2008 International Symposium on Nonlinear Theory and its Applications
NOLTA'08, Budapest, Hungary, September 7-10, 2008

# Power Aware Elastic Scheduling with the Resolution of Trade-off between CPU Power Consumption and Task Performance

Sayuri Terada† and Toshimitsu Ushio†

†Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka, 560-8531 Japan
Email: terada@hopf.sys.es.osaka-u.ac.jp

**Abstract**—Using dynamic voltage scaling (DVS) technique, we can reduce the CPU power consumption by setting a low CPU frequency. In this paper, we improve the elastic scheduling proposed by Marinoni and Buttazzo. We adopt a force compressing tasks as the performance degradation and resolve a trade-off between the power consumption and the performance of tasks with desired rate. We propose a power-aware elastic scheduling algorithm, by which we optimize an objective function given by weighted sum of the power consumption and the force. By simulation, we demonstrate the efficiency of the proposed algorithm. Its computational complexity is polynominal.

## 1. Introduction

Since embedded systems have been spread to various engineering fields, the reduction of their power consumption has been one of the important issues. The CPU using dynamic voltage scaling (DVS) can set the supply voltage and the CPU frequency to specified discrete levels and lead to degradation of the power consumption by setting lower levels [1, 2]. However, it also causes increase of the task execution times and the overload situation.

Flexible task models can vary the task execution times so that the achieved performance may change [3]. For example, in the elastic task model, the CPU utilizations of the tasks are modified by changing their periods in each admissible range [4, 5]. Using flexible task models, we can achieve dramatic power saving with avoiding the deadline miss. Marinoni and Buttazzo introduce the elastic task model to a DVS management where the CPU frequency is selected according to a strategy [6], and they generalize it to dynamical scheduling [7]. They propose three strategies: Energy saving mode, High performance mode, and User mode. However, in User mode, user decides the CPU frequency with no precise indicators. In this paper, we propose an optimization-based method for the DVS management to resolve the trade-off.

The rest of paper is organized as follows: Section 2 reviews elastic scheduling proposed in [6]. Section 3 introduces an objective function. In Section 4, we propose an algorithm minimizing the objective function and verify the suspensive conditions. Section 5 presents a simulation result and Section 6 concludes the paper.

## 2. Elastic Scheduling

The elastic scheduling proposed in [6] is composed by three parts, calculating the speed selection range, setting the operating speed according to a selected strategy, and computing task periods. In this section, we review the task model and these three procedures. A CPU frequency is assumed to take $m$ discrete levels in a frequency range $[f_{min},\ f_{max}]$, and we set the supply voltage as the minimum level compatible with the CPU frequency. Processor speed is defined as normalized frequency $s = f/f_{max}$ ($s \in [s_{min},\ s_{max}] = [f_{min}/f_{max},\ 1]$).

### 2.1. Task Model

In [6], the task execution time is generalized to be divided into two parts: one is dependent on the CPU frequency, and the other is independent. The task set $\{\tau_1, \tau_2, \cdots, \tau_n\}$ is assumed to be independent and periodic. Each task $\tau_i$ is characterized by a 5-tuple $\tau_i(C_{i_{max}}, \phi_i, T_{i_{min}}, T_{i_{max}}, E_i)$, where $C_{i_{max}}$ is the execution time at the maximum processor speed $s_{max}$, $\phi_i$ is the rate of the frequency-dependent part of the execution time, $T_{i_{min}}$ is the minimum period, $T_{i_{max}}$ is the maximum period, and $E_i$ is the elastic coefficient.

The task execution time is expressed as

$$C_i(s) = \frac{\phi_i C_{i_{max}}}{s} + (1-\phi_i)C_{i_{max}}. \tag{1}$$

Let $T_i$ be the actual period of task $\tau_i$. Then the CPU utilization of task $\tau_i$ is expressed as

$$U_i(s) = \frac{C_i(s)}{T_i} = \frac{\phi_i C_{i_{max}}}{sT_i} + \frac{(1-\phi_i)C_{i_{max}}}{T_i}. \tag{2}$$

Task period $T_i$ varies continuously in the range $[T_{i_{min}},\ T_{i_{max}}]$ and the CPU utilization $U_i$ varies in $[U_{i_{min}}(s),\ U_{i_{max}}(s)]$, where

$$U_{i_{min}}(s) = \frac{\phi_i C_{i_{max}}}{sT_{i_{max}}} + \frac{(1-\phi_i)C_{i_{max}}}{T_{i_{max}}},$$
$$U_{i_{max}}(s) = \frac{\phi_i C_{i_{max}}}{sT_{i_{min}}} + \frac{(1-\phi_i)C_{i_{max}}}{T_{i_{min}}}.$$

