Real-time scheduling for optimal energy use

Bruno Gaujal, Nicolas Navet, Cormac Walsh et Eric Thierry

LIP, ENS-Lyon et Loria, Nancy

March 5,, 2004
Statement of the problem

Consider a dynamically variable voltage processor dedicated to execute jobs with real-time constraints:
\[
\{(a_n, s_n, d_n)\}_{n=1}^{N}
\]
where
- \( a_i \) is the arrival time,
- \( s_i \) is the size and
- \( d_i \) is the deadline (sorted as the \( a_i \))
of task number \( i \).

The objective is to choose at each time \( t \) the speed \( u(t) \) so that all tasks are executed within their deadline constraints and the total energy consumption is minimal.

This was solved by [Yao et Al]. We propose an alternative solution in the FIFO case which has two advantages:
it can be computed in linear time,
it provides a framework for several extensions and generalizations.
Motivation

1. This problem is relevant for all autonomous devices.
2. The problem looses all interest without deadline constraints.
3. The theoretical model used here gives good indications for many practical cases.
Mathematical model

Consider the cumulative work arrival function: 
$$A(t) = \sum_{i=1}^{N} s_i \cdot 1_{[a_i \leq t]}$$

and the cumulative work requirements: 
$$D(t) = \sum_{i=1}^{N} s_i \cdot 1_{[d_i \leq t]}$$

(F01) $A$ and $D$ are “staircase” functions (piecewise constant and non-decreasing).
(F02) $A(0) = D(0) = 0$.
(F03) Set $T = \max d_i$. Then, $t \geq T$; $A(t) = D(t) = D(T)$.
(F04) $\forall i, a_i < d_i$ implies that $\forall 0 \leq t \leq T$, $A(t) \leq D(t)$.

In the following, we will also consider arbitrary non-decreasing functions $A$ and $D$ (not necessarily related to a finite set of tasks) verifying (F02),(F03) and (F04).
Examples

Figure 1: Functions $A$ and $D$. 
Mathematical Program

The problem can be set into the form of a mathematical program:

**Problem 1**: find an integrable function $u : [0, T] \to \mathbb{R}$ such that

$$
\int_{0}^{T} g(u(s)) ds \text{ is minimized,}
$$

under the constraints

$$
u(t) \geq 0 \quad \forall t \in [0, T],
$$

$$
\int_{0}^{t} u(s) ds \leq A(t) \quad \forall t \in [0, T],
$$

$$
\int_{0}^{t} u(s) ds \geq D(t) \quad \forall t \in [0, T].
$$
Power consumption

The quantity $g(u(t))$ is the instantaneous power consumption by the processor at time $t$, $P(t)$. We assume that $g$ is a convex and increasing function. Depending on the technology, the function $g$ can be a polynomial of degree 2 or 3 or more. Some models also use an exponential function.

Our study is rather independent of the technology. For instance (from [Gruian]) in CMOS circuits, $P \approx \alpha f CV_{dd}^2$ and $f \approx V_{dd}/(V_{dd} - V_t)^\gamma$ so that,

$$P \approx \alpha C f^{3/\gamma}$$

In the following, we will show that the actual form of $P$ (or $g$) is not important. The solution is the same for all convex and increasing functions to minimize the total energy consumption: $\int_0^T g(u(t))dt = \int_0^T P(t)dt$
Scheduling policy

In this talk we focus on the problem of determining the speed of the processor $u(t)$.

However, there is another question that must be addressed: which task should be executed at time $t$?

In the following, we implicitly consider that the processor uses the EDF policy: the active task with the earliest deadline at time $t$ is being executed by the processor.

Note that here, the EDF policy is the same as the FIFO policy.

This scheduling policy is known to be optimal (in terms of deadlines).
First results on Problem 1

Result 1 If the function $g$ is strictly convex, then the optimal solution of Problem 1 is unique (up to a set of measure 0).

Result 2 If $u^*$ is the optimal solution of Problem 1 where $g$ is strictly convex and non-decreasing, then $u^*$ is also an optimal solution of Problem 1 for any other non-decreasing convex function.

Consider the convex function $\sqrt{1 + u^2}$. The integral is the length of the curve $u$. This means that $U^* = \int_0^t u^*$ is the function with the shortest length between $A$ and $D$. 
Some examples

Figure 2: The function $U^*$ is the shortest path from point 0 to $T$. 
Computational Issues

Result 4 The function $U^*$ can be computed in linear time $O(N)$, (in the number of tasks) if they are sorted, and $O(N \log N)$ otherwise. This complexity is optimal if one believes that $O(N \log N)$ is the optimal arithmetic complexity for sorting.

