes at deadline. Therefore, it will be executed in the time interval [2, 4]. The accrued value of this case is calculated as 1×3+1×4=7. According to the shape of the IVF and due to the required execution time of the job, the most updatedsensor information and the most valuable respective computations are gained in this case. Therefore, the most accruable value for this job is 7.

3) $fr=fr_{max}/2$ and $J_{i,j}$is executed such that it finishes at deadline. In this case, to meet the deadline,$J_{i,j}$ should be started at time 0 and finished at time 4. Using (3), the IVF of this case is transformed as follows:

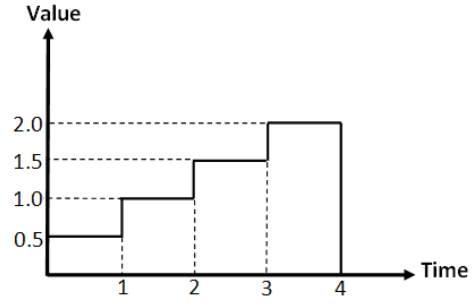

Figure 5. The effect of applying DVS to the IVF of Figure 4

$$V_{J_{i,j}}^{fr_{\max}/2}(t)=\frac{fr_{\max}/2}{fr_{\max}}.V_{J_{i,j}}^{fr_{\max}}(t)=\frac{V_{J_{i,j}}^{fr_{\max}}(t)}{2} \quad (4)$$

The resulting IVF is shown in Figure 5 The computation of the accrued value of this case is based on (4), and is done as 1×1/2+ 1×1+1×3/2+1×2=5. The justification for the relatively lower value in this case (with respect to the second case) is as follows: Since the job execution is lengthened to 4 time units, some sensor information may not be updated enough to get the best result, and thus, the accrued value of this case is lower.

5. Energy Bounded Value Accrual Scheduling Algorithm

In this section, we describe our approach for scheduling energy bounded systems in which time constraints are described by IVF. The intended scheduling algorithm provides a suboptimal solution and we call it Energy Bounded Sub Optimal Value Accrual (EBSOVA). Without loss of generality, we describe EBSOVA algorithm for non-decreasing IVFs. However, it can simply be applied to arbitrary shaped IVFs. As mentioned in Section 3.4, the goal for the intended system can be expressed as follows: 1) all the jobs should meet their deadlines, 2) while all deadlines are met, the system is to accrue the maximum possible value regarding the energy bound E_b. Since missing a job deadline causes a system failure and imposes a substantial non-compensable negative value to the system, thus we first discuss a method to guarantee that all deadlines are met.

Therefore, we have to specify the permissible time intervals for each job. To do this, the arrival times and the deadlines of all jobs are sorted in the ascending order. In this manner, 2*l* points are produced which divide *H* in at most to $2l-1$distinctsubintervals (*l*is the number of jobs in a hyper-period). We refer to eachsubinterval as $\Delta_s, s=1,2,\ldots$. A schematic view of subintervals is shown in Figure 6 The starting time of Δ_s is represented by Δ_s^{st}, and its end time is shown by Δ_s^e. The length of Δ_s is denoted as $|\Delta_s|$. Each Δ_s may be shared among some jobs, and the problem is to allocate the intervals to the jobs in order to satisfy (2).

Each Δ_s has a list of jobs, which is denoted as $J(\Delta_s)$, i.e., if $J_{i,j}\in J(\Delta_s)$, then $r_{i,j}\le\Delta_s^{st}\le\Delta_s^e\le d_{i,j}\cdot\Delta_s$ is allocated to jobs of $J(\Delta_s)$, thus, it may be divided into some chunks. We refer to the collection of these chunks as $JS(\Delta_s)$, i.e., we have $J_{i,j,k}\in JS(\Delta_s)$ for some $1\le k\le m_{i,j}$, if $J_{i,j}\in J(\Delta_s)$. Accordingly, we have $\sum_{J_{i,j,k}\in JS(\Delta_s)}|J_{i,j,k}^{fr}|\le|\Delta_s|$. As an example, consider Figure 6 here $J(\Delta_1)=\{J_{1,1},J_{1,2},\ldots,J_{1,n}\}$. In addition, $JS(\Delta_1)=\{J_{1,1,1}^{fr},J_{1,2,1}^{fr},\ldots,J_{1,n,1}^{fr}\}$ and $|J_{1,1,1}^{fr}|+|J_{1,2,1}^{fr}|+\cdots+|J_{1,n,1}^{fr}|\le|\Delta_1|$.

In the following, we first discuss the case which all IVFs have constant values within the range of the subintervals. Then we generalize the problem to an approximate solution for systems with non-constant IVFs in the subintervals. To divide Δ_s among the jobs of $J(\Delta_s)$, suppose that any $V_{J_{i,j}}^{fr}(t)$ ($J_{i,j}\in J(\Delta_s)$ and $\Delta_s^{st}\le t\le\Delta_s^e$) is equal to a constantvalue in the time interval Δ_s (Figure 7(a)).

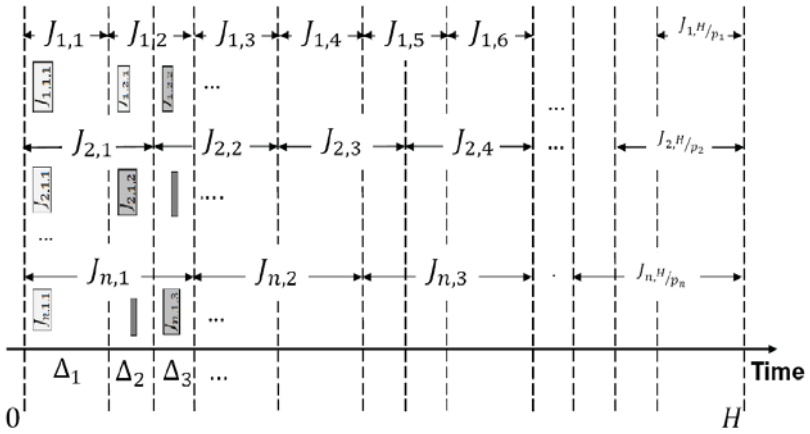

Figure 6. A schematic view of the relationship between subintervals and corresponding jobs

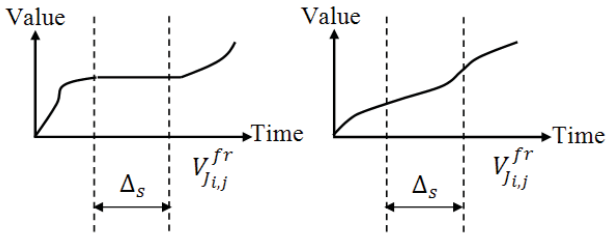

Figure 7. Two cases of IVFs in a time interval: (a) constant IVF (b) non-constant IVF

