Minimum Power Consumption for Rate Monotonic Tasks

Except where reference is made to the work of others, the work described in this thesis is my own or was done in collaboration with my advisory committee. This thesis does not include proprietary or classified information.

Chiao Ching Huang

Certificate of Approval:

Cheryl Seals
Associate Professor
Computer Science and Software Engineering

Sanjeev Baskiyar, Chair
Associate Professor
Computer Science and Software Engineering

Tin-Yau Tam
Professor
Mathematics and Statistics

George T. Flowers
Dean
Graduate School

Minimum Power Consumption for Rate Monotonic Tasks

Chiao Ching Huang

A Thesis

Submitted to

the Graduate Faculty of

Auburn University

in Partial Fulfillment of the

Requirements for the

Degree of

Master of Science

Auburn, Alabama
December 19, 2008

Minimum Power Consumption for Rate Monotonic Tasks

Chiao Ching Huang

Permission is granted to Auburn University to make copies of this thesis at its discretion, upon the request of individuals or institutions and at their expense. The author reserves all publication rights.

Signature of Author

Date of Graduation

Vita

Chiao Ching Huang was born in Tainan City, Taiwan, on March 19, 1978. She graduated from Feng-Chia University with a Bachelor of Science degree in 2000. After working for Chinese Petroleum Corp. for almost five years, in 2005 fall, she entered the graduate program in the Department of Computer Science and Software Engineering at Auburn University. While in pursuit of her Master of Science degree, she worked as a graduate assistant for Auburn University Graduate School under Deans George Flowers and Joe Pittman.

Thesis Abstract

Minimum Power Consumption for Rate Monotonic Tasks

Chiao Ching Huang

Master of Science, December 19, 2008
(B.S., Feng Chia University, 2000)

62 Typed Pages

Directed by Sanjeev Baskiyar

Embedded computer systems are widely used in modern life and their use is expanding. One of the typical constraints in embedded systems, particularly in standalone devices, is their low power capacity. One way to expand the lifetime of battery is to reduce its power consumption; because of the quadratic relationship between power consumption in CMOS circuits and CPU voltage, researchers now can achieve power reduction by scaling down its supply voltage by applying Dynamic Voltage Scaling (DVS). However, reducing supply voltage also slows down CPU speed since supply voltage has a proportional relationship with clock frequency of processor, namely, CPU speed. As a result, DVS succeeds at the cost of system performance. However, in a Real-Time embedded environment, especially in Hard Real-Time embedded Systems, timing constraint is a critical element that cannot be ignored. Therefore, it is difficult to balance the power savings and system throughput so that all tasks will still complete before their deadlines.

In this thesis, we focus on tasks scheduled by Rate Monotonic (RM) algorithm in a Hard Real-Time embedded environment. We derive an equation for scaling worst case execution time (WCET) of each task and expanding each WCET with different factors according to corresponding task computation time until slack times are fully occupied. Different from other approaches, we combine the power consumption equation with the constraint of Rate Monotonic schedulability test (RM test). From the result of this solution, we find the minimum power consumption, at which the RM task set can still pass the RM test and guarantee all tasks will meet deadlines. Our approach can be categorized as an off-line Intra-Task Dynamic Voltage Scaling (IntraDVS).

Acknowledgments

I would like to thank God for giving me an opportunity to attend graduate school and pursue a master's degree. I would also like to thank Him for the immense strength, constant support and unconditional love.

I would like to extend my sincere appreciations to the members of my master's committee for their knowledge, time and assistance.

I am thankful to my academic advisor, Dr. Sanjeev Baskiyar, for his support, patience, encouragement and priceless guidance. His enthusiasm, creativity, and careful review have helped me accomplish an enormous task.

I am especially thankful to Dr. Tin-Yau Tam for guiding me through the most critical part of my research. The math proofs and the Matlab program were written under his valuable directions.

I would like to take this opportunity to thank the Graduate School for providing me with an assistantship and financial support.

Lastly, I thank my beloved family and friends for their endless love, support, patience and encouragement.

Style manual or journal used Journal of Approximation Theory (together with the style known as "aums"). Bibliograpy follows van Leunen's *A Handbook for Scholars.*

Computer software used The document preparation package TEX (specifically LATEX) together with the departmental style-file `aums.sty`.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Current Problem

Maturity of wireless technology and mobile markets have dramatically changed lifestyles of people. More and more consumers depend on wireless or portable devices, such as cellular phone, personal digital assistant (PDA), digital camera or global positioning system (GPS). With the greatly growing popularity of handheld devices, power supply or battery lifetime has always been one of the major determinants in the purchase decision. For decades, scientific researchers have sought to find the optimal solution to save power in handhelds and produce more energy efficient power supplies.

As far as Real-Time embedded systems are concerned, power consumption becomes an even bigger issue. One of the well-known characteristics of a Real-Time Operating System is its time constraint. In a Soft Real-Time system, missed deadlines may not affect the whole system's functionality. However, a missed deadline in a Hard Real-Time system could lead to a non-recoverable system failure. Reducing power dissipation often means sacrificing system performance. That is, tasks have the potentials of late completions, which cannot be tolerated in a Hard Real-Time system [12].

1.2 Motivation

This thesis focuses on Rate Monotonic (RM) schedulable tasks for Hard Real-Time embedded systems. The task model is preemptive and static priority driven. The approach is to expand computation time of each task until its combined CPU utilization approximately reaches the upper bound of Rate Monotonic schedulability test (RM test) [9]; thus, lowering CPU speed and supplied voltage, yet guaranteeing none of the tasks misses its deadline. With the method of Lagrange Multiplier, the approach deduces a specific formula and iteration yielding the exact scaling factor for each task. Then, the computation time of each corresponding task is increased by applying this scaling factor. Consequently, the minimum power consumption can be achieved and all the tasks will be processed within their time constraints.

1.3 Thesis Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews recent studies. Specifically, different kinds of DVS approaches and their challenges will be introduced and analyzed. Chapter 3 gives a detailed explanation for the formulation and deduction of the problem. In Chapter 4, we discuss mathematical part and explain the solution to the problem described in Chapter 3. We provide a complete proof and present the algorithm inferred from this study. Chapter 5 includes some experiments and observations during this research and discusses issues regarding the result and suggestions for future work. Chapter 6 concludes this study.