The parameter $\phi_i$ can be obtained by measuring $C_{i_{max}} = C_i(s_{max})$ and $C_{i_{min}} = C_i(s_{min})$, that is, from Eq. (1), we have

$$C_{i_{min}} = \frac{\phi_i C_{i_{max}}}{s_{min}} + (1-\phi_i)C_{i_{max}} \Leftrightarrow \phi_i = \frac{C_{i_{min}} - C_{i_{max}}}{C_{i_{max}}} \frac{s_{min}}{1-s_{min}}.$$

### 2.2. Speed selection by three strategies

The speed selection range $[s_e,\ s_p]$ is calculated as

$$s_e = \min_k\{s_k | s_k \geq s_e^*\},\ s_p = \min_k\{s_k | s_k \geq \min(s_p^*, s_{max})\},$$

where $s_e^* = \dfrac{\sum_{i=1}^n \frac{\phi_i C_{i_{max}}}{T_{i_{max}}}}{U_d - \sum_{i=1}^n \frac{(1-\phi_i)C_{i_{max}}}{T_{i_{max}}}},\ s_p^* = \dfrac{\sum_{i=1}^n \frac{\phi_i C_{i_{max}}}{T_{i_{min}}}}{U_d - \sum_{i=1}^n \frac{(1-\phi_i)C_{i_{max}}}{T_{i_{min}}}}.$

The lowest and the highest performance are achieved with $s_e^*$ and $s_p^*$, respectively. Since the processor speed has discrete range, they may not be configurable and need to be restricted to discrete values $s_e$ and $s_p$. Note that if $s_e^*$ is out of the range [0, 1], the task set is not feasible.

The operating speed $s$ is set in the range $[s_e,\ s_p]$ according to a selected strategy. If Energy saving mode and High performance mode are selected, it is set to $s_e$ and $s_p$, respectively. If User mode is selected, user sets the operating speed with no systematic policies for the selection. In this paper, we propose an algorithm for the selection of the operating speed to resolve the trade-off with a desired rate.

### 2.3. Elastic Algorithm

Let $U$ be the total CPU utilization of the task set and $U_d$ be the maximum total CPU utilization guaranteeing the schedulability. The elastic algorithm determines the task periods under the constraint $U = U_d$, described as follows:

$$^\forall\tau_i \in \Gamma_v \qquad U_i(s) = U_{i_{max}}(s) - \left(U_{v_{max}}(s) - U_d + U_f(s)\right)\frac{E_i}{E_v}, \quad (3)$$

where

$$U_{v_{max}}(s) = \sum_{\tau_i \in \Gamma_v} U_{i_{max}}(s), \quad (4)$$

$$U_f(s) = \sum_{\tau_i \in \Gamma_f} U_{i_{min}}(s), \quad (5)$$

$$E_v = \sum_{\tau_i \in \Gamma_v} E_i, \quad (6)$$

where $\Gamma_f$ and $\Gamma_v$ are the fixed task set and the variable task set, respectively. If task $\tau_i \in \Gamma_v$ is compressed less than its minimum CPU utilization by Eq. (3), it is fixed ($U_i(s) = U_{i_{min}}(s)$), and then we recompute the CPU utilization of the new variable task set. These steps are repeated until all tasks lie in the utilization range $[U_{i_{min}}(s),\ U_{i_{max}}(s)]$.

## 3. Optimization-based Resolution of the Trade-off

### 3.1. Power Consumption Model

In general, the power consumption in computing systems is modeled as

$$P(s) = K_3 s^3 + K_1 s + K_0, \quad (7)$$

where each term represents the power consumption of the subsystem in which both of the supply voltage and the processor speed are variable, only the processor speed is variable, and both of them are fixed, respectively[1]. All power coefficients $K_3, K_1, K_0$ are assumed to be nonnegative.

