Optimized Slowdown in Real-Time Task Systems via Geometric Programming

Almir Mutapcic[†], Srinivasan Murali[†], Stephen Boyd[†], Rajesh Gupta[‡], David Atienza[+§], Giovanni De Micheli[+]
[†]Department of Electrical Engineering, Stanford University, Stanford, USA, {almirm,smurali,boyd}@stanford.edu
[‡]Department of Computer Science and Engineering, UCSD, San Diego, USA, rgupta@ucsd.edu
[+]LSI, EPFL, Lausanne, Switzerland, {david.atienza, giovanni.demicheli}@epfl.ch
[§]DACYA, Complutense University of Madrid (UCM), Madrid, Spain

***Abstract*— In [1], Jejurikar and Gupta investigated energy savings due to optimal slowdown of periodic tasks in real-time task systems, where tasks have varying power characteristics and task deadlines are less than the periods. The authors presented a bisection method for computing near-optimal constant slowdown factors, when all the tasks are assigned the same slowdown factor. For the case when tasks have different slowdown factors, they presented a method for computing near-optimal slowdown factors as a solution to a convex optimization problem, using the ellipsoid method. In this note, we show a method to cast the problem of finding near-optimal slowdown factors that minimize the total energy consumption as a geometric program (GP), which can be efficiently solved using modern interior-point methods. More importantly, we show that the problem of finding near-optimal constant slowdown factors has an analytic solution. We demonstrate the GP approach by solving several numerical instances using a publicly available interior-point GP solver.**

***Index Terms*— EDF scheduling, real-time systems, low power scheduling, dynamic voltage scaling, slowdown factors, convex optimization, geometric programming.**

I. Introduction

We consider a real-time task system with a single processor as in [1], and briefly describe the problem setup for completeness. When the peak processing capacity is not needed for an application, we can reduce the system energy consumption by operating the processor at a lower frequency than the maximum possible frequency. Such slowdown, using frequency and voltage scaling, is widely deployed in real-time systems [2]. We assume that the amount of slowdown is constant during the execution of a task instance, and we refer to it as the *task slowdown factor*.

The goal of the *optimized slowdown* problem is to find the slowdown factors for each task such that we minimize the energy consumption of the system, while satisfying the task deadlines. In [1], Jejurikar and Gupta addressed two instances of the slowdown problem: one with *constant slowdown*, where all the tasks are assigned the same slowdown factor, and the other with *uniform slowdown*, where all instances of a task have the same slowdown factor, but different tasks can have different slowdown factors. The authors use a bisection method to solve the constant slowdown problem, and the ellipsoid method to solve the uniform slowdown problem, after formulating it as a convex optimization problem.

In this note, we cast the uniform slowdown problem as a *geometric program* (GP), with the slowdown factors, processor voltages, and task execution times as the optimization variables. We also give an explicit solution to the constant slowdown problem. We use GPPOSY [3], a publicly available interior-point GP solver, to solve instances of uniform slowdown problems with hundreds of tasks in a couple of seconds (on a generic personal computer). Compared to the ellipsoid method, which can be slow in practice [4], our approach is far faster.

A. Task system model

We briefly present the task system model from [1]. The real-time task system processes a set of n periodic tasks on a single processor, where we denote the task set as $\Gamma = \{\gamma_1, \ldots, \gamma_n\}$. A task γ_i is a 3-tuple (T_i, D_i, C_i), where T_i is the period of the task, D_i is the relative deadline with $D_i \leq T_i$, and C_i is the worst-case execution time (WCET) for the task, with the processor is running at its maximum (speed) frequency $f_{\max}$. Without loss of generality, we can assume that T_i, D_i, and C_i are natural numbers [5]. The hyper-period of the task set, H, is defined as the least common multiple (lcm) of the task periods. The tasks are scheduled using a preemptive Earliest-Deadline First (EDF) scheduling policy and we assume that the task set Γ is ordered in terms of the non-decreasing deadlines. Finally, we assume that all tasks are independent.