The algorithm is similar to the one computing the convex hull of a set of points in the plane by alternating between the concave envelope of $D$ and the convex envelope of $A$.

Corollary This provides a test for feasibility in linear time for FIFO tasks.
Example

Figure 3: construction of $u^*$
Possible extensions (I)

Problem 2: find an integrable function $u : [0, T] \to \mathbb{R}_+$ such that

$$\int_0^T g(u(s)) \, ds$$

is minimized, \hspace{1cm} (5)

under the constraints (2), (3) and (4) and the additional constraint

$$u(t) \leq v_{max} \hspace{0.5cm} \forall t \in [0, T],$$

$$\hspace{1cm} \text{(6)}$$

Imposing this constraint on the instantaneous speed models the fact that the processor has a maximum speed $v_{max}$. 
Results on Problem 2

Feasibility: The set of jobs is feasible if it is possible to guarantee all the deadlines using the maximal speed at all times.

Result 3 The set of tasks is feasible if and only if Problem 1 and Problem 2 are equivalent.

The proof is based in the fact that $\sup u^*(t) \leq \sup u(t)$ for all admissible function $u$ (verifying constraints (2),(3) and (4)).
Possible extensions (II)

Problem 3 Find an integrable function $z : [0, T] \rightarrow \mathbb{R}_+$ such that

$$\int_0^T g(z(s)) ds$$

is minimized,

under Constraints (3) and (4) and the additional constraint

$$z(t) \in \{v_1, \cdots, v_\ell\}.$$  \hspace{1cm} (7)

Here $v_1 \leq \cdots \leq v_\ell$ is the finite set of admissible speeds for the processor. Nowadays technology for processors with varying speeds only permits to select among a finite set of frequencies. Problem 3 is somehow more realistic, especially in the case of staircase functions.
If the processor can only use a finite number of frequencies, the optimal solution $z^*$ can be computed using the following procedure:

1. Partition $[0, T]$ into intervals $I_1 \cdots I_M$ defined by the discontinuities of $A$ and $D$.
2. On each $I_i$, $u^*$ is constant equal to $u_i$.
3. On each $I_i$, replace $u_i$ by the two neighboring speeds.
4. This gives an optimal solution for Problem 3.

**Result 4** The computation time of $z^*$ given $u^*$ is linear in the number of tasks.
Figure 4: \( Z^* \) is the integral of an optimal solution \( z^* \) using speeds \( v_1, v_2 \) and \( v_3 \).
Minimal number of speed changes

Unlike for Problems 1 and 2, Problem 3 admits many optimal solutions. It is possible to find an optimal solution with fewer speed changes than $z^*$.

The main two ideas of the construction:
1) Switch between speeds only when absolutely necessary.
2) Use a Viterbi algorithm to glue the segments optimally.
Figure 5: An optimal solution with a minimal number of speed changes.
Extension 1: fluid tasks

We generalize the problem for arbitrary functions $A$ and $D$, without assuming that they are staircase functions. This models infinitesimal tasks by a fluid arrival process.

$A$ and $D$ satisfy the following properties:

$(F_1)$ $A$ is a non-decreasing left-continuous function, with right and left derivatives in $\mathbb{R} \cup \{-\infty, +\infty\}$ and $A(0) = 0$.

$(F_2)$ $D$ is a non-decreasing right-continuous function, with right and left derivatives in $\mathbb{R} \cup \{-\infty, +\infty\}$, $D(0) = 0$, and $D \geq A$.

**Result 5** The shortest path between $A$ and $D$ is the optimal solution of Problem 1.
Example

Figure 6: The optimal solution with arbitrary function $A$ and $D$. 

Epfl seminar
Finite number of speeds

Additional technical assumptions are needed:

\[(F_3) \ \exists \delta > 0, \ \forall 0 \leq a \leq b \leq T, \ \int_a^b A(s) - D(s)ds \geq \delta(b - a).\]

\[(F_4)\text{ there exists a finite number of points } x \text{ between } 0 \text{ and } T \text{ such that } A(x) = D(x), \text{ and } \forall x \text{ s.t. } A(x) = D(x), \exists v, w \in \{v_1, \cdots v_\ell\} \text{ s.t. } \frac{dA}{dt_+}(x) \geq v, \frac{dD}{dt_+}(x) \leq v, \text{ and } \frac{dA}{dt_-}(x) \leq w, \frac{dD}{dt_-}(x) \geq w.\]

Construct the function \(y_n\) in intervals of length \(T/n\). In each such interval, the function \(y_n\) is composed of the two speeds neighboring the average speed of \(U^*\) in the interval.