### 3.2. Compressing Force and Adequate Task Sets

We consider a force compressing tasks as performance degradation. From Eq. (3), for all tasks $\tau_i \in \Gamma_v$, we obtain

$$F(s)_{\Gamma_f,\Gamma_v} = \frac{U_{v_{max}}(s) - U_d + U_f(s)}{E_v} = \frac{U_{i_{max}}(s) - U_i(s)}{E_i}. \quad (8)$$

Since the force is applied to compress tasks in the variable task set, the increase of fixed tasks results in the degradation of the force. However, to control compressing rate corresponding to the elastic coefficients, we have to minimize the number of fixed tasks keeping the task set feasible. We define adequate sets as the sets satisfying this condition. We can obtain them by the elastic algorithm with the initial fixed state $\Gamma_{f0} = \emptyset$, and we have the following proposition.

**Prop. 1** *Let* $(\Gamma_f^*,\ \Gamma_v^*)$ *be the adequate sets for the operating speed s, then the following inequation holds.*

$$F(s)_{\Gamma_f^*,\Gamma_v^*} > F(s)_{\Gamma_f,\Gamma_v}\ ^\forall\Gamma_f \neq \Gamma_f^*. \quad (9)$$

Proof: Since the speed is fixed, we omit the notation of $s$. Since the adequate sets satisfy Eqs. (3)-(6), we have

$$^\forall\tau_i \in \Gamma_v^*\ U_i^* = U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*} \geq U_{i_{min}}, \quad (10)$$

$$^\forall\tau_i \in \Gamma_f^*\ U_i^* = U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*} < U_{i_{min}}. \quad (11)$$

Using the elastic algorithm, $U = U_d$ always holds. Then, for the pair $(\Gamma_f^*,\ \Gamma_v^*)$ and $(\Gamma_f,\ \Gamma_v)$, we have

$$U_d = \sum_{\tau_i \in \Gamma_v^*} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) + \sum_{\tau_i \in \Gamma_f^*} U_{i_{min}}, \quad (12)$$

$$U_d = \sum_{\tau_i \in \Gamma_v} \left(U_{i_{max}} - E_i F_{\Gamma_f,\Gamma_v}\right) + \sum_{\tau_i \in \Gamma_f} U_{i_{min}}. \quad (13)$$

The relations between the two pairs can be described as

$$\Gamma_f = \Gamma_f^* \backslash \Gamma_v \cup \Gamma_v^* \backslash \Gamma_v,\ \ \Gamma_v = \Gamma_f^* \backslash \Gamma_f \cup \Gamma_v^* \backslash \Gamma_f. \quad (14)$$

From Eqs. (10), (11), (12), (14), we obtain

$$\begin{aligned} U_d &= \sum_{\tau_i \in \Gamma_v^*} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) + \sum_{\tau_i \in \Gamma_f^*} U_{i_{min}} \\ &= \sum_{\tau_i \in \Gamma_v^* \backslash \Gamma_f} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) + \sum_{\tau_i \in \Gamma_v^* \backslash \Gamma_v} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) \\ &\quad + \sum_{\tau_i \in \Gamma_f^* \backslash \Gamma_f} U_{i_{min}} + \sum_{\tau_i \in \Gamma_f^* \backslash \Gamma_v} U_{i_{min}} \\ &> \sum_{\tau_i \in \Gamma_v^* \backslash \Gamma_f} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) + \sum_{\tau_i \in \Gamma_v^* \backslash \Gamma_v} U_{i_{min}} \\ &\quad + \sum_{\tau_i \in \Gamma_f^* \backslash \Gamma_f} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) + \sum_{\tau_i \in \Gamma_f^* \backslash \Gamma_v} U_{i_{min}} \\ &= \sum_{\tau_i \in \Gamma_v} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) + \sum_{\tau_i \in \Gamma_f} U_{i_{min}}. \end{aligned} \quad (15)$$

From Eqs. (13), (15), we have

$$\begin{aligned} \sum_{\tau_i \in \Gamma_v} \left(U_{i_{max}} - E_i F_{\Gamma_f,\Gamma_v}\right) &> \sum_{\tau_i \in \Gamma_v} \left(U_{i_{max}} - E_i F_{\Gamma_f^*,\Gamma_v^*}\right) \\ \Leftrightarrow F_{\Gamma_f,\Gamma_v} &< F_{\Gamma_f^*,\Gamma_v^*}. \end{aligned}$$