Chapter 2

Dynamic Voltage Scaling

A large number of research activities have been conducted in the past decades to achieve power reduction. In this chapter, we introduce Dynamic Voltage Scaling (DVS) techniques and present some of its strategies.

Before we start, we explain some terms in scheduling Hard Real-Time tasks.

Worst Case Execution Times (WCETs)

As implied by the name, worst case execution time (WCET) is the estimated maximum possible run-time for a request to complete its task without missing its deadline. Just as W. Kim et al.[7] explained, “In scheduling Hard Real-Time tasks, in order to guarantee the timing constraint of each task, the execution times of tasks are usually assumed to be the worst case execution times (WCETs).” WCET is often used as a standard design scheme to schedule Hard Real-Time tasks.

Slack Time

Slack times can be classified as static slack times and dynamic slack times [7]. Static slack time is defined as a time frame between the moment a task finishes its execution and its next invocation. Off-line (or static) approaches, e.g. static voltage scaling, path-based method, stochastic method and maximum constant speed should fully utilize these static slack times known in advance by reducing CPU frequency and consequently reducing supply voltage.

Dynamic slack time is the difference between actual execution time and WCET[1]. The actual task execution time is unpredictable because of processor speed at run-time. In fact, a large number of tasks finish earlier than their WCETs [5]. Due to this reason, static approaches cannot significantly save power consumption. On-line approaches such as stretching to NTA, priority-based slack stealing and utilization updating exploit this characteristic by reclaiming dynamic slack times and recompute CPU frequency at task release time to lower power consumption.

Dynamic Voltage Scaling (DVS) is one way to reduce power consumption and it basically tries to change power supply voltage dynamically to slow down CPU speed in order to save energy. DVS is based on two concepts and has been a mainstream in the area of power reduction. One concept is the quadratic relationship between power dissipation and supply voltage ($p \propto V^2$). Another concept of developing DVS is that even if in Hard Real-Time environment, as long as tasks can complete on time, there is no reason to require higher system throughput.

2.1 CMOS Circuit Design

Since the hardware makers make 'limitless' number of transistors possible, A. P. Chandrakasan et al. [2] came up with an idea of redesigning system architecture by

[1]In this case, actual execution time is less than WCET

delaying CMOS circuit behaviors as much as possible while using the lowest potential supply voltage resulting in dynamic power dissipation. This architecture-driven voltage scaling technique is capable of achieving the goal of power savings without regard to its complexity.

2.2 DVS Algorithm with DVS Processor

With the help of CMOS circuit technique improvement, more and more variable-voltage microprocessors based on CMOS logic have been brought to the market. Researchers then exploited this trend to develop all kinds of DVS algorithms so as to make use of these available DVS processors.

DVS algorithms can be further classified into intra-task DVS (IntraDVS) and inter-task DVS (InterDVS) algorithms according to how they use slack times. IntraDVS passes on slack times from the current task to itself in the next cycle [4, 13]. Usually, IntraDVS determines supply voltage off-line. InterDVS distributes the slack times from the current task to the following tasks. Most currently existing approaches belong to InterDVS [14, 12, 5, 3, 8]. Besides these two, researchers further incorporate the strengths of IntraDVS and InterDVS and comprise a hybrid DVS (HybridDVS) algorithm [7, 3, 8].

First DVS Algorithm was proposed by M. Weiser et al. [17]. This approach is an average throughput-based DVS algorithm and only considers Soft Real-Time environment. It cannot guarantee Hard Real-Time deadlines will be met. Shin and Choi [14] considered delay overheads from power-down mode and voltage switching, and developed Low Power Fixed Priority Scheduling (LPFPS) with slight modification of Real-Time scheduler, in which they combined off-line and on-line approaches. However, voltage switching incurs delay overhead. Therefore, their solution still needs a trade-off analysis between increased task execution time and power consumption.

Similar to our approach, the scheme proposed by Manzak and Chakrabarti [11] also tried to find the operating voltage for the processor while it executes each task. They formulate their equation to minimize total energy consumption subject to a constant total computation time. With the method of Lagrange Multiplier, Manzak and Chakrabarti obtain a relationship among task voltages with the constraint of minimum energy consumption. They further develop an iterative algorithm accordingly to adjust task slack times and therefore adjust voltage assignments. For RM scheduling, the iterative voltage assignments start until combined CPU utilization reaches upper bound of RM test. Still, there is a trade off between energy/power savings and the complexity of this algorithms.

Stachastic IntraDVS described by F. Gruian [4] not only scaled voltage to fill static slack times but also concerned dynamic slack times caused by run-time task executions and combined on-line slack distribution method to achieve the goal of minimizing energy consumption while meeting all deadlines.

Swaminathan and Chakrabarty [15] took into consideration the effect of voltage switching times on the energy consumption of the task set and presented a novel mixed-integer linear programming model called extended-low-energy earliest-deadline-first (E-LEDF) for the NP-complete scheduling algorithm. Swaninathan and Chakrabarty's approach is specified for non-preemptive Earliest-Deadline-First (EDF) task set rather than preemptive RM task set.

Pillai and Shin [12] used a scaling factor obtained from lowest CPU frequencies in a given task set for all tasks and the results show statically scaled RM tasks cannot reduce energy consumption aggressively. Y. H. Tsai et al. [8] further exploited Pillai and Shin's static voltage scaling approach and incorporate with their Deferred-Workload-based inter-task DVS (dwDVS).

Liu and Mok [10] defined the available cycle function (ACF) and the required cycle function (RCF) to limit CPU speed so that there is no idle cycle when the CPU executes and no job misses its deadline. To solve energy minimization problem, they propose an off-line algorithm and an low complexity on-line dynamic algorithm to reclaim dynamic slack cycles. Liu and Mok's solution can be used with different Real-Time schedulers.

Except that W. Kim et al. [13], A. Manzak and C. Chakrabarti [11] only used off-line approaches, most of researchers combined on-line and off-line approaches to achieve power savings. Shin and Choi [14] combined not only off-line, on-line approaches but also brought the processor to a power-down mode. Hakan et al. [5] included an off-line solution to compute the optimal speed and presented on-line speed adjustment mechanism to reclaim dynamic slack times and further predict task future execution. Similarly, Gruian [4] addressed off-line task stretching to obtain scaling factor and incorporated on-line slack distribution to further utilize dynamic slack times.