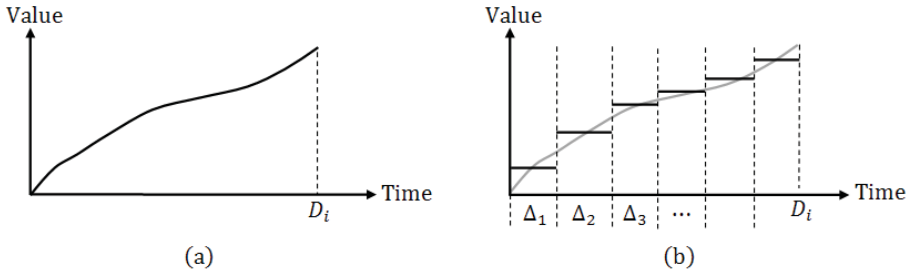

Figure 8. A sample of IVF estimation (a) main IVF (b) estimated IVF

This constant valueis called $V_{i,j,s}^{fr}$. When $V_{J_{i,j}}^{fr}(t)$ is constant in Δ_s, the operating frequency at which $J_{i,j,k}$ is executed does not affect the job segment accrued value (Lemma 1), and therefore, could be considered as a constant shown by $fr_{i,j,s}$. In this regards, the notation of $\left|J_{i,j,k}^{fr_{i,j,s}}\right|$ represents the length of $J_{i,j,k}$ when it is executed at $fr_{i,j,s}$.

When IVFs are constant in Δ_s, there is no difference that which pieces of Δ_s are allocated to which jobs in $J(\Delta_s)$, and the total accrued value of Δ_s can be calculated as follows:

$$\sum_{J_{i,j,k}\in JS(\Delta_s)} V_{i,j,s}^{fr_{i,j,s}} \cdot \left|J_{i,j,k}^{fr_{i,j,s}}\right| \tag{5}$$

In this case, the permutation of job segments has no effect on the amount of accrued value. On the other hand, if job IVFs in $J(\Delta_s)$ are non-constant, the permutation of segments would change thetotal accrued value (Figure 7(b)).

In this regards, a reasonable policy will split Δ_s into some smaller subintervals (shown again with Δ_s) until any $V_{J_{i,j}}^{fr}(t)$ ($J_{i,j}\in J(\Delta_s)$) can be estimated with a constant. After this step, the primary IVF will be converted into a multi-step function. A schematic view of this process is shown in Figure 8 The procedure of this splitting is discussed with moredetails in Section 5.1.

5.1. Estimation of IVF

This section describes the procedure of estimating an IVF by a multi-step function. As $V_{J_{i,j}}^{fr}(t)$ is unknown a prior, and we know only $V_{J_{i,j}}^{fr_{\max}}(t)$, we must find a way to use $V_{J_{i,j}}^{fr_{\max}}(t)$ instead of $V_{J_{i,j}}^{fr}(t)$ for the estimation. To do so, we first present the following lemma:

Lemma 1: If $V_{J_{i,j}}^{fr}(t)$ is constant in Δ_s, the operating frequency does not affect the accrued value of $J_{i,j,k}$.

Proof: Using (3), the relation between $V_{i,j,s}^{fr_{i,j,s}}$ and $V_{i,j,s}^{fr_{\max}}$ is calculated as $V_{i,j,s}^{fr_{i,j,s}} = \left(fr_{i,j,s} / fr_{\max} \right).V_{i,j,s}^{fr_{\max}}$, $J_{i,j,k} \in JS(\Delta_s)$. Therefore, the accrued value gained from executing $J_{i,j,k}$is derived as follows:

$$V_{i,j,s}^{fr_{i,j,s}} \cdot \left| J_{i,j,k}^{fr_{i,j,s}} \right| = \left(\frac{fr_{i,j,s}}{fr_{\max}} \right).V_{i,j,s}^{fr_{\max}} \cdot \left(\frac{fr_{\max}}{fr_{i,j,s}} \right) \cdot \left| J_{i,j,k}^{fr_{\max}} \right| = V_{i,j,s}^{fr_{\max}} \cdot \left| J_{i,j,k}^{fr_{\max}} \right|$$

Thus, if $V_{i,j,s}^{fr_{i,j,s}}$ is constant, the amount of accrued value by $J_{i,j,k}^{fr_{i,j,s}}$ is equalto $V_{i,j,s}^{fr_{\max}} \cdot \left| J_{i,j,k}^{fr_{\max}} \right|$, which is independent of the value of $fr(t)$ and the proof is completed. ■

As $V_{J_{i,j}}^{fr}(t)$ is estimated by a constant value in Δ_s ,the accrued value in Δ_s can be calculated using (5). After applying Lemma 1, (5) can be rewritten as:

$$\sum_{J_{i,j,k} \in JS(\Delta_s)} V_{i,j,s}^{fr_{\max}} \cdot \left| J_{i,j,k}^{fr_{\max}} \right| \tag{6}$$

There fore, $V_{J_{i,j}}^{fr_{\max}}(t)$ can be used in the estimation. Here, we define two parameters:

1) δ : This parameter determines the minimum acceptable length of Δ_s for a system, i.e., $\delta \le |\Delta_s|$

2) λ : This parameter determines the maximum acceptable mean slope of any $V_{J_{i,j}}^{fr_{\max}}(t)\left(J_{i,j} \in J(\Delta_s)\right)$ in the scope of Δ_s. As also indicated previously, the intended IVF is considered as non-decreasing, thus the maximum value is equal to $V_{J_{i,j}}^{fr_{\max}}\left(\Delta_s^e\right)$, and the minimum value is equal to $V_{J_{i,j}}^{fr_{\max}}\left(\Delta_s^{st}\right)$. Therefore this condition can be expressed as $\left(V_{J_{i,j}}^{fr_{\max}}\left(\Delta_s^e\right) - V_{J_{i,j}}^{fr_{\max}}\left(\Delta_s^{st}\right)\right) \Big/ \Delta_s \le \lambda$.

Accordingly, each Δ_s will repeatedly be broken into two equal segments until one of the above thresholds is exceeded. In other words, Δ_s will be split as long as its length is more than δ and the slope of the value function of at least one job in $J(\Delta_s)$is more than λ. Then, $V_{J_{i,j}}^{fr_{\max}}(t)$in $\Delta_s$will be estimated with a constant value. We refer to this estimated constant value as $v_{i,j,s}^{mean}$, which is calculated as follows:

$$v_{i,j,s}^{mean} = \frac{\int_{\Delta_s^{st}}^{\Delta_s^e} V_{J_{i,j}}^{fr_{\max}}(t)\,dt}{|\Delta_s|} \tag{7}$$

After applying this procedure, the total number of achieved subintervals will be increased with respect to the initial number of subintervals. The set of subintervals is shown by*S* and the number of subintervals in *S* is represented by |*S*|. Therefore, the value of |*S*| depends both on δ and λ.

