

# Procrastinating Voltage Scheduling with Discrete Frequency Sets\*

Zhijian Lu, Yan Zhang, Mircea Stan, John Lach, Kevin Skadron<sup>§</sup>

Department of Electrical and Computer Engineering, <sup>§</sup>Department of Computer Science  
University of Virginia  
Charlottesville, VA 22904

{zl4j, zhangyan, mircea, jlach}@virginia.edu, skadron@cs.virginia.edu

## Abstract

*This paper presents an efficient method to find the optimal intra-task voltage/frequency scheduling for single tasks in practical real-time systems using statistical workload information. Our method is analytic in nature and proved to be optimal. Simulation results verify our theoretical analysis and show significant energy savings over previous methods. In addition, in contrast to the previous techniques in which all available frequencies are used in a schedule, we find that, by carefully selecting a subset of a small number of frequencies, one can still design a reasonably good schedule while avoiding unnecessary transition overheads.*

## 1. Introduction

With the scaling of semiconductor technology, power consumption has become a serious issue for both high performance and embedded systems. Dynamic voltage scaling (DVS) has become an efficient technique for power reduction due to the quadratic dependence of circuit switching energy on operating voltage.

However, DVS trades off energy with performance. Many researchers propose various voltage scheduling techniques such that energy is minimized while performance is still guaranteed. One of the major difference among these techniques lies in their workload assumptions. For example, Rao and Vrudhula [8] assume the exact amount of computation is known in advance. In reality, different instances of the same task might have various computation requirements. Therefore some work such as [10] uses the worst-case execution time (WCET) to schedule the voltage profile offline and reclaim the slacks on runtime, while others [6] use formal feedback controller to adapt the system to the workload variations in runtime. On the other hand, the amount of computation for the same task could be well characterized by probability density functions. This observation inspires probabilistic approaches in DVS studies [2, 5, 11, 12, 13].

Lorch and Smith [5] show that using statistical information in DVS is superior to other heuristic DVS techniques whose performances are strongly dependent on the workload distribution. In DVS with probabilistic workload information, a good strategy is to begin with a low frequency, and increase the frequency gradually as the task progresses, such that the task can be finished before deadline even in the worst-case. In other words, high voltage (power) operating points are procrastinated during task execution. Following [13], we call this form of voltage scheduling procrastinating DVS in the rest of the paper. Jejurikar *et al.* [4] introduce the concept of “procrastination scheduling” which is different from the subject studied here. They focus on inter-task scheduling policies and look for opportunities to delay the execution of tasks such that system shutdown intervals can be increased, thus minimizing leakage energy consumption, while we study intra-task scheduling technique.

The optimal procrastinating DVS for single task is derived in [2, 5, 12] using an ideal processor model. Our previous work [13] extends their solutions to deal with multiple tasks. However, there are several limitations on these prior works. First they assume voltage/frequency can be continuously scaled, and have to round their solutions to the available frequencies in the real system. The rounding could result in energy-inefficient design when the number of available frequency is relatively small [10]. Second, the overhead of frequency/voltage transition is ignored, which is dangerous for real-time systems and leads to non-optimality in practical systems [1, 7]. Third, system-wide power consumption is not explicitly modeled. In this work, we try to address these practical issues. Recently, Xu *et al.* [11] try to solve the same problem. They adopt a search-based approach to find the optimal scheme, while our solution is analytic in nature and solves the problem very efficiently.

Specifically, we make the following contributions. First, we derive an analytical solution for systems with non-ideal processors when using procrastinating DVS, enabling fast voltage scheduling with arbitrary workload distribution. Our analysis does not assume any specific frequency-voltage relationship (i.e. analytic models), making it suit-

---

\* This work is supported in part by the National Science Foundation under grant Nos. EHS-0410526, CCF-0429765, CCR-0133634 (Career) and CCR-0306404.

able for various systems and different voltage scaling techniques, such as combined supply voltage and body bias scaling [1]. Second, our solutions minimize the total system energy consumption by including both dynamic and static on-chip power as well as off-chip component power. Third, our results indicate that all efficient operating points (i.e. voltage/frequency pairs) in procrastinating DVS must lie on a *convex* energy-delay curve. This interesting observation is helpful in low power system design by avoiding inefficient frequency sets. Finally, we find that, for a given deadline, a small number of frequencies are sufficient for forming a schedule to maintain energy savings comparable to that of using all frequencies while avoiding unnecessary transition overheads.