2.3 Problems

Although off-line DVS algorithms have lower complexity and are easy to implement, on-line algorithms can result in even more energy savings; however, due to the difficulty of run-time workload's prediction as mentioned above, on-line approaches also have higher complexity and higher chances to miss tasks deadlines. In order to compromise with on-line approaches, researchers often speed up tasks to catch up with their deadlines. Yet, accelerating tasks also presents increasing supply voltage from CPU frequency. Therefore, it rarely guarantees the minimum power consumption; instead, it has the risk to cause higher energy.

Also, even though some of above studies are for Hard Real-Time environment and the task set is scheduled with RM algorithm, few of them mentioned if the combined CPU utilization of task set passes the upper bound of RM test. In other words, it could still meet all the deadlines of tasks in run-time but without assurance.

Chapter 3
Formalization

Since this research is based on RM scheduling algorithm, the system model has the same assumption as Liu and Layland's [9]. Given a task set of n number of tasks t_i, $i = 1, 2, \ldots, n$, we assume that the time period T_i, $i = 1, 2, \ldots, n$, are periodic and each deadline is equal to its period. All the tasks are independent and has its own computation time C_i, $i = 1, 2, \ldots, n$.

3.1 Rate Monotonic Schedulability Test

The combined CPU utilization u for multiprogramming in a Hard Real-Time Embedded Operating System is:

$$u = \sum_{i=1}^{n} \frac{C_i}{T_i} = \frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_n}{T_n} \tag{3.1}$$

where C_i is the computation time, T_i is the time period and n is the number of tasks in a task set.

For Rate Monotonic (RM) Scheduling Algorithm, if $u \leq n(2^{\frac{1}{n}} - 1)$, Liu and Layland [9] indicated that the tasks t_i, $i = 1, 2, \ldots, n$, will be schedulable and will not miss their deadlines and is called Rate Monotonic schedulability test (RM test). Thus, the upper bound of RM test is $n(2^{\frac{1}{n}} - 1)$. RM test "is said to be sufficient but not necessary." [1] On the other hand, if a task set passes RM test, it will meet all deadlines; if it fails the test, it may or may not fail at run-time.

3.2 Power Consumption

The power consumed per CPU cycle is

$$p = \alpha f N V^2 \tag{3.2}$$

where p is the power dissipation, α is the average activity factor, f is the clock frequency of processor or the CPU frequency, N is the switching capacitance of wires and transistors gates and V is the supply voltage.

From (3.2), the power dissipation p is proportional to CPU frequency f, switching capacitance N and supply voltage V^2. Since switching capacitance N is proportional to the numbers of transistors on the chip [16] and all the tasks use the same processor, N is constant. We only consider to reducing CPU frequency f or the supply voltage V here to achieve power reduction.

Since V is approximately proportional to CPU frequency f, reducing f also reduces V proportionally [18]. We can say the power dissipation p is proportional to the cube of its CPU frequency f.

$$p \propto f^3 \tag{3.3}$$

Reducing the CPU frequency f by a factor $\frac{1}{X_i}$, where X_i is a scaling factor and is always greater than or equal to 1, results in increasing the computation time C_i to

X_iC_i. Thus we have the new combined CPU utilization equation u' as follows:

$$u' = \sum_{i=1}^{n} \frac{X_iC_i}{T_i} = \frac{X_1C_1}{T_1} + \frac{X_2C_2}{T_2} + \cdots + \frac{X_nC_n}{T_n} \leq n(2^{\frac{1}{n}} - 1) \tag{3.4}$$

3.3 Energy Consumption

The energy consumption E_i for executing task t_i is:

$$E_i \propto C_i \times p_i \tag{3.5}$$

where C_i is the computation time of task t_i, p_i is the power dissipation of the processor.

According to (3.3), we could modify energy consumption equation E_i as the following form

$$E_i \propto C_i \times {f_i}^3 \tag{3.6}$$

where f_i is the CPU frequency.

Since f is reduced to $\frac{f}{X_i}$, along with the equivalent scaling X_iC_i, the scaled energy consumption E_i' becomes

$$E_i' \propto X_iC_i \times (\frac{f}{X_i})^3 = \frac{C_i \times f^3}{X_i^2} \tag{3.7}$$

3.4 Problem Definition

Based on the above derivations, we can restate our problem in the following form:

Given $u = \sum\limits_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$, our goal is to find $\hat{X} := (\hat{X}_1, \ldots, \hat{X}_n)$ which yields

$$\min \sum_{i=1}^{n} \frac{C_i}{\hat{X}_i^2}$$

subject to the constraints $\sum\limits_{i=1}^{n} \frac{\hat{X}_i C_i}{T_i} \leq n(2^{\frac{1}{n}} - 1)$ and $1 \leq \hat{X}_i \leq \frac{T_i}{C_i}$.

Associated with our problem are the following parameters:

- u: combined CPU utilization
- n: number of tasks in a task set
- C_i: computation time of task t_i
- T_i: time period of task t_i
- X_i: scaling factor of task t_i

Chapter 4

Solution to the optimization problem

4.1 Mathematical Statements and Proofs

Based on the above derivations, we restate the problem in the following mathematical form:

Problem: Find $\hat{X} := (\hat{X}_1, \ldots, \hat{X}_n)$ which yields

$$\min \sum_{i=1}^{n} \frac{C_i}{X_i^2}, \tag{4.1}$$

where $0 < C_i$ for all i, subject to the constraints

1. $\sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1)$ with $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$.

2. $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, n$.

Let

$$f(X) := \sum_{i=1}^{n} \frac{C_i}{X_i^2}.$$

Here n is the number of tasks in a task set, C_i is the computation time of task t_i, T_i is the time period of task t_i, and X_i is the scaling factor of task t_i.

When $n = 1$ the minimization problem is trivial: $f(X) = \frac{C_1}{X_1^2}$ and its minimum is attained at $X = \frac{T_1}{C_1}$. So from now on we only need to consider $n \geq 2$.