5.2. Computing the Total Accrued Value

By estimating the value functions with multi-step functions, we will have an estimation of $EAV_{J_{i,j}}$,which is referred to as $EEAV_{J_{i,j}}$ (EEAV is defined as estimated EAV). According to (1), (6), and (7), the $EEAV_{J_{i,j}}$ can be calculated as follows:

$$EEAV_{J_{i,j}} = \sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} v_{i,j,s}^{mean} \cdot \left| J_{i,j,k}^{fr_{\max}} \right| \tag{8}$$

If the frequency is equal to $fr_{i,j,s}$ during the execution of $J_{i,j,k}$, $E_{i,j,k}$ can be calculated as follows:

$$E_{i,j,k} = P_d = \alpha . fr_{i,j,s}^3 . \left| J_{i,j,k}^{fr_{i,j,s}} \right|, \qquad J_{i,j,k} \in JS(\Delta_s) \tag{9}$$

Using (9), N_H can be computed in the following manner:

$$N_H = \frac{E_b}{E_H} = \frac{E_b}{\sum_{i=1}^{n} \sum_{j=1}^{H/p_i} \sum_{k=1}^{m_{i,j}} E_{i,j,k}} = \frac{E_b}{\sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} E_{i,j,k}} = \frac{E_b}{\sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} \alpha . fr_{i,j,s}^3 . \left| J_{i,j,k}^{fr_{i,j,s}} \right|} \tag{10}$$

Now, according to the recent definitions, some inequality constraints in (2)are replaced with some new ones. More precisely, the first inequality in (2) can be replaced as follows. By the definition of Δ_s, the allocated time to jobs in thesthtime interval must be less than Δ_s :

$$\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right| \le |\Delta_s| \tag{11}$$

By combining (2), (8), (10), and (11), the intended problem can be specified in the following form:

$$\max \frac{\sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} v_{i,j,s}^{mean} \cdot \left| J_{i,j,k}^{fr_{\max}} \right|}{\sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s}^3 \cdot \left| J_{i,j,k}^{fr_{i,j,s}} \right|}$$

subject to

$$\begin{aligned}
&\sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{\max}} \right| = e_{i,j} && i = 1,\ldots,n \\
& && j = 1,\ldots,H/p_i \\
&\sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{i,j,s}} \right| \le |\Delta_s| && s = 1,\ldots,|S| \\
&fr_{\min} \le fr_{i,j,s} \le fr_{\max} && i = 1,\ldots,n \\
& && j = 1,\ldots,H/p_i \\
& && s = 1,\ldots,|S|
\end{aligned} \tag{12}$$

As $\left| J_{i,j,k}^{fr_{i,j,s}} \right|$ and $\left| J_{i,j,k}^{fr_{\max}} \right|$ are dependent parameters ($\left| J_{i,j,k}^{fr_{i,j,s}} \right| = \left(fr_{i,j,s} / fr_{\max} \right) . \left| J_{i,j,k}^{fr_{\max}} \right|$), one of them can be omitted from (12). By substitution of $\left| J_{i,j,k}^{fr_{i,j,s}} \right|$ with $\left(fr_{i,j,s} / fr_{\max} \right) . \left| J_{i,j,k}^{fr_{\max}} \right|$, (12) can be written as:

$$\max \frac{\sum_{s=1}^{|S|}\sum_{J_{i,j,k}\in JS(\Delta_s)} v_{i,j,s}^{mean}.\left|J_{i,j,k}^{fr_{\max}}\right|}{\sum_{s=1}^{|S|}\sum_{J_{i,j,k}\in JS(\Delta_s)} fr_{i,j,s}^{3}.\left|J_{i,j,k}^{fr_{i,j,s}}\right|}$$

subject to

$$\begin{aligned}
&\sum_{k=1}^{m_{i,j}}\left|J_{i,j,k}^{fr_{\max}}\right| = e_{i,j} && i=1,\ldots,n\\
& && j=1,\ldots,H/p_i\\
&\sum_{k=1}^{m_{i,j}}\frac{fr_{i,j,s}}{fr_{\max}}.\left|J_{i,j,k}^{fr_{\max}}\right| \le \left|\Delta_s\right| && s=1,\ldots,|S|\\
&fr_{\min} \le fr_{i,j,s} \le fr_{\max} && i=1,\ldots,n\\
& && j=1,\ldots,H/p_i\\
& && s=1,\ldots,|S|
\end{aligned} \tag{13}$$

Now, for computing EEAV, (13) is to be solved. The complete solution is given in Appendix B. The given solution determines all job segments and the corresponding frequencies in a hyper-period. Then, EEAV can be calculated as follows:

$$EEAV_{J_{i,j}} = \left(\sum_{s=1}^{|S|}\sum_{J_{i,j,k}\in JS(\Delta_s)} v_{i,j,s}^{mean}.\left|J_{i,j,k}^{fr_{\max}}\right|\right).\frac{E_b}{\alpha\sum_{s=1}^{|S|}\sum_{J_{i,j,k}\in JS(\Delta_s)} fr_{\max}.fr_{i,j,s}^{2}.\left|J_{i,j,k}^{fr_{\max}}\right|} \tag{14}$$

5.3. Determination of the Operating Frequency in Δ_s

As it is shown in Appendix A, when $fr_{i,j,s} = fr_s$ with the formulation of the

$$fr_s = \min\left(fr_{\min}, \frac{\sum_{J_{i,j,k}\in JS(\Delta_s)} fr_{\max}.\left|J_{i,j,k}^{fr_{\max}}\right|}{\left|\Delta_s\right|}\right) \tag{15}$$

The consumed energy in Δ_s is minimized. Using fr_s, EEAV is calculated as follows:

$$EEAV_{J_{i,j}} = \left(\sum_{s=1}^{|S|}\sum_{J_{i,j,k}\in JS(\Delta_s)} v_{i,j,s}^{mean}.\left|J_{i,j,k}^{fr_{\max}}\right|\right).\frac{E_b}{\alpha\sum_{s=1}^{|S|}\sum_{J_{i,j,k}\in JS(\Delta_s)} fr_{\max}.fr_{s}^{2}.\left|J_{i,j,k}^{fr_{\max}}\right|} \tag{16}$$

This result has another important outcome. If each job segment has a different operating frequency, a great amount of timing and energy is imposed due to frequency switching overhead. However, as it is proved in Appendix A, the frequency in the entire Δ_s is the same for all job segments; thus, the frequency switching overhead becomes negligible through the system if δ and λ are chosen properly. We will use this result in Section 6.

6. Experimental Results

In this section, we evaluate EBSOVA with several different task sets and IVFs. EBSOVA is compared with SOVA [7] and Earliest Deadline First (EDF) [22]. EDF does not consider the value functions in its decision makings and the only criterion for its scheduling is meeting deadlines. Moreover, a hypothetical optimal condition is considered, in which the maximum possible value that could be achieved by a job - without the presence of other jobs - is computed. Such computed values for all jobs are then summed together and considered as the optimal value of the system. This optimal value simply referred to as Optimal Accrual Value (OAV). Although OAV is an unrealistic measure, it provides a good reference for evaluating the effectiveness of EBSOVA.