The execution time of the task i under slowdown, denoted τ_i, is

$$\tau_i = \frac{C_i}{\eta_i}. \tag{1}$$

Here, η_i is the *slowdown factor* for the task i and is given by

$$\eta_i = \frac{f_i}{f_{\max}}, \tag{2}$$

where f_i is the operating frequency of the processor during the execution of the task i. The task slowdown factor can be viewed as the normalized operating frequency: at any given instance, it is the ratio of the current operating frequency to the maximum operating frequency. For example, if we run the processor at half its maximum frequency, then $\eta = 1/2$, and the execution time of the current task is doubled, *i.e.*, $1/\eta = 2$. We always have $f_i \leq f_{\max}$, $\eta_i \leq 1$, and $\tau_i \geq C_i$.

B. Feasibility tests

A task system is said to be feasible if all tasks meet their deadlines. In [1], two methods are used to determine the feasibility of the task system: Devi Test Method (DTM) and Optimal Test Method (OTM). The constraints that need to be satisfied for the DTM [6] are given by:

$$F_k^{\mathrm{dtm}}(\tau) = \sum_{i=1}^{k} \frac{\tau_i}{T_i} + \frac{1}{D_k} \sum_{i=1}^{k} \frac{(T_i - D_i)}{T_i} \tau_i \leq 1,\ k = 1, \ldots, n, \tag{3}$$

and for the OTM [5] are given by:

$$F^{\mathrm{otm}}(\tau, t) = \sum_{i=1}^{n} \left(\left\lfloor \frac{t - D_i}{T_i} \right\rfloor + 1 \right) \tau_i \leq t,\ t \leq t_{\max},\ t \in \mathcal{T}, \tag{4}$$

where $\mathcal{T} = \{kT_i + D_i \mid i = 1, \ldots, n,\ k \geq 0\}$ and $t_{\max} = H$ is the hyper-period of the task set.

The computational complexity of DTM is significantly less than that for OTM. In DTM we have to check feasibility of n linear constraints in the variable $\tau = (\tau_1, \ldots, \tau_n)$, while the number of constraints in OTM can grow exponentially in the number of variables. However, note that DTM is only a sufficient feasibility test that can identify a large percentage of feasible task sets; it will produce results that are less accurate than OTM. In [1], an approximation to OTM is given, where the number of constraints is limited by using

$$t_{\max} = \frac{1}{\epsilon} \max_i (T_i - D_i) < H, \tag{5}$$

which gives pseudo-polynomial number of constraints in (4). In this case, OTM will identify feasible task sets that will achieve total system utilization up to $(1 - \epsilon) \times 100\,\%$, where $0 < \epsilon < 1$ is the measure of suboptimality.

C. Energy model

We will assume that the energy consumption of the system is dominated by the dynamic power consumption, which arises due to switching activity in circuits, while the static power consumption, which is present even when no logic operations are performed, is negligible. The power consumption of the system implemented in CMOS technology is given by

$$P = C_{\mathrm{eff}} V_{\mathrm{dd}}^2 f,$$

where C_{eff} is the effective switching capacitance, V_{dd} is the operating supply voltage, and f is the operating frequency. We obtain the energy spent in the system by integrating power over the time of operation. For example, if the system is running at a constant operating frequency f and voltage V_{dd} for the period of T seconds, then the energy spent is

$$E = C_{\mathrm{eff}} V_{\mathrm{dd}}^2 f T.$$

The energy spent during the execution of the task i is

$$E(V_{\mathrm{dd}}) = C_{\mathrm{eff}} V_{\mathrm{dd}}^2 f_i \tau_i,$$

which simplifies to

$$E(V_{\mathrm{dd}}) = N_i V_{\mathrm{dd}}^2, \tag{6}$$

where $N_i = C_{\mathrm{eff}} C_i f_{\max}$. Thus the energy spent depends only on the supply voltage and the given constants for task i.

D. Voltage and slowdown factor relationship

The transistor gate delay increases with a decrease in voltage, forcing a decrease in the operating frequency. The relationship between the transistor gate delay, t_{inv}, and supply voltage is given by:

$$t_{\mathrm{inv}} = \frac{k V_{\mathrm{dd}}}{(V_{\mathrm{dd}} - V_{\mathrm{th}})^{\alpha}}, \tag{7}$$

where V_{dd} is the operating supply voltage, V_{th} is the threshold voltage, α has a value in the range 1 to 2, and k is a technology constant.