Remark 4.1. Set

$$S = \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1), 1 \leq X_i \leq \frac{T_i}{C_i}\}. \tag{4.2}$$

The minimization problem can be written as: find $\hat{X} \in S$ that gives

$$\min_{X \in S} f(X).$$

The condition $\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n(2^{1/n} - 1)$ ensures that the point $(1, \ldots, 1)$ is in the region. So the constraint set S (the domain of the minimization problem) is nonempty. Notice that S is a closed and bounded set in $\mathbb{R}^n$, that is, S is a compact set. Moreover the function $f(X)$ is continuous on the compact set S and f is always positive over S. By Weierstrass's theorem [6], there is a minimum of f in S. So the minimization problem (4.1) is solvable. However Weierstrass's theorem does not provide a mean to find the point(s) $\hat{X} \in S$ such that $f(\hat{X})$ is the minimum.

Lemma 4.2. The sequence $h(n) := n(2^{1/n} - 1)$ is strictly monotonic decreasing and $\lim_{n\to\infty} h(n) = \ln 2$. Moreover $h(n) < 1$ for all positive integers n except $n = 1$; $h(1) = 1$.

Proof. Clearly $h(1) = 1$. Let $a := 2^{\frac{1}{n(n+1)}} > 1$. By using the identity $a^n - 1 = (a-1)(a^{n-1} + a^{n-2} + \cdots + a + 1)$, we obtain

$$\begin{aligned} h(n) - h(n+1) &= n(2^{\frac{1}{n}} - 1) - (n+1)(2^{\frac{1}{n+1}} - 1) \\ &= n\left[\left(2^{\frac{1}{n(n+1)}}\right)^{n+1} - 1\right] - (n+1)\left[\left(2^{\frac{1}{n(n+1)}}\right)^{n} - 1\right] \\ &= (a-1)\left[n(a^n + a^{n-1} \cdots + 1) - (n+1)(a^{n-1} + \cdots + 1)\right] \\ &= (a-1)[na^n - (a^{n-1} + \cdots + 1)] \geq 0 \end{aligned}$$

since $a^n \geq a^i$ for $i = 1, \ldots, n-1$ as $a > 1$. So $h(n)$ is strictly monotonic decreasing. By l'Hospital's rule,

$$\lim_{n\to\infty} h(n) = \lim_{n\to\infty} \frac{2^{1/n} - 1}{\frac{1}{n}} = \lim_{n\to\infty} \frac{-\frac{1}{n^2} 2^{1/n} \ln 2}{-\frac{1}{n^2}} = \lim_{n\to\infty} 2^{1/n} \ln 2 = \ln 2.$$

□

Remark 4.3. When $n \geq 2$, one can rewrite the set S as

$$S = \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1), 1 \leq X_i\},$$

i.e., one can ignore the condition $X_i \leq \frac{T_i}{C_i}$ in (4.2) since it follows from $\sum_{i=1}^{n} \frac{X_i C_i}{T_i} \leq n(2^{1/n} - 1) < 1$ by Lemma 4.2 and $1 \leq X_i$. But we keep using (4.2) as the description of S.

Each $X \in S$ must satisfy

$$X_i < \frac{T_i}{C_i}, \quad i = 1, \ldots, n, \tag{4.3}$$

otherwise $X_j = \frac{T_j}{C_j}$ for some j so that by Lemma 4.2 we would have

$$\sum_{i=1}^{n} \frac{X_i C_i}{T_i} \geq \frac{X_j C_j}{T_j} = 1 > n(2^{1/n} - 1),$$

a contradiction.

Lemma 4.4. The minimization problem (4.1) only has solution(s) in the compact set R, where

$$R := \{x \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} = n(2^{1/n} - 1), 1 \leq X_i < \frac{T_i}{C_i}\} \subset S. \tag{4.4}$$

Proof. Set

$$X_\epsilon := X + \epsilon(1, \ldots, 1) = (X_1 + \epsilon, \ldots, X_n + \epsilon).$$

If the minimum of $f(X)$ occurred at X over the region defined by

$$\sum_{i=1}^{n} \frac{X_i C_i}{T_i} < n(2^{1/n} - 1),$$

then by (4.3) for sufficiently small ϵ

$$1 \leq (X_\epsilon)_i \leq \frac{T_i}{C_i} \qquad i = 1, \ldots, n$$

and

$$\sum_{i=1}^{n} \frac{(X_\epsilon)_i C_i}{T_i} = \sum_{i=1}^{n} \frac{(X_i + \epsilon) C_i}{T_i} = \sum_{i=1}^{n} \frac{X_i C_i}{T_i} + \epsilon \sum_{i=1}^{n} \frac{C_i}{T_i} < n(2^{1/n} - 1).$$

In other words, $X_\epsilon \in S$. Clearly $f(X_\epsilon) < f(X)$, a contradiction. $\square$

From now on we set

$$K := n(2^{1/n} - 1).$$

By Lemma 4.4 the minimization problem 4.1 is reduced to: find $\hat{X} \in R$ that gives

$$\min_{X \in R} f(X). \tag{4.5}$$

The following result reveals a special ordering of the solution according to the ordering of T's.

Theorem 4.5. Suppose that $T_{\sigma(1)} \geq \cdots \geq T_{\sigma(n)}$ with some permutation $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and the minimization problem attains its minimum at $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_n) \in R$. Then $\hat{X}_{\sigma(1)} \geq \cdots \geq \hat{X}_{\sigma(n)}$ (≥ 1). In particular, if $T_1 \geq \cdots \geq T_n$, then $\hat{X}_1 \geq \cdots \geq \hat{X}_n$.

Proof. Without loss of generality, we may assume that σ is the identity, i.e., $T_1 \geq \cdots \geq T_n$. Suppose on the contrary that $\hat{X}_j > \hat{X}_i$ (≥ 1) for some $1 \leq i < j \leq n$. By

(4.3)

$$\hat{X}_i < \frac{T_i}{C_i}, \quad i = 1, \dots, n.$$

Let

$$\hat{X}_i(\delta) = \hat{X}_i + \delta, \qquad \hat{X}_j(\delta) = \hat{X}_j - \delta \frac{C_i T_j}{C_j T_i}$$

and $\hat{X}_k(\delta) = \hat{X}_k$ for all $k \neq i, j$. So

$$\sum_{k=1}^{n} \frac{C_k \hat{X}_k(\delta)}{T_k} = \sum_{k \neq i,j} \frac{C_k \hat{X}_k(\delta)}{T_k} + \frac{C_i \hat{X}_i(\delta)}{T_i} + \frac{C_j \hat{X}_j(\delta)}{T_j} = K.$$