Three IVFs are examined: non-increasing, non-decreasing and sine-shaped. The non-increasing IVF is computed as $-V_i^{\max}/p_i$, and non-decreasing IVF is set to $V_i^{\max}/p_i$. Sine-shaped IVF is expressed as $V_i^{\max}\sin(\pi/p_i)$. In these experiments $V_i^{\max}$ is set to p_i.

The phases of all tasks are set to 0 in all experiments. In our configuration, we vary the system utilization in the range of [0.1, 0.9]. We perform several experiments with different task sets for evaluating EBSOVA. Each task set is composed of 5 periodic tasks. The period of each task is uniformly distributed in the interval [1, 40]. The execution times are randomly chosen with uniform distribution in a way that the generated task set has the desired utilization.

In these experiments, we set fr_{max} and fr_{min} using the specification of the AMD Athlon processors [23]. The required LPs in Appendix B aresolved using MATLAB 7.4 [24] running on a Pentium 4 processor with the speed of 3 GHz and 1 GB of RAM.

δ must be considered very small, so that the difference between the total accrued EAV and the obtained EEAV becomes negligible. While it is impractical to measure accrued EAV of the system (because this leads to converge δ to zero), by selecting a small δ, δ/H becomes negligible, which provides good estimation of the total accrued EAV.

Beside the accuracy of computations, other factors can also affect the determination of δ and λ. For example, the overhead of context switching and frequency switching may motivate us to select larger parameters. As it is mentioned in Section 5.3, frequency switching is occurred on the Δ_s s' boundaries. Thus, the amount of overhead can be controlled by tuning the values of δ and λ.

6.1. Effectiveness of EBSOVA

In this section, we study the effectiveness of our energy saving method. The comparison of EBSOVA with SOVA and EDF for non-increasing, non-decreasing and sine-shaped IVFs are shown in Figures 9, 10, and 11, respectively. The results are normalized by OAV. In these set of experiments, fr_{max} is set to 1000*MHz*, and fr_{min} is equal to 360*MHZ*.

As it is shown in Figures 9, 10, and 11, for task sets of any utilization, EBSOVA results in gaining more accrued EEAV. From these figures, weobserve that EBSOVA performs well when load is low. This is due to the fact that in the low utilization, Δ_s is occupied partially, and thus, f_{rs} is low. Therefore, in this case the consumed Energy in Δ_s is low. As the load increases, f_{rs} is also increases. As a consequence, the difference between the performances of the methods decreases.

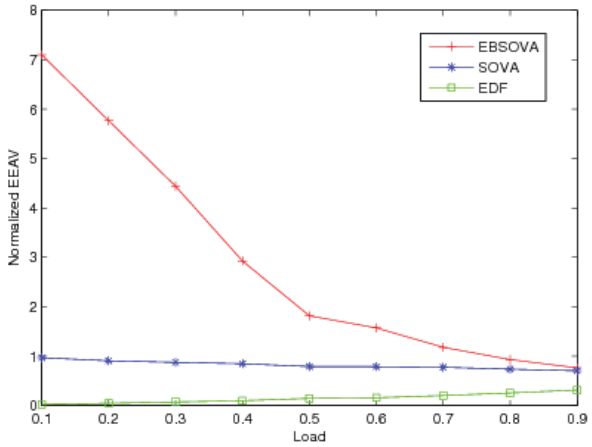

Figure 9. Non-decreasing IVF

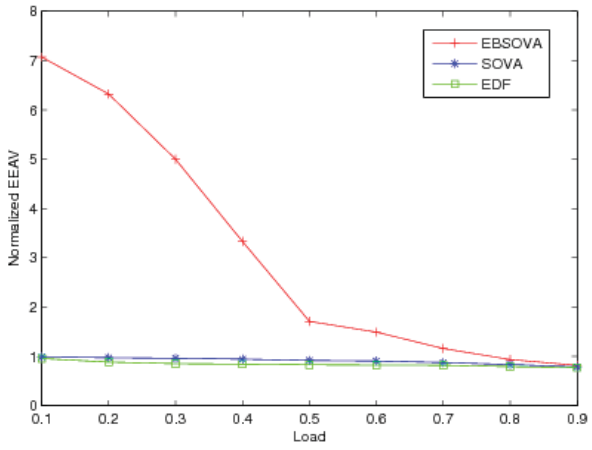

Figure 10. Non-increasing IVF

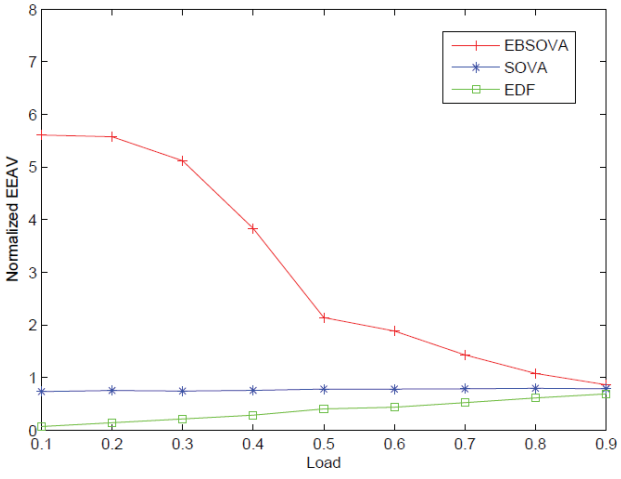

Figure 11. Sine-shaped IVF

Figure 12 shows some results to compare the accrued EEAV by EBSOVA using different IVFs. The attained results are normalized with respect to SOVA. As shown in this figure, the behaviors of all IVFs with respect to different loads are similarto each other which is an indication of the effectiveness of EBSOVA.

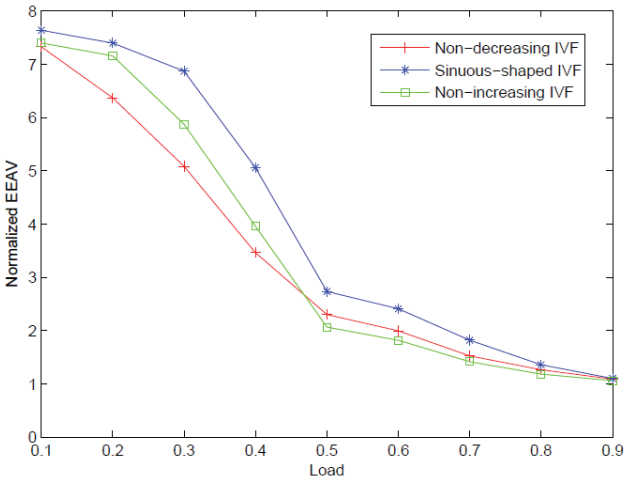

Figure 12. Comparison between different IVFs

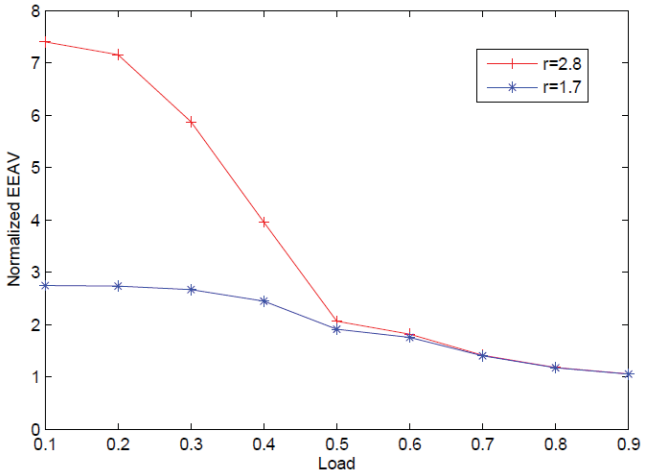

Figure 13. Comparison between different processorspeeds

6.2. The Effect of Max-Ratio on the Performance

An important factor in the determination of EBSOVA's effectiveness comparing to SOVA is the amount of *max-ratio.* Here, we examine two cases:

1) fr_{max}=1000*MHZ* and fr_{min}=360*MHZ*, and hence, *max-ratio* $\approx$ 2.8.

