Minimising the Energy Consumption of Real-Time Tasks with Precedence Constraints on A Single Processor

Hui Wu and Sridevan Parameswaran

School of Computer Science and Engineering
The University of New South Wales
{huiw, sridevan}@cse.unsw.edu.au

Abstract. Energy-aware task scheduling is critical for real-time embedded systems. Although dynamic power has traditionally been a primary source of processor power consumption, leakage power is becoming increasingly important. In this paper, we present two optimal energy-aware polynomial-time algorithms for scheduling a set of tasks with release times, deadlines and precedence constraints on a single processor with continuous voltages. Our algorithms are guaranteed to minimise the total energy consumption of all tasks while minimising their maximum lateness under two power models: the dynamic power model where the dynamic power dominates the processor power consumption and the dynamic and leakage power model where both dynamic power and leakage power are significant sources of the processor power consumption. The time complexities of both algorithms are $O(n^3)$, where n is the number of tasks.

1 Introduction

In mobile real-time embedded systems, energy is a precious resource. Minimising the energy consumption while satisfying the performance constraints is a key issue in the design of such systems. Energy reduction is also important in other real-time embedded systems. A real-time system with less energy consumption generates less heat and therefore has a longer lifetime.

Traditionally, dynamic power is the main source of processor power consumption. There are two techniques, namely DVS (Dynamic Voltage Scaling) and DPM (Dynamic Power Management) that can be used to reduce the dynamic power consumption of processors. In DVS, different tasks are run at different voltages and clock frequencies to fill up the idle periods in the schedule, while still satisfying the performance constraints. DPM aims to shut down system components not currently in use. DVS is more efficient than DPM in reducing the energy consumption of processors. DPM is used only if DVS is not applicable.

A lot of work has been done in DVS for real-time embedded systems. In the case of a single processor, many algorithms and heuristics have been proposed. Yao et al [1] studied the problem of scheduling a set of independent tasks

with individual release times and deadlines on a single processor with continuous voltage such that the energy consumption of the processor is minimised. They proposed an optimal EDF-based (Earliest Deadline First) algorithm for static scheduling and suboptimal algorithms for online scheduling. Kwon and Kim [16] proposed a static scheduling algorithm for the same problem with discrete voltages. Li and Yao [2] proposed a faster algorithm for the same problem solved by Kwon and Kim.

Ishihara and Yasuura [3] proposed a model of dynamically variable voltage processor and a static voltage scheduling algorithm formulated as an integer linear programming problem. In addition to task execution times and deadlines, their algorithm also requires the average switched capacitance for each task. However, their algorithm does not consider precedence constraints and release times.

Hong et al [4] proposed an energy-aware on-line algorithm for scheduling both periodic tasks and sporadic tasks. Their algorithm guarantees the deadlines of all periodic tasks and tries to maximise the number of aperiodic tasks that can be finished by their deadlines.

Quan et al [6] studied the problem of minimising the total energy consumption of a set of tasks with individual release time and deadline on a single processor using fixed priority scheduling. They proposed a heuristic for this problem. Yun and Kim [15] also studied the same problem. They proved the NP-hardness of the problem and proposed an approximation algorithm.

Sinha and Chandrakasan [5] proposed an energy-aware SEDF (Slack Earliest Deadline First) algorithm for scheduling a set of independent tasks with release times and deadlines on a single processor. Their algorithm is stochastically optimal in minimising processor energy consumption and maximum lateness.

Shin et al [9] proposed an intra-task scheduling approach that can further reduce energy consumption of tasks by partitioning a task into several segments, each assigned with a separate a voltage.

As technology feature size continues to scale, leakage power is increasing and will limit power savings obtained by DVS alone. Therefore, the optimisation objective of task scheduling is to minimise the sum of dynamic energy and leakage energy. Recently, a number of researchers studied the problem of combining DVS and adaptive body biasing (ABB) to simultaneously optimise both dynamic energy consumption and leakage energy consumption.