The rest of the paper is structured as follows. In Section 2, the system model is described and the optimization problem is formulated. We solve the problem in Section 3, and extend it to account for frequency switching overheads in Section 4. We present simulation results in Section 5 and conclude the paper in Section 6.

## 2. System Model and Problem Formulation

### 2.1. Energy Model

In many low-power systems, there are usually two power states: active state and idle state. In the idle state, the system is in a low-power mode with zero clock frequency, and no useful work can be done. Therefore, we model the system power by  $P_{sys} = P_f + P_0$ , where  $P_0$  is the idle power which is a constant and exists whenever the system is powered on, and  $P_f$  is the power used for computation in active state, which is the sum of on-chip and off-chip power consumption, including both leakage and dynamic power. In general,  $P_f$  is dependent both on supply voltage and clock frequency. We define energy efficiency  $e(f) = \frac{P_f}{f}$ , which represents the energy spent on each cycle for computation. It is obvious that energy efficiency is also voltage/frequency dependent. If a task with  $X$  cycles is finished before its deadline  $D$ , the total energy spent in the period  $D$  is  $E_{sys} = \int_0^X e(f)dx + P_0D$ . The second term in the total energy is independent on voltage/frequency scaling. In the following discussion, we will ignore  $P_0$  as if it is zero.

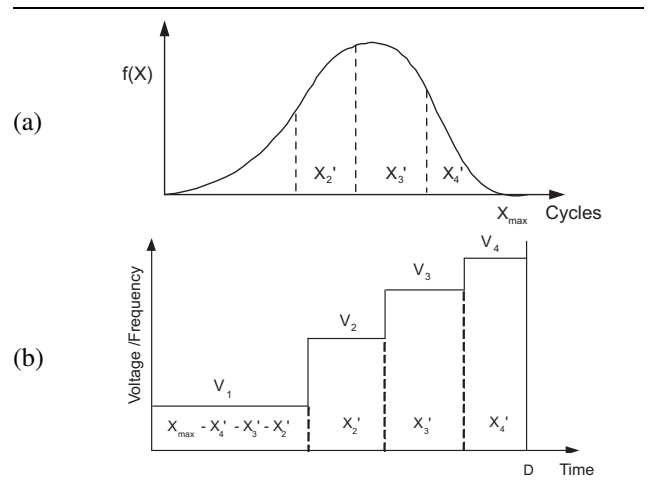
The system is capable of operating on variable voltages/frequencies. Let  $\{f_1, f_2, f_3, \dots, f_r\}$  denotes the set of available frequencies, with total number equal to  $r$ , and we have  $f_1 < f_2 < f_3, \dots < f_r$ . For each frequency point, there is a voltage and power consumption associated with it, and thus a corresponding energy efficiency under that frequency. Therefore, we have a set of energy efficiencies  $e_1, e_2, e_3, \dots, e_r$ , with  $e_1 < e_2 < e_3 < \dots < e_r$ . That is true because if  $f_i < f_j$  and  $e_i \geq e_j$ ,  $f_j$  can finish tasks faster while spend less energy than  $f_i$ , and  $f_i$  becomes inefficient and never used. *Therefore, in a usable frequency set, energy efficiency should be an increasing function of clock frequency, or decreasing function of clock period.*

### 2.2. Procrastinating DVS

A task executed on the system has a deadline  $D$  and its actual amount of computation (clock cycles) is randomly distributed between 0 and  $X_{max}$  and governed by a probability density function (PDF)<sup>1</sup>, which can be obtained, for example, from online profiling. Furthermore, we assume frequency/voltage switching points can be inserted anywhere during task execution. Figure 1 gives an example of procrastinating DVS along with its workload distribution.