2) fr_{max}=500*MHZ* and fr_{min}=300*MHZ*, and hence, *max-ratio* $\approx$ 1.7.

Figure 13 shows the gathered EEAV by EBSOVA in both cases for non-decreasing IVF. The attained results are normalized with respect to SOVA. As shown in the figure, the difference between the curves is much higher when the load is low. As the load increases, the gap between them decreases. This is due to the fact that, typically, when the load is low, $\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{\max} . \left| J_{i,j,k}^{fr_{\max}} \right| / |\Delta_s|$ is less than fr_{min}. Thus, fr_{min} is chosen as the working frequency, and the ratio of the EBSOVA's EEAV to the SOVA's EEAV is proportional to $fr_{\max}^2 / fr_{\min}^2$. However, when the load is high,fr_s is set to $\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{\max} . \left| J_{i,j,k}^{fr_{\max}} \right| / |\Delta_s|$,and *max-ratio* losses its effect on the determination of the amount of obtained EEAV.

7. Related Work

The most related works to this study that consider more general timing constraints with respect to classic deadlines, are those that take the TUF and gravitational time constraints for the jobs into account. TUF was first introduced by Jensen [2]. In [25], Strayer presents a similar framework called "importance function". However, no new scheduling algorithm is introduced in this reference. In general, both TUF and "importance function" can take arbitrary shapes and have similar meanings.

The first utility-accrual (UA) scheduling algorithm was proposed by Locke [3]. This algorithm considers arbitrary-shaped TUFs for independent task model. Locke's work was then extended in several studies, including [16-17,26-28]. The main criteria that most of the UA scheduling algorithms [16-17,26,29] use in their decision makings is Potential Utility Density (PUD). The PUD of a job simply measures the amount of utility that can be accrued per unit of time by executing the job and the jobs that they depend upon. Most efforts on UA scheduling algorithms [16-17, 29-31] consider some specific shapes of TUFs (e.g., step or non-increasing TUFs) rather than the general shapes of TUFs studied in [3].

In [4], Clark has proposed a UA scheduling algorithm, called DASA, which allows jobs mutuallyexclusively share non-CPU resources under the single-unit resource request model. DASA also provides assurance on timeliness behavior during under-load situations. However, DASA is restricted to step TUFs. In [32], a probabilistic method that guarantees a lower bound on tasks' accrued utilities is provided.

In addition to [3], [26] also considers arbitrary-shaped TUFs. The paper assumes the TUF as a "black box" that simply gets a job completion time and returns the respective definite utility. The problem of scheduling tasks with arbitrarily shaped TUFs and mutual exclusion resource constraints is also addressed in the latter study.

Another time constraint model that has a closer concept to IVF is the gravitational model [5]. This model describes the behavior of applications that have instant sensitive time constraints. Each job in the model has a specific instant, namely target point, at which execution results in the highest utility. Further, according to the job importance, the execution of each job can shift around this instant which resultin lower utility accrual based on the respective deviation from the target point.

The gravitational model is based on a pendulum. A pendulum is an object that is attached to a pivot point and can swing freely. The equilibrium state of a single object in a pendulum is the central point. An object placed in this position will not swing. If there is more than one object, they will push each other aside and their equilibrium state will depend on their relations between weight and size. The equilibrium state of the objects is mapped to the state of gaining maximum utility in the corresponding real-time system.

The gravitational model mentioned above can be considered as a special case of IVF-constrained systems. The solution provided in [6] for scheduling task sets described with the gravitational model is mapped to the problem of scheduling the IVF constrained task sets with half-elliptical IVFs. As also mentioned in Section 2, in spite of the gravitational model that has only one importantinstant as the target execution time, IVF specifies the value of all instants of job execution. In addition, due to the physical interpretation of the pendulum as the basis for the gravitational model, it is not proper for modeling preemptive task sets.

Other recent studies that consider close concepts to IVF are [8] and [9] which concentrate on the instants of performing the task input and output (I/O) operations. In the studies, the authors consider the effects of delays between the input and output operations in real-time systems on the accuracy of results and present some heuristic methods to improve the metric. However, in the current study, we focus on the instants of job execution rather than I/O operations, and propose an optimal solution for the problem.

On the other hand, voltage scheduling for variable-voltage processors has recently been extensively studied for various system models. These algorithms are classified into offline and online ones. Offline algorithms compute voltage schedules with the assumption that timing parameters of each job is constant and known a priori, while online algorithms dynamically adjust the processor's frequency along with the supply voltage based on the execution history.

For static job sets, where each job has its own release time, deadline, and worst case execution time known offline, Swaminathan et al. [10] proposed an offline voltage scheduling algorithm using generalized network flow. In [33], an optimal offline voltage scheduling algorithm assuming EDF scheduling policy for static job sets has been proposed. The offline scheduling problem for the static job model with arbitrary priority assignment (including Rate-Monotonic and Deadline-Monotonic) was proved as an NP-hard problem. Then a fully polynomial time approximation scheme for the problem was presented [34]. The problem of integrating DVS mechanism into the priority driven schedulers has been explored in some other studies, too. Pillai and Shin [35] derived the minimal frequency that can make a task set schedulable under EDF, and proposed a near-optimal method under RM. [36] also provides an algorithm to find the optimal frequency for fixed-priority assignments, assuming that the frequency of the processor can be varied continuously in a given range.

Several online voltage scheduling algorithms have also been developed in the literature. Since the tasks' demands are not always the worst-case, some reclamation techniques have been proposed to reclaim the unused cycles to save more energy for both dynamic-priority periodic tasks [11,37] and fixed-priority periodic tasks [12,14]. These reclamation techniques first run the CPU fast (assuming the worst-case demand) and then decelerate when a job finishes earlier. In addition, more aggressive techniques that proposed in [37] assume that the current and future instances of a task will most probably present a computational demand which is lower than the worst-case. The aggressive techniques start a job with the lowest frequency required to meet the minimum work that must be done now, and then accelerate as it progresses.