Quan et al [8] proposed a scheduling technique that can effectively reduce the overall energy consumption for hard real-time systems scheduled according to a fixed priority scheme. Experimental results show that a processor using their strategy consumes as less as 15 percent of the idle energy of a processor employing the conventional strategy.

Andrei et al [12] investigated the problem of overhead-aware voltage selection for dynamic and leakage energy reduction of time-constrained systems where tasks are subject to deadline and precedence constraints. They optimally solved the continuous voltage selection problem by using non-linear programming and proved NP-hardness in the discrete case.

Jejurikar et al [13] proposed a leakage-aware algorithm for scheduling periodic tasks on a single processor. Their algorithm uses EDF scheduling strategy. Unlike our algorithms, their algorithm does not consider critical task sets. Instead, it computes the execution speed of each task individually. Jejurikar and Gupta [14] proposed leakage-aware algorithms for fixed-priority systems. Both algorithms are not guaranteed to minimise the total energy consumption of all tasks.

In this paper, we propose polynomial-time task scheduling algorithms for minimising the total energy consumption of a set of real-time tasks with individual release times, deadlines and precedence constraints on a single processor with continuous voltages. Assuming that the voltage transition overheads are negligible, our algorithms are guaranteed to minimise the total energy consumption of all tasks under two power models: the dynamic power model where the dynamic power dominates the processor power consumption and the dynamic power and leakage power model where both dynamic power and leakage power are significant sources of the processor power consumption. In addition, our algorithms are guaranteed to minimise the maximum lateness of all tasks.

We make the following significant contributions.

1. Under the dynamic power and leakage power model our algorithm is the first algorithm for minimising the total energy consumption of a set of tasks with release times, deadlines and precedence constraints on a single processor. The previous algorithm proposed by Jejurikar et al [13] is not guaranteed to minimise the total energy consumption of all tasks and does not consider the precedence constraints. [1]
2. Under the dynamic power model our algorithm generalises Yao's algorithm [1] by considering additional precedence constraints.

2 Power Models and Definitions

2.1 Dynamic Power Model

Under the dynamic power model, the processor power is dominated by the dynamic power [17] which is given by:

$$P_{dynamic} = C_{eff} V_{dd}^2 f \tag{1}$$

where C_{eff} is the effective switching capacitance, V_{dd} is the supply voltage and f is processor clock frequency.

Processor clock frequency f, is almost linearly related to the supply voltage:

$$f = \frac{(V_{dd} - V_{th})^{\alpha}}{k V_{dd}} \tag{2}$$

[1] The algorithm proposed by Jejurikar et al considers periodic tasks without any precedence constraint and uses EDF scheduling strategy. Since a periodic task can be represented by a set of non-periodic tasks with individual release times and deadlines, the problem they studied is a special case of our problem where no precedence constraint exists.

where k is a constant, V_{th} is the threshold voltage and $1 < \alpha \leq 2$. Based on the SPICE simulation of the Berkley predictive models for a $0.07\mu m$ process [18], the threshold voltage [10] is given by the following equation:

$$V_{th} = V_{th1} - k_1 V_{dd} - k_2 V_{bs} \tag{3}$$

where V_{th1}, k_1 and k_2 are constants and V_{bs} is body bias voltage.
Substitute (3) into (2) gives the expression of f in terms of V_{dd} and V_{bs}.

$$f = \frac{((1+k_1)V_{dd} + k_2 V_{bs} - V_{th1})^\alpha}{kV_{dd}} \tag{4}$$

Substituting (4) into (1), we have the dynamic power function $P(V_{dd})$ where the dynamic voltage V_{dd} is the only variable:

$$P(V_{dd}) = \frac{C_{eff}}{k} V_{dd}((1+k_1)V_{dd} + k_2 V_{bs} - V_{th1})^\alpha \tag{5}$$

Note that $P(V_{dd})$ is a convex function.