**Result 6** The function \(y^* = y_n\) is an optimal solution of Problem 3 for the smallest \(n\) such that \(y_n\) is admissible.
Extension 2: non-convex costs

Consider the case where the function $g$, which gives the instantaneous energy consumption is not convex and increasing. This is typically the case when the static power dissipated by the processor is not negligible. In this case, the typical behavior of $g$ is displayed in Figure 7.

Figure 7: Example of a non-convex energy consumption function $g$, with its convex hull $h$, and the sets $C$ and $B$. 
We assume that the function $g$ is semi-continuous but not convex and increasing. For technical reasons, we will further assume that $g$ has a finite number of inflexion points.

1. Construct the convex hull $h$ of $g$.

2. Solve Problem 1 using $h$ instead of $g$, as the instantaneous cost. Since $h$ is convex and increasing, we get as before the shortest path $U^*$ between $A$ and $D$.

3. Construct a set of functions, $v_n, n \in \mathbb{N}$ as follows. If $u^*(t) \in C$, then $v_n(t) = u^*(t)$. If $u^*(t) \in B$, then $v_n(t)$ is made of $n$ segments with slopes being the value of $g$ at the end points of the interval.

Result 7 The function $v^* = v_n$ is the optimal solution to Problem 1 for the smallest $n$ such that $v_n$ is admissible.
Finite number of speeds

If $g$ is not convex and if the number of speeds available is finite $\{v_1 \cdots , v_\ell\}$, then the problem becomes easier:

1. Construct the convex hull $h$ of $g$.

2. Define a new set of speeds by removing from $\{v_1 \cdots , v_\ell\}$ all the speeds such that $g(v_i) \neq h(v_i)$.

3. Solve the problem as before with $h$ instead of $g$ and the new set of speeds.

This gives the optimal solution for Problem 3.
Extension 3: non FIFO tasks

The previous method no longer works here. The constraints 3 and 4 do not guarantee the real time constraints of all tasks.

We can still use a similar method based on the decomposition of the set of tasks in "levels".

The complexity increases sensibly.
The Hasse diagram

We order the tasks using strict inclusion and we build the associated Hass diagram.
Construction of the speed function

For all tasks with level not smaller than \( k \) in the Hasse diagram, we construct two functions \( A_k(t) \) and \( D_k(t) \) over the interval \([0, T]\) using the following definition:

\[
A_k(t) = \sum_{L(i) \geq k} s_i \cdot 1_{[a_i < t]}, \quad D_k(t) = \sum_{L(i) \geq k} s_i \cdot 1_{[d_i \leq t]}.
\]

Program \( P_k \): Find an integrable function \( u_k^* : [0, T] \to \mathbb{R} \) such that

\[
\int_0^T g(u_k^*(s)) ds \text{ is minimized,}
\]

under the constraints

\[
\begin{align*}
& u_k^*(t) \geq 0 \quad \forall t \in [0, T], \\
& \int_0^t u_k^*(s) ds \leq A_k(t) \quad \forall t \in [0, T], \\
& \int_0^t u_k^*(s) ds \geq D_k(t) \quad \forall t \in [0, T].
\end{align*}
\]
Construction of the speed function(II)

1. For all $k \in \{1 \cdots , K\}$
   
   (a) Construct $u_k^*(t)$ the optimal solution of $P_k$.
   
   (b) Define $r_k := \sup_{0 \leq t \leq T} u_k^*(t)$.

2. Compress the time interval over which $\sup_{0 \leq t \leq T} u_k^*(t) = r_k$.

3. Restart at step 1.
Average complexity

The complexity of this algorithm is $O(KN^2)$.

For random sets of tasks, $K \leq \sqrt{N}$ by the theorem on the largest increasing subsequence.

In our particular case, $K$ should be much smaller than that. For example, if the arrivals are Poisson and the relative deadlines are exponentially distributed, then $K \leq \log N$. 
Extension 4: optimal choice of the speeds

Through statistical analysis of the load over time, one can derive the distribution of the frequencies that the processor tends to use more often. This knowledge is given under the form of a measure $\mu$ over the interval $[0, v_{\text{max}}]$.