Since $\hat{X}_i(\delta) < \frac{T_i}{C_i}$ and $1 < \hat{X}_j(\delta)$ for sufficiently small $\delta > 0$, we have $\hat{X}(\delta) \in R$. Set

$$\begin{aligned}
\varphi(\delta) &:= f(\hat{X}(\delta)) \\
&= \frac{C_1}{\hat{X}_1^2} + \cdots + \frac{C_i}{(\hat{X}_i(\delta))^2} + \cdots + \frac{C_j}{(\hat{X}_j(\delta))^2} + \cdots + \frac{C_n}{\hat{X}_n^2} \\
&= \frac{C_1}{\hat{X}_1^2} + \cdots + \frac{C_i}{(\hat{X}_i + \delta)^2} + \cdots + \frac{C_j}{(\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^2} + \cdots + \frac{C_n}{\hat{X}_n^2}.
\end{aligned}$$

Now

$$\begin{aligned}
\frac{d\varphi(\delta)}{d\delta} &= \frac{-2C_i}{(\hat{X}_i + \delta)^3} + \frac{2\frac{C_i T_j}{T_i}}{(\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^3} \\
&= \frac{2C_i \left[(\hat{X}_i + \delta)^3 \frac{T_j}{T_i} - (\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^3 \right]}{(\hat{X}_i + \delta)^3 (\hat{X}_j - \delta \frac{C_i T_j}{C_j T_i})^3}.
\end{aligned}$$

So

$$\varphi'(0) = \frac{2C_i\left[\hat{X}_i^3\frac{T_j}{T_i} - \hat{X}_j^3\right]}{\hat{X}_i^3\hat{X}_j^3} < 0,$$

since $1 \leq \hat{X}_i < \hat{X}_j$ and $T_i \geq T_j > 0$. This contradicts the assumption that $f(\hat{X})$ is the minimum. $\square$

We will first solve a slightly different problem: find the $X \in R'$ that yields

$$\min_{X\in R'} f(X)$$

where

$$R' = \{X \in \mathbb{R}^n : \sum_{i=1}^{n} \frac{X_i C_i}{T_i} = K\}.$$

In other words, we remove the condition $1 \leq X_i \leq \frac{T_i}{C_i}$ from the original minimization problem (4.1). Note that $R \subset R'$ and the region R' is represented by the equation

$$\sum_{i=1}^{n} \frac{X_i C_i}{T_i} = K$$

and is a hyperplane in $\mathbb{R}^n$. By the method of Lagrange Multiplier, set

$$\nabla f = \lambda \nabla g$$

which implies

$$-\frac{2C_i}{X_i^3} = \lambda\frac{C_i}{T_i}.$$

Since $C_i \neq 0$,

$$X_i^L = \left(\frac{2T_i}{-\lambda}\right)^{1/3}. \tag{4.6}$$

Substitute (4.6) into $\sum_{i=1}^n \frac{X_i C_i}{T_i} = K$ to have

$$\sum_{j=1}^n \left(\frac{2T_j}{-\lambda}\right)^{1/3} \frac{C_j}{T_j} = K.$$

Thus we have

$$\frac{1}{(-\lambda)^{1/3}} \sum_{j=1}^n (2T_j)^{1/3} \frac{C_j}{T_j} = K. \tag{4.7}$$

So from (4.6) and (4.7)

$$X_i^L = \frac{(2T_i)^{1/3} K}{\sum_{j=1}^n (2T_j)^{1/3} \frac{C_j}{T_j}} = \frac{T_i^{1/3} K}{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}} < \frac{T_i}{C_i} \tag{4.8}$$

(since $K < 1$) which is the only critical point of f over R'. Moreover

$$f(X^L) = \sum_{i=1}^n \frac{C_i}{(X_i^L)^2} = \frac{(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j})^2}{K^2} (\sum_{i=1}^n \frac{C_i}{T_i^{2/3}}) \tag{4.9}$$

is the local minimum value of $f(X)$ over R'. Thus it is the global minimum over R'.

If $T_1 \geq \cdots \geq T_n$, then from (4.8)

$$X_1^L \geq \cdots \geq X_n^L.$$

However we cannot conclude that $1 \leq X_i^L$ for all $i = 1, \ldots, n$, i.e., X^L may not be in R since the second constraint of (4.1) is not satisfied. Nevertheless, it is clear that

$$f(X^L) = \min_{X \in R'} f(X) \leq \min_{X \in R} f(X). \tag{4.10}$$

Theorem 4.6. Let $T_1 \geq \cdots \geq T_n$. Let $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_n) \in R$ be a solution to the minimization problem (4.5).

1. $X_n^L \geq 1$ if and only if $\hat{X}_n \geq 1$. In this case, $\hat{X} = X^L$ is a solution to the minimization problem (4.1).

2. If $X_n^L < 1$, then

$$\hat{X}_1 \geq \cdots \geq \hat{X}_{r-1} > \hat{X}_r = \cdots = \hat{X}_n = 1 \tag{4.11}$$

for some $r = 1, \ldots, n$. Moreover $(\hat{X}_1, \ldots, \hat{X}_{r-1})$ is a solution to the following minimization problem

$$\min \sum_{i=1}^{r-1} \frac{C_i}{X_i^2} + \sum_{i=r}^{n} C_i$$

subject to the (new) conditions

(a) $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, r-1$,

(b) $\sum_{i=1}^{r-1} \frac{C_i X_i}{T_i} \leq \hat{K}$, where $\hat{K} := K - \sum_{i=r}^{n} \frac{C_i}{T_i}$.

Namely,

$$\hat{X}_i = \frac{T_i^{1/3} \hat{K}}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}}, \qquad i = 1, \ldots, r-1. \tag{4.12}$$

In either case, $\hat{X}$ is the unique solution to the minimization problem (4.5).

Proof. By Theorem 4.5 the solution $\hat{X} \in R$ to the minimization problem (4.5) satisfies $\hat{X}_1 \geq \cdots \geq \hat{X}_n$.

(1) By (4.8) $X_1^L \geq \cdots \geq X_n^L$, $X_i^L < \frac{T_i}{C_i}$ for all $i = 1, \ldots, n$, and $\sum_{i=1}^n \frac{X_i^L C_i}{T_i} = K$. So $X_n^L \geq 1$ if and only if $X^L \in R$. In this case, it amounts to $\hat{X} = X^L$ by (4.10) and it is the unique solution since we only have one critical point for the minimization problem over R'.