2.2 Dynamic and Leakage Power Model

Under dynamic power model, the processor power dissipation is dominated by both dynamic power and leakage power [17], where the leakage power [11] can represented as:

$$P_{leakage} = I_s(\frac{W}{L})V_{dd}e^{\frac{-V_{th}}{nV_T}} + |V_{bs}|(I_j + I_b) \tag{6}$$

where I_s and n are technology parameters, W and L are device geometries, I_j is drain-body junction leakage current, I_b is source-body junction leakage current, and V_T is the thermal voltage.

Substituting (3) into (6), we have:

$$P_{leakage} = k_3 V_{dd} e^{k_4 V_{dd} + k_5 V_{bs}} + |V_{bs}|(I_j + I_b) \tag{7}$$

where k_3, k_4 and k_5 are new constants. Therefore, the total power consumption can be represented by:

$$\begin{aligned} P &= P_{dynamic} + P_{leakage} \\ &= C_{eff} V_{dd}^2 f + k_3 V_{dd} e^{k_4 V_{dd} + k_5 V_{bs}} + |V_{bs}|(I_j + I_b) \end{aligned} \tag{8}$$

From Equation (4), we have

$$V_{bs} = k_6 V_{dd} + k_7 (f V_{dd})^{\frac{1}{\alpha}} + k_8 \tag{9}$$

where k_6, k_7 and k_8 are new constants.

Substituting (9) into (8), we have the dynamic and leakage power function $P(V_{dd}, f)$ where V_{dd} and f are the only two variables:

$$\begin{aligned} P(V_{dd}, f) =& C_{eff} V_{dd}^2 f + k_9 V_{dd} e^{k_{10} V_{dd} + k_{11} (f V_{dd})^{\frac{1}{\alpha}}} \\ &+ k_{12} V_{dd} + k_{13} (f V_{dd})^{\frac{1}{\alpha}} + k_{14} \end{aligned} \tag{10}$$

where $k_9 - k_{14}$ are new constants. Note that $P(V_{dd}, f)$ is a convex function.

2.3 Problem and Definitions

In this section, we propose two optimal algorithms for energy-aware real-time task scheduling on a single processor. We assume two power models: the dynamic power model and the dynamic and leakage power model. Under the dynamic power model, the dynamic power is the main source of the processor power consumption and the leakage power is negligible. Under the dynamic and leakage power model, both dynamic power and leakage power are significant sources of the processor power consumption.

A problem instance P consists of a set $V = \{T_1, T_2, \cdots, T_n\}$ of n tasks with the following constraints:

1. Worst case execution times. The worst case execution time of task T_i, denoted by c_i, is the longest execution time of T_i when the processor runs at the maximum frequency.
2. Release times. Each task T_i has a pre-assigned release time r_i.
3. Deadlines. Each task T_i has a pre-assigned deadline d_i.
4. Precedence constraints represented by a DAG (Directed Acyclic Graph) $G = (V, E)$, where $E = \{(T_i, T_j) : T_i \text{ precedes } T_j\}$.
5. A single processor with continuous voltages.

The energy-aware single processor scheduling problem is described as follows. Given a problem instance P, find a valid schedule with minimum lateness for all the tasks such that the total energy consumption of all tasks is minimised. A schedule is called a valid schedule if it satisfies all release times and precedence constraints.

Definition 1. *Given a schedule σ for a problem instance P and a task T_i, the lateness of T_i is $f_i - d_i$, where f_i is the completion time of T_i in σ.*

Definition 2. *Given a problem instance P and a task T_i, the edge-consistent release time of T_i, denoted by r'_i, is recursively defined as follows: $r'_i = max\{r_i, max\{ r'_j + c_j : T_j \text{ is an immediate predecessor of } T_i\}\}$*

Definition 3. *Given a problem instance P and a task T_i, the edge-consistent deadline of T_i, denoted by d'_i, is recursively defined as follows: $d'_i = min\{d_i, min\{ d'_j - c_j : T_j \text{ is an immediate successor of } T_i\}\}$*

Definition 4. *Given a set S of tasks with individual release times and deadlines and a partial schedule for S on a single processor, a time interval $[a, b]$ is a forbidden interval if it is fully occupied by one or more tasks in the partial schedule. A time interval $[a, b]$ is a maximum forbidden interval if it is a forbidden interval and there exists d such that neither $[a-d, a]$ nor $[b, b+d]$ is a forbidden interval.*

Since all forbidden time intervals are occupied by the tasks already scheduled, they cannot be used by other tasks.