Stochastic DVS is also used to handle runtime demand variations [13-15, 38]. [38] aims at general-purpose applications. In the study, a DVS technique called PACE is presented with applying soft deadlines and estimating task work distributions. In [13-14], some real-time DVS techniques are proposed. In [13], offline analysis is combined

with online slack-time stealing and cycle-based stochastic DVS for hard real-timesystems. [15] introduces a stochastic dynamic power management policy for soft real-timesystems. The idea is based on a Markovian model of the system, which presents an analytical technique for tuning the system performance parameters.

The interplay of TUF and energy consumption has been addressed in some studies. The work in [18] considers system-level energy consumption, but it is restricted to resource-independent activities, and provides no assurances on timeliness behaviors. In [39], mutual exclusion resource constraints are also considered. [40] has also solved the problem of UA scheduling with the consideration of the system level power consumption which has been extended in [17] for the consideration of mutual exclusion resource constraints. [16]is an extension of [17], and considered energy constraints in TUF-based systems. In the work, the system is assumed to be battery-powered and must remain functional during a specified time with a bounded energy budget. A stochastic view to a TUF-constrained firm real-time system has also been studied in [19] which proposes some analytical methods for the system performance evaluation. Also, the method is used to find the optimal processor speed in a DVS-enabled single-processor system to obtain the best utility for the jobs.

The problem of real-time scheduling under energy constraints has also been considered in some different researches. [41] considers the IRIS (Increasing Reward with Increasing Service) execution model. In the reward-based scheduling framework, each job is divided into a mandatory part and an optional part. The mandatory part of a task should be completed by its deadline while the optional part can be selectively executed before the deadline. The optional part is assumed to follow the mandatory part in sequence and can be interrupted at any time. The goal of reward-based scheduling is to find optional parts that maximize the total reward while meeting all the deadlines. This goal is achieved in [41] regarding energy constraints. A similar problem is also studied in [42].

However, as also mentioned in Section 1, IVF is a more general model of timing constraints with respect to the previous works which can describe the behavior of specific real-time systems in a more precise manner. Further, to the best of the authors' knowledge, there exists no previous study on the semantics of applying DVS to IVF-basedreal-time systems.

8. Concluding Remarks

Many emerging battery-powered embedded real-time systems are subject to energy bounds, embodied by a battery with a finite lifetime. In this paper, beside a new model of timing behaviors for specific real-time systems, namely IVF, we have discussed on the semantics of applying DVS to IVF-constrained jobs. As shown in this paper, DVS can change the shape of IVF for such tasks, and therefore,can affect the way of scheduling tasks in such a system. Accordingly, an IVF-based real-time scheduling algorithm, EBSOVA, is proposed which considers activities subject to IVF time constraints, periodic job arrivals, and a bounded source of energy. EBSOVA tries to maximize the accrued value of the system under the bounded budget of energy. We have shown the optimality of the algorithm analytically. The simulation experiments also confirm the superiority of EBSOVA over some past algorithms.

Several aspects of the work are directions for further research. Examples include the considerations of more general task arrival models and discrete processor speed levels.

References

[1] K. G. Shin, and P. Ramanathan, Real-Time Computing: A New Discipline of Computer Science and Engineering, in Proc. of the IEEE, 82(1) (1994) 6-24.

[2] E. D. Jensen, C. D. Locke, and H. Tokuda, A Time-Driven Scheduling Model for Real-Time Systems, in Proc. of IEEE Real-Time Systems Symposium (RTSS), 1985, pp. 112-122.

[3] C. D. Locke, Best-Effort Decision Making for Real-Time Scheduling, PhD dissertation, CMU-CS-86-134, Carnegie Mellon University, 1986.

[4] R. K. Clark, Scheduling Dependent Real-Time Activities, PhD dissertation, CMU-CS-90-155, Carnegie Mellon Univ., 1990.

[5] R. Guerra, and G. Fohler, A Gravitational Task Model for Target Sensitive Real-Time Applications, in Proc. of IEEE Euro Micro Conf. on Real-Time Systems (ECRTS), 2008, pp. 309-317.

[6] R. Guerra, and G. Fohler, A Gravitational Task Model with Arbitrary Anchor Points for Target Sensitive Real-Time Applications, J. of Real-Time Systems, 43(1) (2009)93–115.

[7] L. Farzinvash, and M. Kargahi, A Scheduling Algorithm for Execution-Instant Sensitive Real-Time Systems, in Proc. of IEEE Int. l Conf. Embedded and Real-Time Computing Systems and Application (RTCSA), 2009, pp. 511-518.

[8] M. Nasri, M. Kargahi, and M. Mohaqeqi, Scheduling of Accuracy-Constrained Real-Time Systems in Dynamic Environments, IEEE Embedded Systems Letters (ESL), 4 (3) (2012) 61-64.

[9] M. Nasri, and M. Kargahi, A Method for Improving Delay-Sensitive Accuracy in Real-Time Embedded Systems, In Proc. of Int'l Conf. on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2012, pp. 378-387.

[10] V. Swaminathan, and K. Chakrabarty, Network Flow Techniques for Dynamic Voltage Scaling in Hard Real-Time Systems, IEEE Trans. on CAD of Integrated Circuits and Systems, 23(10) (2004)1385-1398.

[11] W. Kim, J. Kim, and S. L. Min, A Dynamic Voltage Scaling Algorithm for Dynamic-Priority Hard Real-Time Systems Using Slack Time Analysis, in Proc. of Design Automation Test Conf. Europe (DATE), 2002, pp. 788-794.

[12] W. Kim, J. Kim, and S. L. Min, Dynamic Voltage Scaling Algorithm for Fixed-Priority Real-Time Systems Using Work-Demand Analysis, In Proc. of Int'l Symp. on Low Power Electronics and Design (ISLPED), 2003, pp. 396-401.

[13] W. Yuan, and K. Nahrstedt, Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems, ACM Symp. on Operating System Principles (SOSP), 2003, pp. 149-163.

[14] F. Gruian, Hard Real-time Scheduling for Low Energy Using Stochastic Data and DVS Processors, In of Proc. Int'l Symp. on Low Power Electronics and Design (ISLPED), 2001, pp. 46-51.

[15] M. Kargahi, and A. Movaghar, Stochastic DVS-based Dynamic Power Management for Soft Real-Time Systems, Microprocessors and Microsystems - Embedded Hardware Design, 32(3) (2008) 121-144.

[16] H. Wu, B. Ravindran, and E. D. Jensen, Utility Accrual Real-Time Scheduling Under the Unimodal Arbitrary Arrival Model with Energy Bounds, IEEE Trans. Computers, 56 (10) (2007) 1358-1371.

[17] H. Wu, B. Ravindran, E. D. Jensen, and P. Li, Energy-Efficient Utility Accrual Scheduling under Resource Constraints for Mobile Embedded Systems, ACM Trans. Embedded Computer Systems, 5(3) (2006) 513-542.

[18] J. Wang, B. Ravindran, and T. Martin, Power Aware Best-Effort Real-Time Task Scheduling Algorithm, in Proc. of IEEE Int'l Symp. on Object-Oriented Real-Time Distributed Computing (ISORC), 2003, pp. 21-28.