The problem now is to find the best selection of the set of speeds $\{v_1, \ldots, v_\ell\}$ so as to minimize the expected energy consumption:

$$
\min_{0 \leq v_1 \leq \cdots \leq v_\ell \leq 1} s(v_1 \cdots v_\ell) \quad \text{where} \quad s(v_1 \cdots v_\ell) = \sum_{i=0}^{\ell} \sigma_i,
$$

with

$$
\sigma_i = \frac{1}{v_{i+1} - v_i} \left( (-g(v_{i+1})v_i + g(v_i)v_{i+1}) \int_{v_i}^{v_{i+1}} d\mu + (g(v_{i+1}) - g(v_i)) \int_{v_i}^{v_{i+1}} v d\mu \right) \nonumber
$$
Dynamic Programming Solution

Let $S^*(\ell, x, y)$ be the value of the optimal energy consumption with $\ell$ speeds distributed between $x$ and $y$. Using this definition,

$$S^*(\ell, 0, 1) = \min_{0 \leq v_1 \cdots \leq v_\ell \leq 1} s(v_1 \cdots v_\ell).$$

It is easy to see that $S^*$ satisfies the following recurrence equation:

$$S^*(\ell, x, y) = \min_{x \leq z \leq y} (S(0, x, z) + S^*(\ell - 1, z, y)).$$
Let us discretize the interval $[0, 1]$ with steps of length $1/k$ where $k$ is large enough (the value of $k$ is given in the following).

Once this is done, we can reformulate the recurrence equation using integer variables $x$ and $y$ between 0 and $k$.

$$S_k^*(\ell, x, y) = \min_{z=x}^y (S_k^*(0, x, z) + S_k^*(\ell - 1, z, y)),$$

where

$$S_k^*(0, x, z) = S(0, \frac{x}{k}, \frac{z}{k}).$$

This is solvable using a dynamic programming approach. The arithmetic complexity of the algorithm is $O(k^2\ell)$ operations with $O(k\ell)$ memory space.

Since $g$ is convex, the slope of $g$ is always bounded by $p := \max(g'_+(0), g'_-(1))$.

**Result 8** If $k > \frac{p}{\varepsilon}$ then $S_k^*(\ell, 0, 1) - S^*(\ell, 0, 1) \leq \varepsilon$. 
The uniform case

The uniform case corresponds to the case where one has no statistical information on the future use of the processor so that all speeds between 0 and 1 seem equally likely to be used. This case admits a solution in closer form. It is a solution of the system:

\[ g'(v_i) = \frac{g(v_{i+1}) - g(v_{i-1})}{v_{i+1} - v_{i-1}}. \] (8)
Example

Figure 8: The choice of speed $v_i$, once $v_{i-1}$ and $v_{i+1}$ are given
If we consider the classical case where the energy is quadratic, \( g(x) = ax^2 \) \((a > 0)\), then the system of equations (??) becomes:

\[
\begin{align*}
    v_1 & = 0 \\
    2v_2 & = v_3 \\
    2v_3 & = v_2 + v_4 \\
    \vdots & = \vdots \\
    2v_{\ell-1} & = v_{\ell-2} + v_{\ell} \\
    v_\ell & = 1
\end{align*}
\]

Which has a unique solution \((0, \frac{1}{\ell-1}, \cdots, \frac{\ell-2}{\ell-1}, 1)\). This means that if the energy is quadratic in the frequency, then the speeds should be evenly distributed over \([0, 1]\).
Some open problems

Another way to define the speed function is through the following mathematical program.

Program $P_k$: Find an integrable function $u_k^* : [0, T] \to \mathbb{R}$ such that

$$
\int_0^T g(u_k^*(s)) ds \text{ is minimized,} \tag{9}
$$

under the constraints

$$
      u_k^*(t) \geq u_{k+1}^*(t) \quad \forall t \in [0, T], \tag{10}
$$

$$
\int_0^t u_k^*(s) ds \leq A_k(t) \quad \forall t \in [0, T], \tag{11}
$$

$$
\int_0^t u_k^*(s) ds \geq D_k(t) \quad \forall t \in [0, T]. \tag{12}
$$
Another open question is to take into account the overhead in speed changes in the synchronous as well as the asynchronous case.