(2) If $X_n^L < 1$, then $X^L \notin R$. Since X^L is the only critical point in R' and $R \subset R'$, we conclude that the minimum point(s) of f over R must be in the boundary ∂R of R:

$$\partial R := \{x \in \mathbb{R}^n : \sum_{i=1}^n \frac{X_i C_i}{T_i} = K, X_i = 1 \text{ for some } i \text{ or } X_j = \frac{T_j}{C_j} \text{ for some } j\}.$$

Since $X_i < \frac{T_i}{C_i}$ for all i by (4.3), $\hat{X}$ must occur in

$$\hat{\partial} R := \{x \in \mathbb{R}^n : \sum_{i=1}^n \frac{X_i C_i}{T_i} = K, X_i = 1 \text{ for some } i\}.$$

In other words, via Theorem 4.5 we have (4.11), i.e.,

$$\hat{X}_1 \geq \cdots \geq \hat{X}_{r-1} > \hat{X}_r = \cdots = \hat{X}_n = 1,$$

for some $r = 1, \ldots, n$. Because the region

$$R_r := \{(X_1, \ldots, X_{r-1}, 1, \ldots, 1) \in \mathbb{R}^n : \sum_{i=1}^{r-1} \frac{C_i X_i}{T_i} + \sum_{i=r}^{n} \frac{C_i}{T_i} = K,\ 1 \le X_i \le \frac{T_i}{C_i},\ i = 1, \ldots, r-1\}$$

is a subset of R, $(\hat{X}_1, \ldots, \hat{X}_{r-1})$ is the solution to the following minimization problem

$$\min \sum_{i=1}^{r-1} \frac{C_i}{X_i^2} + \sum_{i=r}^{n} C_i$$

subject to the (new) constraints

1. $1 \le X_i \le \frac{T_i}{C_i}$, $i = 1, \ldots, r-1$,
2. $\sum_{i=1}^{r-1} \frac{C_i X_i}{T_i} \le \hat{K}$, where $\hat{K} := K - \sum_{i=r}^{n} \frac{C_i}{T_i}$.

Since $T_1 \ge \cdots \ge T_{r-1}$ and $\hat{X}_{r-1} > 1$, by Theorem 4.6(1) the minimum is attained at $(\hat{X}_1, \cdots, \hat{X}_{r-1})$ which is provided by the method of Lagrange multiplier, i.e.,

$$\hat{X}_i = \frac{T_i^{1/3} \hat{K}}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}} > 1, \qquad i = 1, \ldots, r-1.$$

Moreover the solution is unique once r is fixed.

To show that $\hat{X}$ is unique, it is sufficient to show that r is unique. Suppose on the contrary that $(\hat{X}_1, \ldots, \hat{X}_{r-1}, 1, \ldots, 1) \in \mathbb{R}^n$ and $(\tilde{X}_1, \ldots, \tilde{X}_{r'-1}, 1, \ldots, 1) \in \mathbb{R}^n$ both yield the minimum, i.e.,

$$f((\hat{X}_1, \ldots, \hat{X}_{r-1}, 1, \ldots, 1)) = f((\tilde{X}_1, \ldots, \tilde{X}_{r'-1}, 1, \ldots, 1)) = \min_{X \in R} f(X)$$

but $r \neq r'$. In other words,

$$\tilde{X}_i = \frac{T_i^{1/3}\tilde{K}}{\sum_{j=1}^{r-1} T_j^{1/3}\frac{C_j}{T_j}} > 1, \qquad i = 1, \ldots, r'-1.$$

For definiteness, assume $r < r'$. Clearly $R_r \subset R_{r'}$ and $(\tilde{X}_1, \ldots, \tilde{X}_{r'-1})$ is the unique solution to the following minimization problem

$$\min \sum_{i=1}^{r'-1} \frac{C_i}{X_i^2} + \sum_{i=r'}^{n} C_i$$

subject to the constraints

1. $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, r'-1$,

2. $\sum_{i=1}^{r'-1} \frac{C_i X_i}{T_i} \leq \tilde{K}$, where $\tilde{K} := K - \sum_{i=r'}^{n} \frac{C_i}{T_i}$.

We would then have $f((\hat{X}_1, \ldots, \hat{X}_{r-1}, 1, \ldots, 1)) > f((\tilde{X}_1, \ldots, \tilde{X}_{r'-1}, 1, \ldots, 1))$, a contradiction. So r is unique. Hence $(\hat{X}_1, \ldots, \hat{X}_{r-1})$ and $\hat{X}$ are unique. □

When $X_n^L < 1$, to solve Problem 4.1, it suffices to determine $r-1$ which is the largest index i such that $\hat{X}_i > 1$ and apply (4.12). The following is very useful with respect to the determination of $r-1$.

Proposition 4.7. Let $T_1 \geq \cdots \geq T_n$. Let $r-1$ be the largest integer i such that $\hat{X}_i > 1$ ($r-1 = 0$ means that all $\hat{X}$'s are 1). Then

$$r - 1 \leq s_1 \tag{4.13}$$

where s_1 denotes the largest integer i such that $X_i^L > 1$ ($s_1 = 0$ means that all X^L's are less than or equal to 1), i.e.,

$$X_1^L \geq \cdots \geq X_{s_1}^L > 1 \geq X_{s_1+1}^L \geq \cdots \geq X_n^L.$$

Evidently, if $X_n^L \geq 1$, then $\hat{X} = X^L$ and $r - 1 = s_1$.