[19] M. Kargahi, and A. Movaghar, Performance Optimization Based on Analytical Modeling in a Real-Time System with Constrained Time/Utility Functions, IEEE Transactions on Computers(TC), 60 (8) (2011) 1169-1181.

[20] D. P. Maynard, S. E. Shipman, R. K. Clark, J. D. Northcutt, R. B. Kegley, B. A. Zimmerman, and P. J. Keleher, An Example Real-Time Command, Control, and Battle Management Application for Alpha, Archons Project TR-88121, Department of Computer Science, Carnegie Mellon University, 1988, http://www.real-time.org.

[21] D. M. D. Zhu, N. AbouGhazaleh, and R. Melhem, Power Aware Scheduling for And/Or Graphs in Multiprocessor Real-Time Systems, In Proc. of Int'l Conf. on Parallel Processing (ICPP), 2002, pp. 593-601.

[22] C. L. Liu, and J. W. Layland, Scheduling Algorithms for Multiprogramming in Hard Real-Time Environment, J. of ACM, 20(1) (1973) 46-61.

[23] "Mobile AMD-K6-2+ Processor Data Sheet," Advanced Micro Devices Corp., Publication \#23446, 2000.

[24] www.mathworks.com.

[25] W. T. Strayer, Function-Driven Scheduling: A General Framework for Expressing and Analysis of Scheduling, PhD dissertation, Dept. of Computer Science, University of Virginia, 1992.

[26] P. Li, H. Wu, B. Ravindran, and E. D. Jensen, A Utility Accrual Scheduling Algorithm for Real-Time Activities with Mutual Exclusion Resource Constraints, IEEE Trans. Computers , 55 (4) (2006) 454-469.

[27] D. Mosse, M. E. Pollack, and Y. Ronen, Value-Density Algorithm to Handle Transient Overloads in Scheduling, in Proc. of IEEE Euro Micro Conf. on Real-Time Systems (ECRTS), 1999, pp. 278-286.

[28] S. A. Aldarmi, and A. Burns, Dynamic Value-Density for Scheduling Real-Time Systems, in Proc. of IEEE Euro Micro Conf. on Real-Time Systems (ECRTS), 1999, pp. 270-277.

[29] Feizabadi, B. Ravindran, and E. D. Jensen, MSA: A Memory-Aware Utility Accrual Scheduling Algorithm, ACM Symp. on Applied Computing (SAC), 2005, pp. 857-862.

[30] J. Wang, and B. Ravindran, Time-Utility Function-Driven Switched Ethernet: Packet Scheduling Algorithm, Implementation, and Feasibility Analysis, IEEE Trans. on Parallel and Distributed Systems, 15(2) (2004) 119-133.

[31] K. Chen, and P. Muhlethaler, A Scheduling Algorithm for Tasks Described by Time Value Function, J. of Real-Time Systems, 10(3) (1996) 293-312.

[32] P. Li, H. Cho, B. Ravindran, and E. D. Jensen, Stochastic, Utility Accrual Real-Time Scheduling with Task-Level and System-Level Timeliness Assurances, in Proc. of IEEE Int'l Symp. on Object-oriented Real-Time Distributed Computing (ISORC), 2005, pp. 216-223.

[33] F. Yao, A. Demers, and S. Shenker, A Scheduling Model for Reduced CPU Energy, IEEE Symp. on Foundations of Computer Science (FOCS), 1995, pp. 374–382.

[34] H.-S. Yun, and J. Kim, On Energy-Optimal Voltage Scheduling for Fixed-Priority Hard Real-Time Systems, ACM Trans. on Embedded Computing Systems, 2(3) (2003)393-430.

[35] P. Pillai, and K. G. Shin, Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems, ACM Symp. on Operating System Principles (SOSP), 2001, pp. 89-102.

[36] S. Saewong, and R. Rajkumar, Practical Voltage-Scaling for Fixed-Priority RT-Systems, In Proc. of the IEEE Real-Time Technology and Applications Symp. (RTAS), 2003, pp. 106-114.

[37] H. Aydin, R. Melhem, D. Mosse, and P. M. Alvarez, Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems, in Proc. of IEEE Real-Time Systems Symp. (RTSS), 2001, pp. 95-105.

[38] J. Lorch, and A. Smith, Dynamic Voltage Scaling Algorithms with PACE, in Proc. of ACM SIGMETRICS, 2001, pp. 50-61.

[39] H. Wu, B. Ravindran, E. D. Jensen, and P. Li, Energy-Efficient Utility Accrual Scheduling under Resource Constraints for Mobile Embedded Systems, in Proc. of ACM Conf. on Embedded Systems Software (EMSOFT), 2004, pp. 64-73.

[40] H. Wu, B. Ravindran, and E. D. Jensen, Energy-Efficient, Utility Accrual Real-Time Scheduling Under the Unimodal Arbitrary Arrival Model, in Proc. of Design Automation Test Conf. Europe (DATE), 2005, pp. 474-479.

[41] H. Yun, and J. Kim, Reward-Based Voltage Scheduling for Hard Real-Time Systems with Energy Constraints, Design Automation for Embedded Systems, 11(1) (2007) 25-48.

[42] C. Rusu, R. G. Melhem, and D. Mosse, Maximizing Rewards for Real-Time Applications with Energy Constraints, ACM Trans. Embedded Computer Syst. 2(4) (2003) 537-559.

Appendix A. Determining Minimum Consumed Energy in Δ_s

In this appendix, we prove that when $fr_{i,j,s}$ is set to fr_s (fr_s is defined in (15)), the consumed energy in Δ_s is minimum. As the first step, we prove the problem for the condition that there is no slack remaining in Δ_s, i.e, $\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right| = |\Delta_s|$

We define the frequency of fr_{avg} as:

$$fr_{avg} = \frac{\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right|}{\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right|} \quad \text{(A.1)}$$

First, the valid range of fr_{avg} must be determined.

Lemma 2: fr_{avg} is in the range of $[fr_{min}, \ldots, fr_{max}]$

Proof: Since fr_{avg} is the weighted average of $fr_{i,j,s}$'s of Δ_s, the proof in straightforward. ■

In the next step, we show that for any given set of $\left| J_{i,j,k}^{fr_{i,j,s}} \right|$ and $fr_{i,j,s}$, the consumed energy when using fr_{avg}, is minimum. Here we show the consumed energy of operating at fr_{avg} by $E_{fr_{avg}}$, and the consumed energy of operating at $fr_{i,j,s}$ by $E_{fr_{i,j,s}}$. We must show that:

$$E_{fr_{avg}} \le E_{fr_{i,j,s}} \quad \text{(A.2)}$$

Lemma 3: For any given set of $\left| J_{i,j,k}^{fr_{i,j,s}} \right|$ and $fr_{i,j,s}$, $E_{fr_{avg}} \le E_{fr_{i,j,s}}$.