Definition 5. *Given a set S' of independent tasks with individual release times and deadlines, a partial schedule σ for S' and a set S of unscheduled tasks, the processor utilisation of S is defined to be:*

$$U(S) = \frac{\sum_{T_j \in S} c_j}{d_{max} - r_{min} - l(S)}$$

where r_{min} is the minimum release time of all tasks, d_{max} is the maximum deadline of all tasks, and $l(S)$ is the total length of all maximum forbidden intervals within $[r_{min}, d_{max}]$.

Definition 6. *Given a set S' of independent tasks with individual release times and deadlines, a partial schedule σ for S', a subset S of unscheduled tasks is a critical subset if S has the greatest processor utilisation among all subsets of unscheduled tasks.*

Definition 7. *Given a set S of tasks with individual release times and deadlines, the interval of S is $[r_{min}, d_{max}]$, where r_{min} and d_{max} are the minimum release time and the maximum deadline of all tasks, respectively.*

3 Optimal Scheduling Algorithm

3.1 Optimal Task Execution Speed

Our scheduling algorithms are underpinned by critical task subsets. If dynamic voltage and body bias voltage take continuous values, we can show that in an optimal schedule all the tasks in a critical task subset must have the same speed. We call this speed optimal speed. The optimal speed is dependent on the power model. Next we show how to compute the optimal speed $f_{opt}(S')$ for all tasks in a critical task subset $S' = \{T_{i_1}, T_{i_2}, \cdots, T_{i_k}\}$.

Under the power model where the dynamic power is the main source of processor power consumption, $f_{opt}(S')$ is the slowest speed at which the interval of S' is fully occupied by all tasks in S'. Specifically, $f_{opt}(S')$ is equal to $U(S')f_{max}$, where f_{max} is the maximum frequency of the processor. After assigning the optimal speed to all tasks in the critical task set, the interval of S' becomes a forbidden interval.

Under the power model where both dynamic power and leakage power are the significant sources of the processor power consumption, the optimal speed

may not be the slowest speed due to the leakage power. Let F_{opt} be the optimal processor frequency which minimises the total power $P(V_{dd}, f)$ of the processor. F_{opt} can be computed by using gradient search [20]. Given a processor, F_{opt} is a constant. When computing the optimal speed $f_{opt}(S')$ for all tasks in S', we need to consider the following two cases:

1. $U(S')f_{max} < F_{opt}$. Since the power function $P(V_{dd}, f)$ is a convex function and F_{opt} is the speed which minimises $P(V_{dd}, f)$, $f_{opt}(S')$ is equal to F_{opt}.
2. $U(S')f_{max} \geq F_{opt}$. In this case, by the convexity of the power function $P(V_{dd}, f)$, $P(V_{dd}, f)$ decreases as f increases within $[f_{min}, F_{opt}]$, where f_{min} is the minimum processor frequency. As a result, $f_{opt}(S')$ is equal to $U(S')f_{max}$.

In the case where $U(S')f_{max} < F_{opt}$ holds, the interval of S' will contain non-forbidden intervals after all tasks in S' are scheduled. This is because the interval of S' cannot be fully used by all tasks in S'. In the case where $U(S')f_{max} \geq F_{opt}$ holds, since all tasks in S' will fully use the interval, the interval of S' will become a forbidden interval after all tasks in S' are scheduled.

3.2 Minimum Energy Scheduling Algorithms

Our scheduling algorithms consist of the following two main steps.