Proof. On the contrary suppose that $r - 1 > s_1$. We would have $r - 1 \geq s_1 + 1$ so that $X_{r-1}^L \leq 1$. From (4.8)

$$X_{r-1}^L = \frac{T_{r-1}^{1/3} K}{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}} \leq 1 \Longleftrightarrow K \leq \frac{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}}{T_{r-1}^{1/3}}.$$

So

$$\begin{aligned} K \left(\sum_{j=r}^n T_j^{1/3} \frac{C_j}{T_j} \right) &\leq \frac{\left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right)}{T_{r-1}^{1/3}} \left(\sum_{j=r}^n T_j^{1/3} \frac{C_j}{T_j} \right) \\ &= \left(\sum_{j=r}^n \left(\frac{T_j}{T_{r-1}} \right)^{1/3} \frac{C_j}{T_j} \right) \left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right) \\ &\leq \left(\sum_{j=r}^n \frac{C_j}{T_j} \right) \left(\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j} \right) \qquad (4.14) \end{aligned}$$

since $T_1 \geq \cdots \geq T_n$. Now

$$\begin{aligned}
\frac{\hat{X}_{r-1}}{X^L_{r-1}} &= \left[\frac{T_{r-1}^{1/3}K'}{\sum_{j=1}^{r-1} T_j^{1/3}\frac{C_j}{T_j}}\right]\left[\frac{\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j}}{T_{r-1}^{1/3}K}\right] \\
&= \frac{K'}{K}\left[\frac{\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j}}{\sum_{j=1}^{r-1} T_j^{1/3}\frac{C_j}{T_j}}\right] \\
&= \left[\frac{K - \sum_{j=r}^{n}\frac{C_j}{T_j}}{K}\right]\left[\frac{\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j}}{\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j} - \sum_{j=r}^{n} T_j^{1/3}\frac{C_j}{T_j}}\right] \\
&= \frac{K\left(\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j}\right) - \left(\sum_{j=r}^{n}\frac{C_j}{T_j}\right)\left(\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j}\right)}{K\left(\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j}\right) - K\left(\sum_{j=r}^{n} T_j^{1/3}\frac{C_j}{T_j}\right)}
\end{aligned}$$

By (4.14) we have $\hat{X}_{r-1} \leq X^L_{r-1} \leq 1$, contradicting the fact that $\hat{X}_{r-1} > 1$. $\square$

Proposition 4.7 leads to an algorithm for the determination of $r-1$ and thus $\hat{X}$. In general we may not be able to locate $r-1$ right after the first application of Lagrange Multiplier (see Proposition 4.8). However we can repeat the process and eventually will get to the value of $r-1$. The following is an algorithm to compute the solution to Problem (4.1).

The algorithm: Arrange $T_1, \ldots, T_n$ so that $T_1 \geq \cdots \geq T_n$.

Step 1: By the method of Lagrange Multiplier, i.e., from (4.8)

$$X_i^{(1)} := X_i^L = \frac{T_i^{1/3}K}{\sum_{j=1}^{n} T_j^{1/3}\frac{C_j}{T_j}} \tag{4.15}$$

so that $X_1^{(1)} \geq \cdots \geq X_n^{(1)}$ by Theorem 4.6. Notice that

$$\begin{aligned} X_i^{(1)} &= \frac{T_i^{1/3} K}{\sum_{j=1}^n T_j^{1/3} \frac{C_j}{T_j}} < \frac{T_i}{C_i}, \ i = 1, \ldots, n, && (4.16) \\ X_1^{(1)} &\geq \frac{K}{\sum_{j=1}^n \left(\frac{T_j}{T_1}\right)^{1/3} \frac{C_j}{T_j}} \geq \frac{K}{\sum_{j=1}^n \frac{C_j}{T_j}} \geq 1 && (4.17) \end{aligned}$$

Then consider the cases:

(a) if $X_n^{(1)} \geq 1$, then $\hat{X} = (X_1^{(1)}, \ldots, X_n^{(1)})$ by Theorem 4.6.

(b) if $X_n^{(1)} < 1$, i.e., there is $0 \leq s_1 \leq n-1$ such that $X_1^{(1)} \geq \cdots \geq X_{s_1}^{(1)} > 1 \geq X_{s_1+1}^{(1)} \geq \cdots \geq X_n^{(1)}$ then by (4.13) $r - 1 \leq s_1$, where

$$\hat{X}_1 \geq \cdots \geq \hat{X}_{r-1} > 1 = \hat{X}_r = \hat{X}_{r+1} = \cdots = \hat{X}_n$$

and $r-1$ is obviously fixed and to be determined. In other words, we know that $\hat{X}_{s_1+1} = \cdots = \hat{X}_n = 1$.

Remark: $s_0 = 0$ means that $X_i^L \leq 1$ for all $i = 1, \ldots, n$. But it actually means that $X_i^L = 1$ for all i because $g(X^L) = K$.

Then the problem is reduced to the following:

Find $X_1^{(2)} \geq \cdots \geq X_{s_1}^{(2)} \geq 1$ $(= X_{s_1+1}^{(2)} = \cdots = X_n^{(2)})$, which yields

$$\min \sum_{i=1}^{s_1} \frac{C_i}{X_i^2} + \sum_{i=s_1+1}^{n} C_i, \quad \text{or simply} \quad \min \sum_{i=1}^{s_1} \frac{C_i}{X_i^2}$$

subject to the constraints

1. $\sum_{i=1}^{s_1} \frac{X_i C_i}{T_i} = K^{(2)}$, where

$$K^{(2)} := K - \sum_{i=s_1+1}^{n} \frac{C_i}{T_i}$$

and obviously $\sum_{i=1}^{s_1} \frac{C_i}{T_i} \leq K^{(2)}$ (we may set $K^{(1)} := K$ in Step 1).

2. $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, s_1$.

Step 2: Apply the method of Lagrange Multiplier to yield

$$X_i^{(2)} = \frac{T_i^{1/3} K^{(2)}}{\sum_{j=1}^{s_1} T_j^{1/3} \frac{C_j}{T_j}}, \qquad i = 1, \ldots, s_1 \tag{4.18}$$

so that $X_1^{(2)} \geq \cdots \geq X_{s_1}^{(2)}$. Then consider the cases:

(a) if $X_{s_1}^{(2)} \geq 1$, then $\hat{X} = (X_1^{(2)}, \ldots, X_{s_1}^{(2)}, 1, \ldots, 1)$.

(b) if $X_{s_1}^{(2)} < 1$, i.e., there is $0 \leq s_2 \leq s_1 - 1$ such that $X_1^{(2)} \geq \cdots \geq X_{s_2}^{(2)} > 1 \geq X_{s_2+1}^{(2)} \geq \cdots \geq X_{s_1}^{(2)}$, then the problem is reduced to the following problem:

Find $X_1^{(3)} \geq \cdots \geq X_{s_2}^{(3)} \geq 1$ $(= X_{s_2+1}^{(3)} = \cdots = X_{s_1}^{(3)} = \cdots = X_n^{(3)})$ with $s_2 < s_1$, which yields

$$\min \sum_{i=1}^{s_2} \frac{C_i}{X_i^2} + \sum_{i=s_2+1}^{n} C_i, \quad \text{or simply} \quad \min \sum_{i=1}^{s_2} \frac{C_i}{X_i^2}$$

subject to the constraints

1. $\sum_{i=1}^{s_2} \frac{X_i C_i}{T_i} = K^{(3)}$, where

$$K^{(3)} := K - \sum_{i=s_2+1}^{n} \frac{C_i}{T_i}$$

   and obviously $\sum_{i=1}^{s_2} \frac{C_i}{T_i} \leq K^{(3)}$.