**Theorem 1.** *In the optimal procrastinating DVS, the operating frequency is non-decreasing as the number of executed cycles increases.*

We omit the proof for Theorem 1 here due to space limitations. Xu *et al.* [11] presented a similar theorem in their paper. Theorem 1 indicates that in a system with  $r$  operating points, the optimal procrastinating DVS scheduling has at most  $r$  frequency/voltage transitions, as shown in Figure 1(b) for a system with 4 usable frequency/voltage levels. *Therefore the key for the optimal scheduling with discrete frequency sets is to identify the positions where transitions from low frequency to high frequency occur.*



**Figure 1. (a) PDF of task execution cycles. (b) Procrastinating DVS with 4 operating points.**

Xu *et al.* [11] assume that frequency transitions can only happen at some fixed points (e.g. every 10K cycles) and try to find the optimal scheduling by searching a space composed of those transition points. We take a different approach. Observing that a task usually has millions of cycles while the number of frequency transitions is much smaller

<sup>1</sup> We assume the number of cycles executed in each task instance is invariable in spite of the operating frequency. This is a reasonable assumption for CPU-bounded applications. How to model the frequency dependence of clock cycles in memory-bounded applications is left for future investigation.

(i.e. equal to the number of the operating points, which is usually less than 50), the number of cycles executed on each frequency has much finer granularity and can be approximated as continuous. For example, in Figure 1, we use real variables  $X_{max} - X'_4 - X'_3 - X'_2$ ,  $X'_2$ ,  $X'_3$  and  $X'_4$  to represent the number of cycles executed at  $f_1$ ,  $f_2$ ,  $f_3$  and  $f_4$  respectively. And the expected energy consumption could be modeled as:

$$\begin{aligned}
& E(X'_2, X'_3, X'_4) \\
&= e_1 \int_0^{X_{max}-X'_2-X'_3-X'_4} (1-F(y)) dy \\
&\quad + e_2 \int_0^{X'_2} (1-F(X_{max}-X'_4-X'_3-X'_2+y)) dy \\
&\quad + e_3 \int_0^{X'_3} (1-F(X_{max}-X'_4-X'_3+y)) dy \\
&\quad + e_4 \int_0^{X'_4} (1-F(X_{max}-X'_4+y)) dy
\end{aligned} \tag{1}$$

where  $e_i$  is the energy efficiency (i.e. energy/cycle) associated with frequency  $f_i$ , as defined at the beginning of this section.  $F(x)$  is the cumulative distribution function (CDF) of task execution cycles. Therefore, the optimal procrastinating DVS problem can be formulated as a mathematical optimization problem:

$$\begin{aligned}
& \text{Minimize} && E(X'_2, X'_3, X'_4) \\
& \text{Subject to} && \frac{X_{max}-(X'_2+X'_3+X'_4)}{f_1} + \frac{X'_2}{f_2} + \frac{X'_3}{f_3} + \frac{X'_4}{f_4} \leq D
\end{aligned}$$

For simplicity, in the following, we use this example in our calculations, but the results are readily generalizable to systems with  $r$  operating points.

### 3. Solutions for Discrete Frequency Sets

In this section, the optimal procrastinating DVS with discrete frequency sets is solved without considering frequency transition overheads. In the next section, we extend our discussion to account for those overheads.

#### 3.1. Optimal Procrastinating DVS

We apply the Lagrange Multiplier Method to find the number of cycles executed with each operating points. Let

$$\begin{aligned}
& L(X'_2, X'_3, X'_4, \lambda) \\
&= E(X'_2, X'_3, X'_4) \\
&\quad + \lambda \left( \frac{X'_4}{f_4} + \frac{X'_2}{f_2} + \frac{X'_3}{f_3} + \frac{X_{max}-(X'_2+X'_3+X'_4)}{f_1} - D \right)
\end{aligned}$$

We have

$$\begin{aligned}
\frac{\partial L}{\partial \lambda} &= \frac{X'_4}{f_4} + \frac{X'_2}{f_2} + \frac{X'_3}{f_3} + \frac{X_{max}-(X'_2+X'_3+X'_4)}{f_1} - D = 0 \\
\frac{\partial L}{\partial X'_4} &= \frac{\partial E(X'_2, X'_3, X'_4)}{\partial X'_4} + \lambda \left( \frac{1}{f_4} - \frac{1}{f_1} \right) = 0 \\
\frac{\partial L}{\partial X'_2} &= \frac{\partial E(X'_2, X'_3, X'_4)}{\partial X'_2} + \lambda \left( \frac{1}{f_2} - \frac{1}{f_1} \right) = 0 \\
\frac{\partial L}{\partial X'_3} &= \frac{\partial E(X'_2, X'_3, X'_4)}{\partial X'_3} + \lambda \left( \frac{1}{f_3} - \frac{1}{f_1} \right) = 0
\end{aligned} \tag{2}$$