1. Transform the original problem instance P into a precedence-free scheduling problem instance P' as follows.
   (a) Compute the edge-consistent release time for each task and set its release time to its edge-consistent release time.
   (b) Compute the edge-consistent deadline for each task and set its deadline to its edge-consistent deadline.
   (c) Compute a schedule σ for the original problem instance P on the single processor running at the maximum frequency by using EDF strategy.
   (d) Create a precedence-free problem instance P' as follows:
      – For each task T_i, set its release time to its start time in σ and its deadline to $\max\{d_i, e_i\}$, where e_i is the finish time of T_i in σ.
2. Find a minimum energy schedule for the precedence-free problem instance P' as follows. Let V be a set of all tasks in P. Repeat the following steps until V is empty.
   (a) Find a critical task subset S' of V.
   (b) Compute the optimal speed for all tasks in S' as described in the previous subsection.
   (c) Compute a partial schedule for S' using EDF strategy and the optimal speed.
   (d) $V = V - S'$.

The minimum energy schedule for P' is a union of partial schedules for all critical task subsets. As we will prove in the next section, it is also a minimum energy and minimum lateness schedule for the original problem instance P.

Under the dynamic power model, we can use Yao's algorithm [1] to compute

all critical task subsets. Under the dynamic and leakage power model, the interval of a critical task subset may not be fully used. Therefore, the whole interval cannot be removed. As a result, Yao's algorithm is not applicable to the dynamic and leakage power model.

Next, we describe an efficient implementation of our algorithm for computing critical task sets under the dynamic and leakage power model. Given a problem instance P which consists of a set of independent tasks with individual release times and deadlines and a single processor, the critical task subsets are computed as follows:

1. Find a critical task subset S for P as in [1].
2. Compute the optimal speed for S.
3. Compute a partial schedule σ for S using EDF strategy.
4. Find all maximum forbidden intervals in $[r_{min}, d_{max}]$, where r_{min} and d_{max} are the minimum release time and the maximum deadline, respectively, of all tasks in S. Let $[a_1, b_1]$, $[a_2, b_2]$, $\cdots$, $[a_k, b_k]$ be the k maximum forbidden intervals in $[r_{min}, d_{max}]$ with $a_i < b_i$ $(i = 1, 2, \cdots, k)$ and $b_i < a_{i+1}$ $(i = 1, 2, \cdots, k-1)$.
5. Remove all tasks in S from P.
6. Modify the release time of each task T_i in P as follows:
   (a) If $r_i \geq d_{max}$, set the new release time of T_i to $r_i - \sum_{j=1}^{k} (b_j - a_j)$.
   (b) If $r_i \geq r_{min}$ and $r_i < d_{max}$, there are two cases.
      i. Case 1: r_i is within a maximum forbidden interval $[a_s, b_s]$. Set the new release time of T_i to $r_i - \sum_{j=1}^{s} (b_j - a_j)$.
      ii. Case 2: r_i is not within any maximum forbidden interval in $[r_{min}, d_{max}]$. Let $[a_t, b_t]$ be the maximum forbidden interval such that t is the largest integer satisfying $b_t < r_i$. Set the release time of T_i to $r_i - \sum_{j=1}^{t} (b_j - a_j)$.
7. Modify the deadline of each task T_i in P as follows:
   (a) If $d_i \geq d_{max}$, set the new deadline of T_i to $d_i - \sum_{j=1}^{k} (b_j - a_j)$.
   (b) If $d_i \geq r_{min}$ and $d_i < d_{max}$, there are two cases.
      i. Case 1: d_i is within a maximum forbidden interval $[a_s, b_s]$. Set the new deadline of T_i to $a_s - \sum_{j=1}^{s-1} (b_j - a_j)$.
      ii. Case 2: r_i is not within any maximum forbidden interval in $[r_{min}, d_{max}]$. Let $[a_t, b_t]$ be the maximum forbidden interval such that t is the largest integer satisfying $b_t < d_i$ and set the new deadline of T_i to $d_i - \sum_{j=1}^{t} (b_j - a_j)$.
8. Repeat the above steps until there is no task in P.