Thus, we have Eq. (9). □

### 3.3. Objective Function

We define an objective function as weighted sum of the power consumption and the force with a weight parameter $w$, which is the relative importance of the reducing power consumption. Note that $w = 0$ and $w = 1$ correspond to High performance mode and Energy saving mode, respectively. The objective function is described as follows:

$$W(s)_{\Gamma_f,\Gamma_v} = wP(s) + (1-w)\, k \cdot F(s)_{\Gamma_f,\Gamma_v}\ (0 \le w \le 1), \quad (16)$$

where $k$ is the scaling factor to adjust scales of the power consumption and the force. By setting the processor speed $s$ and the adequate sets $(\Gamma_f^*,\ \Gamma_v^*)$ to minimize the objective function, we can resolve the trade-off. When an optimized solution $(s,\ \Gamma_f^*,\ \Gamma_v^*)$ is obtained, task periods are calculated by Eq. (2). From Proposition 1, the objective function is maximized at the adequate sets if $s$ is fixed and it is regarded as a function with respect to $\Gamma_f$ and $\Gamma_v$. Moreover, since $P(s)$ and $F(s)_{\Gamma_f,\Gamma_v}$ are convex with respect to $s$, the objective function is also convex.

We define the threshold force of task $\tau_i$ as

$$F_{i_{max}}(s) = \frac{1}{E_i}\left(U_{i_{max}}(s) - U_{i_{min}}(s)\right). \quad (17)$$

It depends on five task parameters and leads to $U_i(s) = U_{i_{min}}(s)$. If the threshold force of task $\tau_i$ is less than the force compressing the variable task set, task $\tau_i$ has to belong to the fixed task set, otherwise $U_i(s) < U_{i_{min}}(s)$ holds and task $\tau_i$ violates its CPU utilization range.

In this paper, a scaling factor $k$ is given by

$$k = \frac{P(s_p) - P(s_e)}{\min_{i=1..n} F_{i_{max}}(s_e) - F(s_p)_{\Gamma_{fp},\Gamma_{vp}}}. \quad (18)$$

Although the compressing force clearly takes its maximum at the lowest speed $s_e$ and its adequate set $(\Gamma_{fe},\ \Gamma_{ve})$, it is complex to calculate. The threshold force can be calculated easier, and their relation is described as

$$F(s_e)_{\Gamma_{fe},\Gamma_{ve}} \le \min_{i=1..n} F_{i_{max}}(s_e), \quad (19)$$

where we have equality if and only if $s_e^*$ is equal to $s_e$. Therefore, we approximately treat the minimum of the threshold force as the maximum of the force in Eq. (18).

## 4. Algorithm and Suspensive Conditions

### 4.1. Algorithm

We propose the following algorithm to obtain the optimal solution which minimizes the objective function for a given weight parameter $w$. Let $s_{pr}$ be the next level higher than $s_n$, $U_{i_{pr}}$ and $U_i$ be the adequate CPU utilization of tasks for $s_{pr}$ and $s_n$, and $(\Gamma_{fpr},\Gamma_{vpr})$, $(\Gamma_{fn},\Gamma_{vn})$, and $(\Gamma_{fp},\Gamma_{vp})$ be the adequate sets for $s_{pr}$, $s_n$, and $s_p$, respectively. It searches the optimal solution decreasing the speed from the highest speed $s_p$, where STEP 2, 4, and 6 indicate the suspensive conditions. Note that, for calculating $W(s)_{\Gamma_f,\Gamma_v}$, we need to calculate the CPU utilizations for $(s,\Gamma_f,\Gamma_v)$ by Eqs. (3)-(6).

**STEP 1.** Calculate $U_{i_{pr}} = U_i(s_p)$ $(i = 1..n)$ and $W_{pr} = W(s_p)_{\Gamma_{fp},\Gamma_{vp}}$ by Eqs. (3)-(6), (8), (16), and set $s_{pr}$ and $s_n$ to $s_p$ and the next level lower than $s_p$, respectively.

**STEP 2.** If $s_n < s_e$ holds, go to STEP 8.

**STEP 3.** If $s_n \ge s_e$ holds, calculate $W = W(s_n)_{\Gamma_{fpr},\Gamma_{vpr}}$.

**STEP 4.** If $W \ge W_{pr}$ holds, go to STEP 8.

**STEP 5.** If $W < W_{pr}$ holds, calculate $W = W(s_n)_{\Gamma_{fn},\Gamma_{vn}}$.

**STEP 6.** If $W \ge W_{pr}$ holds, go to STEP 8.

**STEP 7.** If $W < W_{pr}$ holds, assign $s_{pr} = s_n$, $W_{pr} = W$, $U_{i_{pr}} = U_i$ $(i = 1..n)$ and decrease $s_n$, and go to STEP 2.