By substituting Equation (1) into (2) and after some simplifications, we obtain:

$$(1-F(X_{max}-X'_4-X'_3-X'_2)) = \lambda \frac{\left(\frac{1}{f_1} - \frac{1}{f_2}\right)}{(e_2 - e_1)} \tag{3}$$

$$(1-F(X_{max}-X'_4-X'_3)) = \lambda \frac{\left(\frac{1}{f_2} - \frac{1}{f_3}\right)}{(e_3 - e_2)} \tag{4}$$

$$(1-F(X_{max}-X'_4)) = \lambda \frac{\left(\frac{1}{f_3} - \frac{1}{f_4}\right)}{(e_4 - e_3)} \tag{5}$$

$$\frac{X_{max}-(X'_2+X'_3+X'_4)}{f_1} + \frac{X'_2}{f_2} + \frac{X'_3}{f_3} + \frac{X'_4}{f_4} = D \tag{6}$$

Therefore, given a deadline  $D$ , the optimal procrastinating DVS schedule can be found by solving Equations (3-6). Transformations of Equations (3-5) reveal an interesting property of the optimal solution, as illustrated in Figure 2. The frequency transition points partition the area of the probability density function such that

$$\begin{aligned}
& \text{area}(II + III + IV) : \text{area}(III + IV) : \text{area}(III + IV) \\
&= (1-F(X_{max}-X'_4-X'_3-X'_2)) \\
&\quad : (1-F(X_{max}-X'_4-X'_3)) \\
&\quad : (1-F(X_{max}-X'_4)) \\
&= \frac{\left(\frac{1}{f_1} - \frac{1}{f_2}\right)}{(e_2 - e_1)} : \frac{\left(\frac{1}{f_2} - \frac{1}{f_3}\right)}{(e_3 - e_2)} : \frac{\left(\frac{1}{f_3} - \frac{1}{f_4}\right)}{(e_4 - e_3)}
\end{aligned}$$

Though it might not be straightforward to solve Equations (3-6) for any given deadline, once the value of  $\lambda$  is determined, solutions can be readily obtained. Here we discuss several special cases for  $\lambda$ , which will be helpful for finding the value of  $\lambda$  for a given deadline.

$\lambda_1 = 0$ . It follows that  $X'_2 = X'_3 = X'_4 = 0$ . Let  $D_1 = \frac{X_{max}}{f_1}$ . The deadline  $D$  in this case is equal to  $D_1$ , and the areas of region (II-IV) are equal to 0.

$\lambda_2 = \frac{(e_2 - e_1)}{\left(\frac{1}{f_1} - \frac{1}{f_2}\right)}$ . As the deadline shrinks from  $D_1$ , higher operating points have to be applied. The area of regions (II-IV) will increase from 0 and keep a constant ratio among them. Finally, the sum of these three regions becomes equal to the total area of the PDF. When this happens, there will be no cycles assigned to  $f_1$ . Let  $D_2$  denote the deadline in this case.

$\lambda_3 = \frac{(e_3 - e_2)}{\left(\frac{1}{f_2} - \frac{1}{f_3}\right)}$ . As the deadline reduces further from  $D_2$ , regions (III and IV) keep increasing and finally occupy the whole area of the PDF. In this case, no cycles are allocated to  $f_1$  and  $f_2$ . Let  $D_3$  denote the deadline in this case.

$\lambda_4 = \frac{(e_4 - e_3)}{\left(\frac{1}{f_3} - \frac{1}{f_4}\right)}$ . When the deadline shrinks more from  $D_3$ , the area of region (IV) increases and finally occupies the whole PDF. In this case, only  $f_4$  is scheduled. Let  $D_4 = \frac{X_{max}}{f_4}$ , which is the minimum deadline this system can satisfy.