4 Optimality Proof and Time Complexity Analysis

We will use the following property of convex functions in our optimality proof.

Lemma 1. *Given a convex function f(X) in the n−dimensional space, the following inequality holds:*

$$\sum_{i=1}^{m} \lambda_i f(X_i) \geq f(\sum_{i=1}^{m} \lambda_i X_i)$$

where X_1, X_2, $\cdots$, X_m are m points in the n-dimensional space and $\sum_{i=1}^{m} \lambda_i = 1$ and $\lambda_i \geq 0 (i = 1, 2, \cdots, m)$

Theorem 1. *Given a set of tasks with individual release times, deadlines and precedence constraints, our scheduling algorithm is guaranteed to find a schedule with minimum lateness on a single processor with continuous voltages.*

Proof: It is known that earliest edge-consistent deadline first strategy is guaranteed to find a schedule with minimum lateness [19]. We just need to prove that in a schedule σ for P' computed by our algorithm, no task will miss its new deadline P'. Suppose that a task T_i misses its new deadline in σ. T_i must be in a critical task subset S' and S' must have at least two tasks, including T_i. Otherwise, T_i cannot miss its new deadline in σ. Since T_i misses its new deadline, its processor utilisation must be greater than that of S'. By the definition of a critical task subset, T_i cannot be included in S', which leads to a contradiction.

Theorem 2. *Given a set of tasks with individual release times, deadlines and precedence constraints, our scheduling algorithm is guaranteed to find a schedule with minimum energy consumption on a single processor with continuous voltages.*

Proof: Let S' be a critical task subset computed by our algorithm, T_1, T_2, $\cdots$, T_m be all tasks in S' and σ be an optimal schedule for S'. We distinguish two power models.

1. Under the power model where the dynamic power is the dominant source of the processor power consumption, there is no idle time within the interval of S' in σ. The total energy consumption W of all tasks in σ is $\sum_{i=1}^{m} \frac{P(v_i)c_i f_{max}}{f_i}$, where v_i and f_i are the dynamic voltage and the clock frequency, respectively, of the processor when task T_i is running. Let $c = \sum_{i=1}^{m} \frac{c_i f_{max}}{f_i}$ and $\eta_i = \frac{c_i f_{max}}{f_i c}$. Note that $\sum_{i=1}^{m} \eta_i = 1$ and $c = d_{max} - r_{min} - l(S')$, where d_{max} and r_{min} are the maximum deadline and minimum release time, respectively, of all tasks in S' and $l(S')$ is the total length of all forbidden intervals in $[r_{min}, d_{max}]$. By the property of convex functions, we have

$$W = \sum_{i=1}^{m} \frac{P(v_i)c_i f_{max}}{f_i} = c \sum_{i=1}^{m} \eta_i P(v_i)$$
$$\geq cP(\sum_{i=1}^{m} \eta_i v_i) \geq cP(v_{opt})$$

   Where v_{opt} is the dynamic voltage which corresponds to the slowest processor frequency for S' i.e. $\frac{\sum_{i=1}^{m} c_i}{c} f_{max}$.
   Note that $cP(v_{opt})$ is the total energy consumption of all tasks in our schedule for S'. Therefore our schedule for S' is a minimum energy schedule.