From equations (2) and (7), we obtain the following relationship between the supply voltage V_i and the slowdown factor for the task i:

$$\eta_i = \frac{(V_i - V_{\mathrm{th}})^{\alpha}}{\kappa_i V_i}, \tag{8}$$

where $\kappa_i = (V_{\max} - V_{\mathrm{th}})^{\alpha} / V_{\max}$ is a constant, and $V_{\max}$ is the operating supply voltage of the system at the maximum operating frequency, which is a known constant.

II. Optimized Slowdown Problem

In this section, we formulate the optimized slowdown problem in GP form. We refer the reader to [7, Sec. 4.5], [8], [9] for an introduction to geometric programming, some of the main approaches used to formulate problems in GP form, and for various examples and applications. From this point on, we will assume that the reader is familiar with terms such as monomial, posynomial, *etc.*

A. Uniform slowdown problem

In the uniform slowdown problem we wish to choose the slowdown factors $\eta_1, \ldots, \eta_n$ that minimize the energy consumption of the system, while satisfying the feasibility of the tasks. We also want to determine the values of variables $V_1, \ldots, V_n$, which are supply voltages during the execution of the task i, and the task execution times $\tau_1, \ldots, \tau_n$. We note that all of these variables are positive.

The objective is the total energy consumption of the system, which is obtained by summing equation (6) for each task,

$$E(V) = \sum_{i=1}^{n} N_i V_i^2. \tag{9}$$

This is a quadratic function and also a posynominal in V.

The problem constraints are the task feasibility constraints (where we can use DTM or OTM constraints), lower and upper bounds on the variables, and equality relationships between the slowdown factors, voltages, and task execution times.

Finally, the uniform slowdown problem with DTM constraints is given by

$$\begin{array}{ll} \text{minimize} & E(V) \\ \text{subject to} & F_i^{\mathrm{dtm}}(\tau) \leq 1, \quad i = 1, \ldots, n \\ & C_i / \eta_i = \tau_i \\ & (V_i - V_{\mathrm{th}})^{\alpha} / (\kappa_i V_i) = \eta_i \\ & V_{\min} \leq V_i \leq V_{\max}, \quad \eta_{\min} \leq \eta_i \leq 1, \end{array} \tag{10}$$

where $V_{\min} > V_{\mathrm{th}}$ is the minimum allowable operating supply voltage, and $\eta_{\min}$ is the corresponding minimum slowdown.

Here we can also add bounds on τ, but they are redundant. The uniform slowdown problem with OTM constraints is the same as the above problem except that the feasibility constaints (3) are replaced by constraints (4).

Problem (10) is not in GP form since the voltage/slowdown relationship (8) is not GP compatible (valid GP problems can only have monomial equalities). However, we can formulate an equivalent GP problem which is given by

$$\begin{array}{ll} \text{minimize} & E(V) \\ \text{subject to} & F_i^{\text{dtm}}(\tau) \leq 1, \quad i = 1, \ldots, n \\ & C_i/\eta_i = \tau_i \\ & V_{\text{th}} + (\eta_i \kappa_i V_i)^{1/\alpha} \leq V_i \\ & V_{\min} \leq V_i \leq V_{\max}, \quad \eta_{\min} \leq \eta_i \leq 1. \end{array} \tag{11}$$

This is a valid GP since the objective is a posynomial, there are n monomial equalities, and all of the inequalities are either monomial or posynomial inequalities.

Next we show that (10) and (11) are equivalent problems. The slowdown factor η_i (8) is a monotone increasing function of voltage V_i, since for $\alpha \geq 1$ and $V_i > V_{\text{th}}$ we have

$$\frac{d\eta_i}{dV_i} = \frac{(V_i - V_{\text{th}})^{\alpha} \left(\alpha V_i/(V_i - V_{\text{th}}) - 1\right)}{\kappa V_i^2} > 0.$$