**STEP 8.** Using the optimal solution $(s_{pr},\ U_{i_{pr}}\ (i = 1..n))$, calculate the optimal task periods by Eq. (2).

### 4.2. Verification of Suspensive Conditions

In the proposed algorithm, we have three suspensive conditions: $s_n < s_e$, $W(s_n)_{\Gamma_{fpr},\Gamma_{vpr}} \ge W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}}$, and $W(s_n)_{\Gamma_{fn},\Gamma_{vn}} \ge W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}}$. The first condition means completion of comparing the objective functions for all speeds in the range $[s_e,\ s_p]$. We have the following proposition for the second and the third suspensive condition.

**Prop. 2** *If the second or the third suspensive condition holds, the optimal solution is* $(s_{pr},\ \Gamma_{fpr},\ \Gamma_{vpr})$*, that is,*

$$W(s)_{\Gamma_f^*,\Gamma_v^*} > W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}}\ (^\forall s < s_n). \quad (20)$$

Proof: We define $(\Gamma_f^*,\ \Gamma_v^*)$ as the adequate sets for $s$.

For the second suspensive condition, since $W(s)_{\Gamma_{fpr},\Gamma_{vpr}}$ is convex, we have

$$\begin{aligned} & W(s_n)_{\Gamma_{fpr},\Gamma_{vpr}} \ge W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}} \\ \Leftrightarrow\quad & W(s)_{\Gamma_{fpr},\Gamma_{vpr}} > W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}}\ (^\forall s < s_n). \end{aligned} \quad (21)$$

From Proposition 1, the objective function takes its maximum value at the adequate sets. Then we have

$$W(s)_{\Gamma_f^*,\Gamma_v^*} > W(s)_{\Gamma_{fpr},\Gamma_{vpr}}. \quad (22)$$

From Eqs. (21), (22), we obtain

$$W(s)_{\Gamma_f^*,\Gamma_v^*} > W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}}\ (^\forall s < s_n). \quad (23)$$

Thus, we have Eq. (20).

For the third suspensive condition, from Proposition 1 and the assumption, we have

$$W(s_n)_{\Gamma_{fn},\Gamma_{vn}} \ge W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}} > W(s_{pr})_{\Gamma_{fn},\Gamma_{vn}}. \quad (24)$$

Since $W(s)_{\Gamma_{fn},\Gamma_{vn}}$ is convex, we have

$$W(s)_{\Gamma_{fn},\Gamma_{vn}} > W(s_n)_{\Gamma_{fn},\Gamma_{vn}}\ (^\forall s < s_n). \quad (25)$$

For the adequate sets $(\Gamma_f^*,\ \Gamma_v^*)$, we have

$$W(s)_{\Gamma_f^*,\Gamma_v^*} > W(s)_{\Gamma_{fn},\Gamma_{vn}}. \quad (26)$$

From Eqs. (24), (25), (26), we obtain

$$W(s)_{\Gamma_f^*,\Gamma_v^*} > W(s_{pr})_{\Gamma_{fpr},\Gamma_{vpr}}\ (^\forall s < s_n). \quad (27)$$

Thus, we have Eq. (20). □

Table 1: Task set for the simulation.

| | $C_{i_{max}}$ | $\phi_i$ | $T_{i_{min}}$ | $T_{i_{max}}$ | $E_i$ |
|---|---|---|---|---|---|
| Task1 | 0.80 | 0.20 | 4.0 | 14.0 | 5.5 |
| Task2 | 0.80 | 0.70 | 4.0 | 14.0 | 5.5 |
| Task3 | 0.25 | 0.65 | 4.5 | 12.0 | 6.0 |
| Task4 | 0.90 | 0.80 | 7.0 | 15.0 | 0.5 |
| Task5 | 1.2 | 0.80 | 3.0 | 21.0 | 4.0 |

Table 2: The optimal task periods with respect to the operating speed $s$.

| | $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ |
|---|---|---|---|---|---|
| $s = 1.0$ | 4.48 | 4.48 | 7.79 | 7.11 | 3.12 |
| $s = 0.80$ | 6.10 | 5.77 | 12.0 | 7.31 | 3.36 |
| $s = 0.60$ | 14.0 | 9.08 | 12.0 | 7.57 | 3.72 |
| $s = 0.40$ | 14.0 | 14.0 | 12.0 | 8.69 | 6.01 |
| $s = 0.20$ | 14.0 | 14.0 | 12.0 | 14.3 | 21.0 |

## 5. Simulation Result

In this section, we present a simulation result performed to verify the proposed algorithm. We assume that the CPU frequency takes 10 discrete levels (0.15, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.0 [GHz]), the power consumption coefficients are set to $(K_3, K_1, K_0) = (15.3, 0, 0)$, the maximum total CPU utilization is set to $U_d = 0.90$, and the task set is given in Table 1. The speed selection range is calculated as $s_e = 0.20$, $s_p = 1.0$.