In other words, when  $D_1 \leq D$ , the whole task is executed with  $f_1$ , and  $\lambda(D) = \lambda_1 = 0$ . When  $D_2 < D < D_1$ , all four frequencies are scheduled along task execution, and  $\lambda_1 < \lambda(D) < \lambda_2$ . When  $D_3 < D \leq D_2$ , only  $f_2, f_3$  and  $f_4$  are scheduled, and  $\lambda_2 \leq \lambda(D) < \lambda_3, \dots$ , and so on. Finally when  $D = D_4$ , only  $f_4$  is used to execute the whole task. The value of  $\lambda$  increases from its minimum  $\lambda_{min} = 0$  to its maximum  $\lambda_{max} = \frac{(e_4 - e_3)}{(\frac{1}{f_3} - \frac{1}{f_4})}$  as the deadline decreases. In fact, it can be shown that  $\frac{d\lambda}{dD} \leq 0$ . We will know the numerical range of  $\lambda$  when a deadline is given, by comparing the actual deadline to  $D_1$  through  $D_4$ , as well as the operating points to be used in the schedule. In practice, we can represent this numerical range of  $\lambda$  by a set of constant numbers (samples of this range) and find the approximate numerical value of  $\lambda$  very efficiently by the bisection method. Thus, the complexity of our method is  $O(1)$  with respect to the number of bins in the workload distribution histogram and linear to the number of available frequencies/voltages.

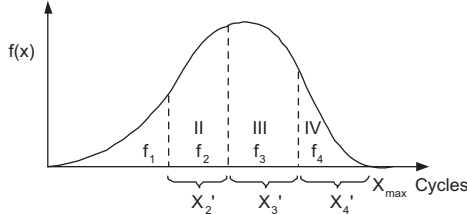


Figure 2. Graphical interpretations of Equations (3), (4) and (5).

### 3.2. Efficient Operating Points

As shown in Figure 2, Equations (3-5) implicitly require that, for  $f_{i-1} < f_i < f_{i+1}$ ,

$$\frac{(e_i - e_{i-1})}{(\frac{1}{f_{i-1}} - \frac{1}{f_i})} < \frac{(e_{i+1} - e_i)}{(\frac{1}{f_i} - \frac{1}{f_{i+1}})} \quad (7)$$

In fact, we have the following theorem.

**Theorem 2.** *In a system with a set of usable operating points  $\{f_1/e_1, f_2/e_2, f_3/e_3, \dots, f_r/e_r\}$ , where  $f_1 < f_2 < f_3, \dots < f_r$  and  $e_1 < e_2 < e_3, \dots < e_r$ , if there exists an operating point  $f_i/e_i$  such that Equation (7) is not hold, there will be no cycles allocated to this operating point in the optimal procrastinating DVS.*

Due to the space constraint, the proof is not presented here. Theorem (2) says that a usable operating point is not necessarily an efficient one. As an example, in Figure 3, we plot the energy (per cycle) vs. delay ( $1/f$ ) curves for some existing processors with DVS capability including IBM PPC405LP ([11]), Intel Xscale ([11]), ARM8 ([3]), AMD Mobile Athlon (our own measurements), and an ideal processor using the “ $\alpha$  current model” [9].  $\frac{(e_i - e_{i-1})}{(\frac{1}{f_{i-1}} - \frac{1}{f_i})}$  represents the slope between two consecutive operating points

in the curves. Equation (7) specifies that, in order for all operating points to be efficient, the slope of the energy–delay curve should decrease as the delay increases, or, all operating points should lie on a convex curve.

As one may expect, the energy efficiency decreases as the delay increases for all processor models, as illustrated by the curves in Figure 3. In the ideal processor model, we also include the static power by assuming that it is about one third of the dynamic power. Figure 3 shows that the operating points of the ideal processor are strictly on a convex curve. Xscale and ARM8 processors follow a similar trend with all operating points being efficient. While PowerPC and Mobile Athlon have some operating points that will never be used in the optimal procrastinating DVS, as indicated in the figure.

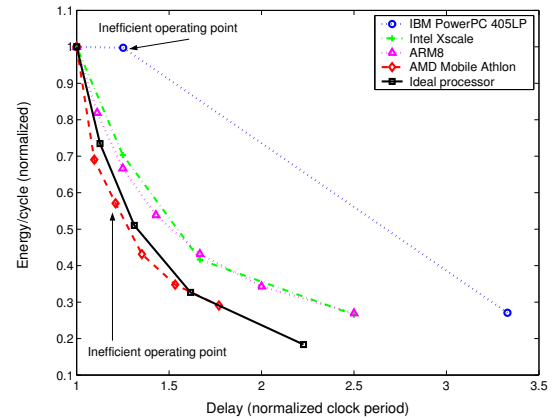


Figure 3. Energy–delay curves for different processors with DVS capability (The values for each processor are normalized to its energy and delay at the full speed.).