The slowdown factor is also a concave function in V, which together with monotonicity can be used to cast problem (10) as a convex problem. However, here we are mainly interested in how to convert this equality into a posynomial inequality, and turn the problem into a GP. We relax the equality (8) in problem (10) as

$$(V_i - V_{\text{th}})^{\alpha}/(\kappa_i V_i) \geq \eta_i \tag{12}$$

and argue that this inequality is always tight at the optimal solution (*i.e.*, it is met with equality). Assume that this is not the case, and that there is an optimal V_i such that the above inequality is slack. We can then decrease the objective value (it is a quadratic) by reducing V_i, which reduces the value of the lefthand side, until the inequality is tight. Then, by contradiction, the relaxed inequalities are always tight at the optimal solution, and the optimal solution of the relaxed problem is also an optimal solution of the original problem (10). After algebraic manipulation, the inequality (12) is equivalent to

$$V_{\text{th}} + (\eta_i \kappa_i V_i)^{1/\alpha} \leq V_i,$$

which is a posynomial inequality and is compatible with GP. We conclude that the problems (10) and (11) are equivalent.

B. Constant slowdown problem

The *constant slowdown problem* is a special case of the uniform slowdown problem, in which the same slowdown factor is applied to all task instances in the system. Therefore the goal of the constant slowdown problem is to choose the overall slowdown factor η that minimize the energy consumption of the system, while satisfying feasibility of the tasks. In addition, we also want to determine the value of the optimal supply voltage V and the task execution times $\tau_1, \ldots, \tau_n$, *i.e.*, we seek to solve

$$\begin{array}{ll} \text{minimize} & NV^2 \\ \text{subject to} & F_i^{\text{dtm}}(\tau) \leq 1, \quad i = 1, \ldots, n \\ & C_i/\eta = \tau_i \\ & (V - V_{\text{th}})^{\alpha}/(\kappa V) = \eta \\ & V_{\min} \leq V \leq V_{\max}, \quad \eta_{\min} \leq \eta \leq 1, \end{array} \tag{13}$$

where $N = \sum_{i=1}^{n} N_i$ is a constant.

We can give an explicit solution to this problem. We first note that the objective is to find the smallest possible voltage V subject to the problem constraints, which translates to finding the smallest possible η (since η is an increasing one-to-one function of V on the domain $[\eta_{\min}, 1]$). Thus, an equivalent problem is

$$\begin{array}{ll} \text{minimize} & \eta \\ \text{subject to} & F_i^{\text{dtm}}(C_i) \leq \eta, \quad i = 1, \ldots, n \\ & \eta_{\min} \leq \eta \leq 1, \end{array} \tag{14}$$

where we have substituted $\tau_i = C_i/\eta$ in the feasibility constraints and pulled η to the righthand side (it factors out). Clearly, the optimal slowdown factor in the DTM case is

$$\eta^{\text{dtm}} = \min\{\eta_{\min}, F_i^{\text{dtm}}(C_i)\}, \quad i = 1, \ldots, n,$$

while in the OTM case is

$$\eta^{\text{otm}} = \min\{\eta_{\min}, F_i^{\text{otm}}(C_i, t)/t\}, \quad t \leq t_{\max},\, t \in \mathcal{T}.$$

III. Experimental Results

The GP problem (11) can be efficiently solved using modern interior-point methods [7, Chap. 11], [10], [11]. For our numerical experiments we use GPPOSY, a publicly available Matlab implementation of an interior-point method for solving GPs in posynomial form [3]. The GPPOSY solver can easily handle problems with hundreds of tasks and thousands of feasibility constraints. For simple prototyping, we also solved smaller problems with up to ten tasks using CVX [12].

We consider the same experimental setup as in [1], *e.g.*, we randomly generate task sets with random periods in the range $[20000, 50000]$, WCETs in the range $[100, 5000]$, and deadlines which are about $1/4$ of the task periods. We take $V_{\text{th}} = 0.36$V and $\alpha = 1.5$, and use an operating voltage range with $V_{\min} = 0.6$V and $V_{\max} = 1.8$V. We use $\epsilon = 0.01$ to obtain 1% near-optimal solutions for every experimental run.

The results are summarized in figure 1, with the plot showing the mean and standard deviation for the total time required to find optimized slowdown over 50 randomly generated task system with n tasks. Figure 2 shows the average number of GP problem constraints as n increases, which explains variation in the solution times for randomly generated problems with n larger than 20 tasks.

References

[1] R. Jejurikar and R. Gupta, "Optimized slowdown in real-time task systems," *IEEE Transactions on Computers*, vol. 55, pp. 1588–1598, Dec. 2006.