2. Under the power model where both the dynamic power and leakage power are the dominant sources of the processor power consumption, idle time may exist within the interval of S' in σ. The total energy consumption W of all tasks in σ is $\sum_{i=1}^{m} \frac{P(v_i,f_i)c_i f_{max}}{f_i}$, where v_i and f_i are the dynamic voltage and the clock frequency, respectively, of the processor when task T_i is running. Let $c = \sum_{i=1}^{m} \frac{c_i f_{max}}{f_i}$ and $\eta_i = \frac{c_i f_{max}}{f_i c}$. Note that $\sum_{i=1}^{m} \eta_i = 1$. By the property of convex functions, we have

$$W = \sum_{i=1}^{m} \frac{P(v_i,f_i)c_i f_{max}}{f_i} = c\sum_{i=1}^{m} \eta_i P(v_i,f_i)$$
$$\geq cP(\sum_{i=1}^{m} \eta_i v_i, \sum_{i=1}^{m} \eta_i f_i)$$

which implies that a single processor frequency for all tasks in a critical task set is the necessary condition for minimising the total energy consumption of all tasks in a critical task set. Since our algorithm computes an optimal processor clock frequency for all tasks in S' which minimises the power function, our schedule for S' is a minimum energy schedule.

Next we analyse the time complexities of our algorithms. First, we analyse the time complexity of transforming the original problem instance P to the precedence-free problem instance P' as follows.

1. The edge-consistent release times and deadline of all tasks can be computed by using breadth-first search, which takes $O(e)$ time, where e is the number edges in the precedence graph.
2. The EDF schedule for P can be computed using a priority queue, which takes $O(n \log n)$, where n is the number of tasks.
3. It takes $O(n)$ time to set the release times and deadlines for all tasks in P'.

Therefore, it takes $O(e + n \log n)$ time to construct the precedence-free problem instance P'.

Under the dynamic power model, we can use Yao's algorithm to compute the critical task sets. The time complexity of Yao's algorithm is $O(n^3)$. [2] Therefore, the time complexity of our algorithm for the dynamic power model is $O(n^3)$.

To facilitate the analysis of the time complexity of our algorithm for the dynamic and leakage power model, we assume that there are m critical task subsets for a problem instance P. Let S_1, S_2, $\cdots$, S_m be the m critical task subsets, k_i the number of remaining tasks in P and p_i the number of forbidden intervals when S_i is computed. We analyse the time complexity of our algorithm for computing the optimal speed for all tasks in each critical task subset $S_i (i = 1, 2, \cdots, m)$ as follows.

[2] The time complexity of Yao's algorithm was said to be reducible to $O(n \log^2 n)$, but the claim was withdrawn in [2].

1. The time complexity for finding a critical task subset is $O(n^2)$ [1].
2. When modifying the release time and the deadline for each remaining task in P, we can sort all maximum forbidden intervals and use binary search to determine which case is applicable. Therefore, it takes $O(p_i \log p_i + k_i \log p_i)$ to modify the release times and deadlines for all tasks in S_i.

Therefore, the time complexity for computing all critical intervals is $O(mn^2) + O(\sum_{i=1}^{m} (p_i \log p_i + k_i \log p_i)) = O(n^3)$.

5 Conclusion

We proposed two polynomial-time algorithms for finding a minimum energy schedule for a set of tasks with individual release times, deadlines and precedence constraints on a single processor with continuous voltages. Our algorithms are guaranteed to find a schedule with both minimum lateness and minimum energy consumption under two power models. Under the first power model, the dynamic power is the dominant source of the processor power consumption and the leakage power is negligible. Under the dynamic and leakage power model, both the dynamic power and the leakage power are significant sources of the processor power consumption. Under the dynamic power model, the time complexities of both algorithms are $O(n^3)$, where n is the number of tasks.

In this paper, voltage transition overheads are ignored. It is not known if the problem can also be optimally solved in polynomial-time if voltage transition overheads are included. Another interesting problem is how to generalise our algorithms to multiple processor scheduling. In the existing work on leakage-aware DVS scheduling for multiple processor real-time systems, the execution speed of each task is computed individually. By the property of convex functions, more energy would be saved if all tasks in a critical subset are executed at the same speed.