## 4. Considering Transition Overheads

### 4.1. Modeling Energy Overheads

As pointed out in [7], the energy overhead due to frequency switching includes idle energy and capacitance charging energy. The idle energy part is already captured by  $P_0$  in Section 2. For capacitance charging energy, it is true that  $E_{ij} = E_{ik} + E_{kj}$ , where  $E_{ij}$  is the energy overhead when switching from  $f_i$  to  $f_j$  and  $f_i < f_k < f_j$ . Let  $E_{total}$  denote the average energy consumption with the existence of energy overheads, and, using a 4-frequency system as an example, it follows

$$E_{total} = E_{ideal} + E_{12}[1 - F(X_m - X'_4 - X'_3 - X'_2)] + E_{23}[1 - F(X_m - X'_4 - X'_3)] + E_{34}[1 - F(X_m - X'_4)] + E_{34}[1 - F(X_m - X'_4)]$$

where  $E_{ideal}$  is the expression for expected energy without overheads (i.e. Equation (1)). Again, using the Lagrange Multiplier Method, the energy-optimal voltage scheduling

is obtained by

$$\begin{aligned}
& (1-F(X_{\max}-X'_4-X'_3-X'_2)) + \frac{E_{12}}{e_2-e_1} f(X_{\max}-X'_4-X'_3-X'_2) \\
&= \lambda \frac{\left(\frac{1}{f_1}-\frac{1}{f_2}\right)}{(e_2-e_1)} \\
& (1-F(X_{\max}-X'_4-X'_3)) + \frac{E_{23}}{e_3-e_2} f(X_{\max}-X'_4-X'_3) = \lambda \frac{\left(\frac{1}{f_2}-\frac{1}{f_3}\right)}{(e_3-e_2)} \\
& (1-F(X_{\max}-X'_4)) + \frac{E_{34}}{e_4-e_3} f(X_{\max}-X'_4) = \lambda \frac{\left(\frac{1}{f_3}-\frac{1}{f_4}\right)}{(e_4-e_3)} \\
& \frac{X_{\max}-(X'_2+X'_3+X'_4)}{f_1} + \frac{X'_2}{f_2} + \frac{X'_3}{f_3} + \frac{X'_4}{f_4} = D
\end{aligned} \tag{8}$$

Usually the energy overhead  $E_{ij}$  is small, e.g., using the data in [7],  $\frac{E_{ij}}{e_j-e_i} \approx 7K(\text{cycles})$ , thus  $\frac{E_{ij}}{e_j-e_i} f(x) \approx F(x) - F\left(x - \frac{E_{ij}}{e_j-e_i}\right)$ . Letting  $\phi(x) = F\left(x - \frac{E_{ij}}{e_j-e_i}\right)$ , Equation (8) can be reduced to the same form as Equations (3–6) by substituting  $F(x)$  with  $\phi(x)$ .

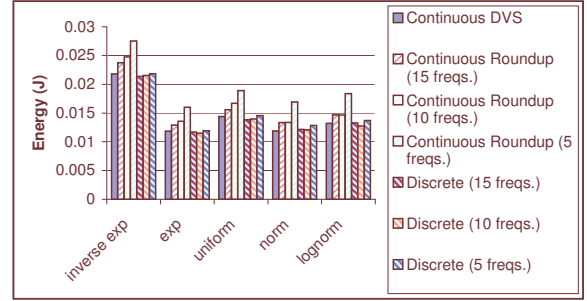
## 4.2. Combined Overheads

Transition time overheads are hard to model using continuous functions. Eliminating an operating point in the scheduling might increase the energy, but it also increases the allowed execution time by releasing the transition time, thus reducing energy. When the number of available operating points is small. We can adopt a brute-force approach. Using the 4-frequency system as an example, we first assume all frequencies are applied and the actual deadline is adjusted by  $D_{\text{actual}} = D - 3t_o$ , where  $t_o$  is the time overhead for one transition. We can find the optimal scheduling in this case by Equation (8). Then we allow only two transitions to happen in task execution, and we solve Equation (8) for all combinations of three operating points, similarly for one transition or even no transitions. Finally we choose the schedule with the lowest energy consumption. This brute-force approach requires  $O(2^r)$  ( $r$  is the number of potential operating points) iterations of solving Equation (8) and is therefore not effective when  $r$  is large.