[2] 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 Symposium*, pp. 95–105, Dec. 2001.

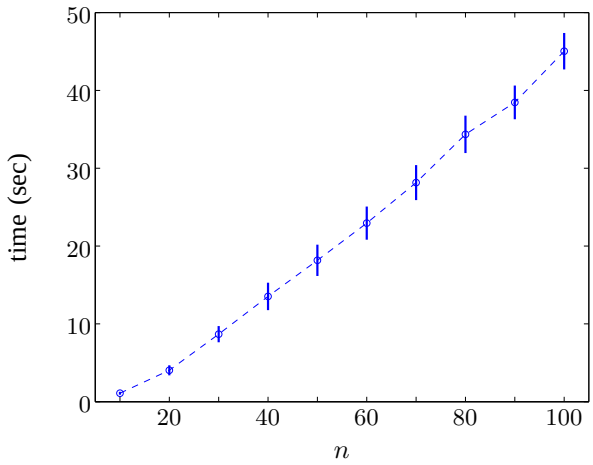

Fig. 1. Average time taken by the solver to solve 50 randomly generated task systems with increasing number of tasks n. Error bars show standard deviation, around the average value, for each value of n.

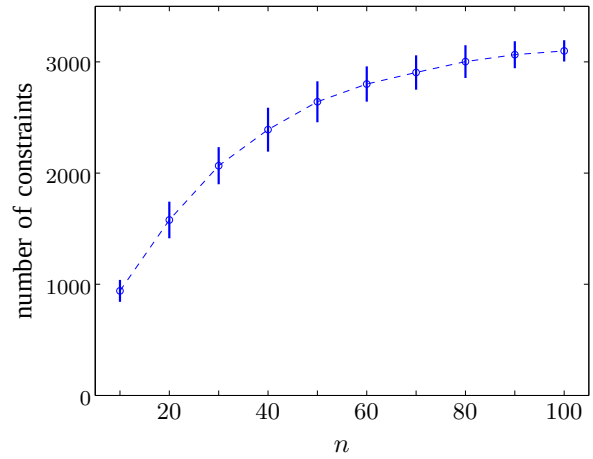

Fig. 2. Average number of GP constraints (size of the problem) for 50 randomly generated task systems with increasing number of tasks n. Error bars show standard deviation, around the average value, for each value of n.

[3] K. Koh, S.-J. Kim, A. Mutapcic, and S. Boyd, "GPPOSY: A Matlab solver for geometric programs in posynomial form." Available at `www.stanford.edu/~boyd/ggplab/`, 2006.

[4] R. Bland, D. Goldfarb, and M. Todd, "The ellipsoid method: A survey," *Operations Research*, vol. 29, no. 6, pp. 1039–1091, 1981.

[5] S. Baruah, A. Mok, and L. Rosier, "Algorithms and complexity concerning the preemptively scheduling of periodic, real-time tasks on one processor," *Real-Time Systems Journal*, vol. 2, pp. 301–324, 1990.

[6] U. Devi, "An improved schedulability test for uniprocessor periodic task systems," in *Proc. of 15th Euromicro Conference on Real-Time Systems*, pp. 23–30, July 2003.

[7] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[8] S. Boyd, S.-J. Kim, L. Vandenberghe, and A. Hassibi, "A tutorial on geometric programming," *Optimization and Engineering*, vol. 8, pp. 67–127, Mar. 2007.

[9] S. Boyd, S.-J. Kim, D. Patil, and M. Horowitz, "Digital circuit optimization via geometric programming," *Operations Research*, vol. 53, no. 6, pp. 899–932, 2005.

[10] S. Wright, *Primal-dual interior-point methods*. Philadelphia, PA, USA: SIAM, 1997.

[11] Y. Nesterov and A. Nemirovsky, *Interior-point polynomial methods in convex programming*, vol. 13 of *Studies in Applied Mathematics*. Philadelphia, PA: SIAM, 1994.

[12] M. Grant, S. Boyd, and Y. Ye, "CVX: Matlab software for disciplined convex programming, version 1.0RC3." Available at `www.stanford.edu/~boyd/cvx/`, Feb. 2007.