Shown in Fig. 1 is the power consumption and the adequate force with respect to the weight parameter $w$. They are piece-wise constant according to the discrete change of the optimal speed $s$. The lower $w$ becomes, the higher the processor speed becomes so that all tasks are executed with higher performances. On the other hand, the higher $w$ becomes, the lower the processor speed becomes so that the power consumption is larger reduced.

Shown in Table 2 is the relationship between the operating speed and the optimal task periods. Tasks 1 and 2 have the same task parameters except $\phi_i$. So, their periods equal at $s = 1.0$. Since $\phi_1 < \phi_2$, however, $C_1(s) < C_2(s)$ holds at $s < 1.0$. Then the period of Task 1 is larger than that of Task 2 at $s = 0.80$ and 0.60 and both tasks are fixed at $s = 0.40$ and 0.20. On the other hand, Task 5 has a wider interval of the CPU utilization $[U_{i_{min}}(s),\ U_{i_{max}}(s)]$ than Task 4 for all configurable speeds. But, $E_5$ is quite larger than $E_4$ so that the increase of the period of Task 5 is more rapid than that of Task 4. Thus, Task 5 is fixed at $s = 0.20$ while Task 4 is in the variable task set for any processor speed.

## 6. Conclusions

In this paper, we extended the DVS management with the elastic scheduling to resolve the trade-off with respect to the desired weight parameter $w$, by adopting the compressing force as the degradation of task performance. The proposed optimization-based algorithm can select the optimal speed for a given $w$ and obtain the optimal task periods according to the elastic coefficients. Its computational complexity is expressed as $O(mn^2)$, where $m$ and $n$ are number of the configurable CPU frequency levels and the tasks, respectively. Practically, it is considered as $O(n^2)$, since $m \leq 10$ for the current commercial CPU supported DVS technique. In future work, we extend the algorithm to dynamical scheduling, considering the usage of idle states generated by early completion of tasks.

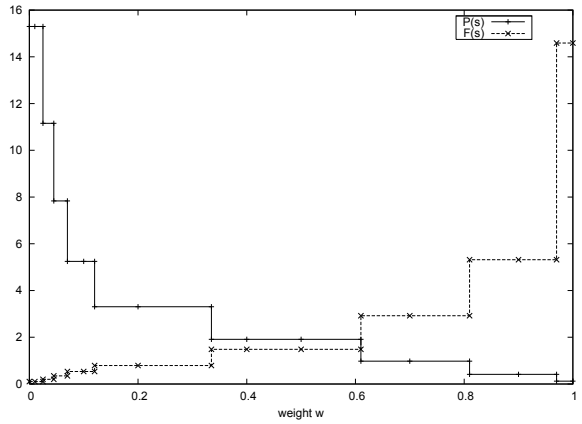


Figure 1: The optimal power consumption $P(s)$ and adequate force $F(s)$ with respect to $w$.

[**Acknowledgement**] This work was supported in part by a Grant-in-Aid for Scientific Research (B), No.17360918, from the Ministry of Education, Culture, Sports Science, and Technology of Japan.

### References

[1] T.L. Martin and D.P. Siewiorek: in Proceedings of the IEEE Transactions on VLSI Systems, Vol. 9, No. 1, pp. 29–34 (2001).

[2] G. Qu: in Proceedings of the IEEE/ACM ICCAD, pp. 560–563 (2001).

[3] J.W.S. Liu: "Real-Time Systems", Prentice Hall (2000).

[4] G. Buttazzo, L. Abeni and G. Lipari: in Proceedings of the 19th IEEE RTSS, pp. 286–295 (1998).

[5] G. Buttazzo and L. Abeni: Real-Time Systems, Vol. 23, No. 1–2 pp. 7–24 (2002).

[6] M. Marinoni and G. Buttazzo: in Proceedings of the 10th IEEE ETFA, pp. 307–313 (2005).

[7] M. Marinoni and G. Buttazzo: in Proceedings of the 12th IEEE International Conference on Embedded and RTCSA, pp. 294–304 (2006).