When the number of efficient operating points is large, we propose a heuristic search algorithm based on dynamic programming. The detailed algorithm is not presented here due to space limitation. We explain the key ideas behind our algorithm. First, we find a schedule using all frequencies by Equation (8), with the deadline adjusted by the transition times. This is the baseline schedule. Second, we define  $\text{energy\_reduction}(f_i)$  as the energy reduction after the elimination of  $f_i$  from the baseline schedule. When  $f_i$  and  $f_j$  are not consecutive in the baseline, it is approximated that  $\text{energy\_reduction}(f_i, f_j) = \text{energy\_reduction}(f_i) + \text{energy\_reduction}(f_j)$ . Therefore, if  $\text{energy\_reduction}(f_i) < \text{energy\_reduction}(f_k)$ , it is established that  $\text{energy\_reduction}(f_i, f_j) < \text{energy\_reduction}(f_k, f_j)$  when  $f_j$  is not next to either  $f_i$  or  $f_k$  in the baseline schedule. Thereby, we can reduce the search space without examining the elimination of both  $f_i$  and  $f_j$  from the baseline. The complexity of our algorithm is  $O(r^3)$ . In the next section, we use this algorithm to find schedules with a subset of frequencies when the number of transitions is fixed.

## 5. Experimental Results

In this section, we compare the proposed method with previous techniques. Since transition overheads were not considered in most prior work, in our experiments, we also ignore energy/time overheads for all schemes in comparison. The simulations use a processor model that provides finite frequencies linearly spaced between  $500\text{MHz}$  and  $1.5\text{GHz}$ , and a power model  $P = \alpha f_{\text{op}}^3$ . We assume the maximum number of cycles in a task is 1,000,000, and generate 5000 task instances for each different workload distribution. The PDFs for the voltage scheduling are obtained from profiling those 5000 task instances using a histogram with  $10K$  cycles in each bins.

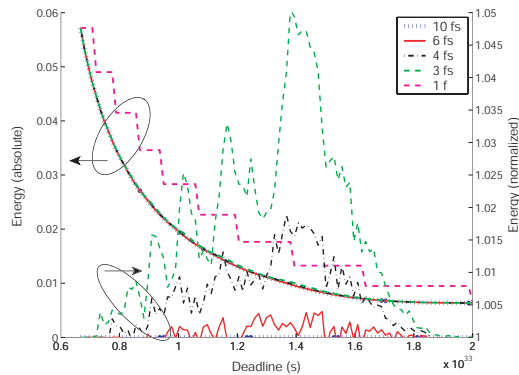


**Figure 4. Energy consumption by different scheduling techniques for workloads with different cycle count distributions.**

Figure 4 presents the average task energy of different task types (distributions) by simulating the execution of each task with different scheduling techniques. The deadline for all tasks is set to  $0.00125s$ . Three techniques are compared: 1) DVS with continuous voltage scaling between the maximum and minimum frequencies [2, 5], 2) Continuous scaling rounded up to the available frequencies, used in prior work [12, 13] to apply procrastinating DVS in practical systems, 3) Optimal procrastinating DVS with discrete frequency sets as proposed in this paper. As indicated by Figure 4, the performance of simple rounding method degrades very quickly as the number of available frequencies is reduced. On the other hand, our solution is quite robust across different numbers of available frequencies. On average, the energy savings brought by our method over the previous rounding method range between 10% and 24% from a 15-frequency set system to a 5-frequency set system.

For a processor with  $r$  efficient operating points, the optimal procrastinating DVS will use by default all frequencies if necessary. A heuristic way to account for switching overheads is to limit the number of frequencies used in the schedule by, say,  $i$  frequencies. Figure 5 plots the average energy consumption for tasks of normal distributions at different deadlines with the number of frequencies used in a schedule being fixed to 6, 4, 3, and 1, respectively, though

the processor is capable of 10 different frequencies. This figure implies that, for a given deadline, a small number of carefully selected frequencies can form a schedule achieving energy savings very close to that of a schedule consisting of the full frequency set (as shown by the right y-axis). Similar results are observed for other workload distributions.