References

1. Frances Yao, Alan Demers and Scott Schenker. A Scheduling Model for Reduced CPU Energy. The proceedings of Annual Symposium on Foundation of Computer Science,1995, Pages 374-382.
2. Minming Li and Frances Yao. An Efficient Algorithm for Computing Optimal Discrete Voltage Schedules. Siam Journal on Computing, 35(3), 2005, Pages 658-671.
3. Tohru Ishihara and Hiroto Yasuura. Voltage Scheduling Problem for Dynamically Variable Voltage Processors. Proceedings of International Symposium on Low Power Electronics and Design, 1998, Pages 197-202.
4. Inki Hong, Miodrag Potkonjak, Mani B. Srivastava. On-line Scheduling of Hard Real-Time Tasks on Variable Voltage Processor. Proceedings of International Conference on Computer-Aided Design, November 1998, Pages 458-470.
5. Amit Sinha and Anantha P. Chandrakasan. Energy Efficient Real-Time Scheduling. Proceedings of International Conference on Computer-Aided Design, 2001, Pages 458-470.

6. Gang Quan and Xiaobo Hu. Energy Efficient Fixed-Priority Scheduling for Real-Time Systems on Variable Voltage Processors. Proceedings of Design Automation Conference, 2001, Pages 828-833.
7. Yumin Zhang, Xiaobo Sharon Hu, and Danny Chen. Task Scheduling and Voltage Selection for Energy Minimisation. Proceedings of Design Automation Conference, June 2002.
8. Gang Quan, Linwei Niu, Xiaobo Sharon H and Mochocki, B. Fixed Priority Scheduling for Reducing Overall Energy on Variable Voltage Processors. Proceedings of Real-Time Systems Symposium, 2004.
9. D. Shin, J. Kim and S. Lee. Low-Energy Intra-Task Voltage Scheduling Using Static Timing Analysis. Proceedings of Design Automation Conference, June, 2001, Pages 438-443.
10. S. M. Martin, K. Flautner, T. Mudge and D. Blauuw. Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Low Power Microprocessors under Dynamic Workloads. Proceedings of International Conference on Computer-Aided Design, Nov. 2002, Pages 721-725.
11. Le Yan, Jiong Luo and Niraj K. Jha. Combined Dynamic Voltage Scaling and Adaptive Body Biasing for Heterogeneous Distributed Real-Time Embedded Systems. Proceedings of 2003 International Conference on Computer-Aided Design, Nov. 2003, Pages 30-38.
12. Alexandru Andrei, Marcus Schmitz, Petru Eles, Zebo Peng, Bashir M. Al-Hashimi: Overhead-Conscious Voltage Selection for Dynamic and Leakage Energy Reduction of Time-Constrained Systems. Proceedings of 2004 Design, Automation and Test in Europe Conference and Exposition (DATE 2004), Pages 518-525.
13. Ravindra Jejurikar, Cristiano Pereira, Rajesh K. Gupta: Leakage aware dynamic voltage scaling for real-time embedded systems. Proceedings of Design Automation Conference, June, 2004, Pages 275-280.
14. Ravindra Jejurikar, Rajesh K. Gupta: Procrastination scheduling in fixed priority real-time systems. Proceedings of ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems, June, 2004, Pages 57-66.
15. Han-Saem Yun and Jihong Kim. On Energy-Optimal Voltage Scheduling for Fixed Priority Hard Real-Time Systems. ACM Transactions on Embedded Computing Systems, Vol. 2, No. 3, August 2003, Pages 393-430.
16. Woo-Cheol Kwon and Taewhan Kim. Optimal Voltage Allocation Techniques for Dynamically Variable Voltage Processors. ACM Transactions on Embedded Computing Systems, Vol. 4, No. 1, February 2005, Pages 211-230.
17. N. H. E Weste and K. Eshraghian. Principle of CMOS VLSI Design. Addison Wesley, 1993.
18. http://www-device.eecs.berkley.edu/ ptm/introduction.html
19. Peter Brucker. Scheduling Algorithms. Springer, 2004.
20. Stephen Boyd and Lieven Vandenberghe. Convex Optimisation. Cambridge University Press, 2001.