Proof: First we show that the following equation is a tautology. This can simply be done by deduction.

$$\left(\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right| \right) . \left(\frac{\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right|}{\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right|} \right)^2 \le \sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s}^2 . \left| J_{i,j,k}^{fr_{i,j,s}} \right| \quad \text{(A.3)}$$

Now, to prove (A.2), (A.3) is multiplied by $\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right| \Big/ \sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right|$. So we have:

$$\frac{\left(\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right| \right)^3}{\left(\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right| \right)^2} \le \frac{\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right|}{\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right|} . \left(\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s}^2 . \left| J_{i,j,k}^{fr_{i,j,s}} \right| \right) \quad \text{(A.4)}$$

Using some simple algebraic manipulations, we find that:

$$\frac{\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right|}{\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right|} . \left(\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s}^2 . \left| J_{i,j,k}^{fr_{i,j,s}} \right| \right) \le \sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s}^3 . \left| J_{i,j,k}^{fr_{i,j,s}} \right| \quad \text{(A.5)}$$

By combining (A.4) and (A.5), (A.2) is derived. Thus the proof is complete. ■

This result implies that for any given set of $\left| J_{i,j,k}^{fr_{i,j,s}} \right|$ and $fr_{i,j,s}$, $E_{fr_{avg}}$ is the minimum consumed energy in Δ_s. Now, consider the case when some slack remains in Δ_s, i.e., $\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right| \le |\Delta_s|$. We define the frequency of fr_s as:

$$fr_s = \frac{\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right|}{|\Delta_s|} \quad \text{(A.6)}$$

We show the consumed energy of operating at fr_s by E_{fr_s}. $fr_s \le fr_{avg}$, because $\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right| \le |\Delta_s|$. Therefore, we have $E_{fr_s} \le E_{fr_{avg}}$. In fact, E_{fr_s} is the minimum achievable amount of consumed energy. The reason is that the denominator of (A.6) shows the amount of work that must be executed in Δ_s, and the numerator of it is the maximum range that the execution of jobs of Δ_s can be delayed without violating timing constraints. In the next step, the valid range of fr_s must be determined. Using $fr_s \le fr_{avg}$ and Lemma 2, we have:

$$fr_s \le fr_{\max} \quad \text{(A.7)}$$

There is no relationship between fr_s and fr_{min}. It may be possible that $\sum_{J_{i,j,k} \in JS(\Delta_s)} \left| J_{i,j,k}^{fr_{i,j,s}} \right|$ be much less than Δ_s, and as a consequence, fr_s becomes less than fr_{min}. Thus, the definition of fr_s must be modified as:

$$fr_s = \max\left(fr_{\min}, \frac{\sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{i,j,s} . \left| J_{i,j,k}^{fr_{i,j,s}} \right|}{|\Delta_s|} \right) \quad \text{(A.8)}$$

By substitution of $\left| J_{i,j,k}^{fr_{i,j,s}} \right|$ by $\left(fr_{\max} / fr_{i,j,s} \right) . \left| J_{i,j,k}^{fr_{\max}} \right|$ in (A.8), (15) is derived.

Appendix B. Solving Maximization Problem

Here we present a solution for (13). Our strategy is to find an initial solution, and improve the solution using local search techniques. To find an initial solution, the limitations of systems' energy are temporary ignored. Therefore, $fr_{i,j,s}$ is set to fr_{max}. By substitution of $fr_{i,j,s}$ with fr_{max}in (13), the following equation is given:

$$\max \frac{\sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} v_{i,j,s}^{mean} . \left| J_{i,j,k}^{fr_{\max}} \right|}{\sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{\max}^3 . \left| J_{i,j,k}^{fr_{\max}} \right|}$$

subject to

$$\sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{\max}} \right| = e_{i,j} \qquad i = 1, \ldots, n$$

$$j = 1, \ldots, H / p_i$$

$$\sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{\max}} \right| \le |\Delta_s| \qquad s = 1, \ldots, |S| \quad \text{(B.1)}$$

The numerator of (B.1) is constant as shown below:

$$\sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} fr_{\max}^3 . \left| J_{i,j,k}^{fr_{\max}} \right| = \sum_{i=1}^{n} \sum_{j=1}^{H/p_i} \sum_{k=1}^{m_{i,j}} fr_{\max}^3 . \left| J_{i,j,k}^{fr_{\max}} \right| =$$

$$fr_{\max}^3 . \sum_{i=1}^{n} \sum_{j=1}^{H/p_i} \sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{\max}} \right| = fr_{\max}^3 . \sum_{i=1}^{n} \sum_{j=1}^{H/p_i} e_{i,j} = Const.$$

Thus, (B.1) can be rewritten as:

$$\max \sum_{s=1}^{|S|} \sum_{J_{i,j,k} \in JS(\Delta_s)} v_{i,j,s}^{mean} . \left| J_{i,j,k}^{fr_{\max}} \right|$$

subject to

$$\sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{\max}} \right| = e_{i,j} \qquad i = 1, \ldots, n$$

$$j = 1, \ldots, H / p_i$$

$$\sum_{k=1}^{m_{i,j}} \left| J_{i,j,k}^{fr_{\max}} \right| \le |\Delta_s| \qquad s = 1, \ldots, |S| \quad \text{(B.2)}$$

Now, to solve (13), (B.2) is to be solved at first stage. This gives the initial solution of $\left| J_{i,j,k}^{fr_{\max}} \right|$ and $fr_{i,j,s} = fr_{\max}$. The solution observes the constraints of (13), and can be used as an initial point. After this step, using local search techniques, $J_{i,j}$ is redistributed between corresponding Δ_s s iteratively. At each iteration, fr_s and EEAV is calculated to find the preferred result. After some iteration, we obtain EEAV.

Leili Farzinvash is currently a Ph.D. student in the Department of Computer Engineering and Information Technology in Amir-Kabir University of Technology (Tehran Poly-Techniques). She received her B.S. and M.S. degrees in Computer Engineering from the Department of Electrical and Computer Engineering at University of Tehran, Iran in 2007 and 2009, respectively.
E-mail: l.farzinwash@ece.ut.ac.ir

Mehdi Kargahi is currently an assistant professor in the Department of Electrical and Computer Engineering at the University of Tehran, Iran. He received his B.S. degree in Computer Engineering from Amir-Kabir University of Technology (Tehran Poly-Techniques) in Tehran (1998) and both M.S. (2001) and Ph.D. (2006) in Computer Engineering from Sharif University of Technology in Tehran. He has also been a researcher in the School of Computer Science at Institute for studies in theoretical Physics and Mathematics (IPM) from 2003. His research interests include performance modeling and power management of dependable and distributed real-time systems with stochastic properties.
E-mail: kargahi@ut.ac.ir, kargahi@ipm.ir

Paper Handling Data:

Submitted: 14.11.2010
Received in revised form: 03.11.2012
Accepted: 06.12.2012
Corresponding author: Dr. Mehdi Kargahi,
School of Electrical and Computer Engineering, College of Engineering, University of Tehran, Tehran, Iran, Iran.