**Figure 5. Average task energy consumption at various deadlines when the number of frequencies in a schedule is fixed (The y-axis on the right shows the energy values normalized to those of using the full frequency set (10 fs)).**

When the total number of efficient frequencies is large, it is computationally expensive to find a good subset of frequencies for scheduling when the allowed number of transitions is fixed. We compare our dynamic programming based heuristic method (subsection 4.2) to a brute-force approach. For a system model with 10 total frequencies, experimental results show that our heuristic method is 5 times faster to find “good” schedules for different numbers of allowed frequencies. The average task energy consumption using the schedules found by our method is very close to that found by the brute-force method, and the maximum deviation we observed is within 2%. In fact, the data presented in Figure 5 are based on the schedules found by our heuristic method.

## 6. Conclusion

In this paper, we introduced the optimal procrastinating DVS for systems with discrete frequency sets. Our techniques efficiently solve the practical issues encountered by previous techniques. The complexity of our method is  $O(1)$  with respect to the number of bins in the task workload histogram, and linear to the number of operating points involved in the schedule. We also find that not all available operating points are necessarily efficient even though they consume less power. We extend our method to deal with transition overheads and find that, for a given deadline, a small number of carefully selected frequencies may be sufficient to form a good schedule achieving energy savings comparable to that using all efficient operating points. We

develop an efficient heuristic algorithm to find good schedules composed of only a subset of available frequencies, thereby avoiding unnecessary transition overheads. Thus, our techniques are readily applicable to various practical systems.

## References

- [1] A. Andrei, M. T. Schmitz, P. Eles, Z. Peng, and B. M. Al-Hashimi. Overhead-conscious voltage selection for dynamic and leakage power reduction of time-constraint systems. In *Proc. of Design, Automation and Test Europe Conference (DATE2004)*, 2004.
- [2] F. Gruian. Hard real-time scheduling for low-energy using stochastic data and dvs processors. In *Proceedings of International Symposium on Low Power Electronics and Design 2001 (ISLPED'01)*, August 2001.
- [3] C. Im and S. Ha. Dynamic voltage scheduling with buffers in low-power multimedia applications. *ACM Transactions on Embedded Computing Systems (TECS)*, 3(4):686–705, November 2004.
- [4] R. Jejurikar and R. Gupta. Procrastination scheduling in fixed priority real-time systems. In *Proc. of the 2004 ACM conference on Languages, compilers, and tools for embedded systems (LCTES'04)*, June 2004.
- [5] J. R. Lorch and A. J. Smith. PACE: A new approach to dynamic voltage scaling. *IEEE Tran. on Computers*, 53(7):856–869, July 2004.
- [6] Z. Lu, J. Lach, M. Stan, and K. Skadron. Reducing multimedia decode power using feedback control. In *Proc. of International Conference on Computer Design*, pages 489–96, October 2003.
- [7] B. Mochocki, X. Hu, and G. Quan. A unified approach to variable voltage scheduling for nonideal dvs processors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23(9):1370–1377, September 2004.
- [8] R. Rao and S. Vrudhula. Energy optimal speed control of devices with discrete speed sets. In *Proc. of the 42nd annual conference on Design automation*, June 2005.
- [9] T. Sakurai and A. Newton. Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas. *IEEE Journal of Solid-State Circuits*, 25(2):584–594, April 1990.
- [10] V. Swaminathan and K. Chakrabarty. Network flow techniques for dynamic voltage scaling in hard real-time systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 23:1385–1398, October 2004.
- [11] R. Xu, C. Xi, R. Melhem, and D. Moss. Practical pace for embedded systems. In *Proc. of the 4th ACM International Conference on Embedded Software(EMSOFT'04)*, 2004.
- [12] W. Yuan and K. Nahrstedt. Energy-efficient soft real-time CPU scheduling for mobile multimedia systems. In *Proc. of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, October 2003.
- [13] Y. Zhang, Z. Lu, J. Lach, K. Skadron, and M. R. Stan. Optimal procrastinating voltage scheduling for hard real-time systems. In *Proc. of the 42nd annual conference on Design automation*, June 2005.