2. $1 \leq X_i \leq \frac{T_i}{C_i}$, $i = 1, \ldots, s_2$.

Step 3: ......

The process continues as long as some of iterated X's is less than 1 and certainly will stop since there are finitely many X's. The process stops at the kth step if all $X_1^{(k)} \geq \cdots \geq X_{s_k}^{(k)} \geq 1$ and $\hat{X} = (X_1^{(k)}, \ldots, X_{s_k}^{(k)}, 1 \ldots, 1)$.

The following describe a general property among consecutive iterations. The proof is similar to that of (4.13) and is omitted. It implies that in general one iteration is not enough in order to determine $\hat{X}$ and $r-1$.

Proposition 4.8. Suppose $T_1 \geq \cdots \geq T_n$. For each $1 \leq i \leq s_\ell$ ($X_i^{(\ell)} > 1$), $X_i^{(\ell+1)} < X_i^{(\ell)}$. In other words, each needed iteration will decrease X's which are larger than 1.

Example 4.9. For instance, take a look at the following experimental example. Suppose

$$T := (T_1, T_2, T_3, T_4) = (25391, 14905, 12913, 5758)$$

and

$$C := (C_1, C_2, C_3, C_4) = (4616, 6073, 575, 515).$$

Notice that $T_1 \geq T_2 \geq T_3 \geq T_4$.

ith iteration	$X_1^{(i)}$	$X_2^{(i)}$	$X_3^{(i)}$	$X_4^{(i)}$
1	1.23	1.03	0.99	0.75
2	1.19	0.99	1	1
3	1.18	1	1	1

So three iterations are needed to get $\hat{X} = (1.18, 1, 1, 1)$ and $r - 1 = 1$.

Remark 4.10. In case Theorem 4.6(2) occurs,

$$Y := (X_1^L, \ldots, X_{s_1}^L, 1, \ldots, 1) \notin R$$

since $\sum_{i=1}^n \frac{Y_i C_i}{T_i} > \sum_{i=1}^n \frac{X_i^L C_i}{T_i} = K$ and because of (4.4). So Y is not a solution to the minimization problem (4.1). Thus there is no contradiction though $f(Y) < f(X^L)(\leq \min_{X \in R} f(X))$. Similar conclusion can be reached for consecutive iterations with respect to Proposition 4.8.

The following is another description from a top-down view. Similar algorithm can be derived from it.

Theorem 4.11. Let $T_1 \geq \cdots \geq T_n$. Suppose that the minimization problem (4.5) has solution $\hat{X} = (\hat{X}_1, \ldots, \hat{X}_n) \in R$.

1. $X_n^L \geq 1$ if and only if $\hat{X}_n \geq 1$. In this case, $\hat{X} = X^L$ is the unique solution to the minimization problem (4.1).

2. If $X_n^L < 1$, then the unique solution $\hat{X}_1 \geq \cdots \geq \hat{X}_{r-1} > \hat{X}_r = \cdots = \hat{X}_n = 1$ has r where $r-1$ is the first integer i such that

$$\frac{T_i^{1/3} K_{(i)}}{\sum_{j=1}^{i} T_j^{1/3} \frac{C_j}{T_j}} > 1 \quad \text{and} \quad \frac{T_{i+1}^{1/3} K_{(i+1)}}{\sum_{j=1}^{i+1} T_j^{1/3} \frac{C_j}{T_j}} \leq 1,$$

where $K_{(k)} := K - \sum_{j=k+1}^{n} \frac{C_j}{T_j}$.

To illustrate, let us consider the special case $n = 2$.

1. $n = 2$: We first arrange T's so that $T_1 \geq T_2$.

   The Lagrange Multiplier yields $X_i^L = \frac{T_i^{1/3} K}{\sum_{j=1}^{2} T_j^{1/3} \frac{C_j}{T_j}}$, $i = 1, 2$. Notice that

$$X_1^L = \frac{T_1^{1/3} K}{T_1^{1/3}\frac{C_1}{T_1} + T_2^{1/3}\frac{C_2}{T_2}} = \frac{K}{\frac{C_1}{T_1} + (\frac{T_2}{T_1})^{1/3}\frac{C_2}{T_2}} \leq \frac{KT_1}{C_1} \leq \frac{T_1}{C_1}, \tag{4.19}$$

$$X_2^L = \frac{T_2^{1/3} K}{T_1^{1/3}\frac{C_1}{T_1} + T_2^{1/3}\frac{C_2}{T_2}} = \frac{K}{(\frac{T_1}{T_2})^{1/3}\frac{C_1}{T_1} + \frac{C_2}{T_2}} \leq \frac{KT_2}{C_2} \leq \frac{T_2}{C_2} \tag{4.20}$$

$$X_1^L = \frac{K}{\frac{C_1}{T_1} + (\frac{T_2}{T_1})^{1/3}\frac{C_2}{T_2}} \geq \frac{K}{\frac{C_1}{T_1} + \frac{C_2}{T_2}} \geq 1. \tag{4.21}$$

   by $K \geq \frac{C_1}{T_1} + \frac{C_2}{T_2}$. Moreover

$$X_1^L \geq X_2^L \tag{4.22}$$

   because $T_1 \geq T_2$.

   So it remains to check whether $X_2^L \geq 1$.

We take another approach and treat f as a function of X_2. Since $\frac{C_1X_1}{T_1}+\frac{C_2X_2}{T_2}=K$, $X_2=(K-\frac{C_1X_1}{T_1})\frac{T_2}{C_2}$ so that

$$f(X)=f(X_2)=\frac{C_1}{[(K-\frac{C_2X_2}{T_2})\frac{T_1}{C_1}]^2}+\frac{C_2}{X_2^2}.$$

There are two singularities on $\mathbb{R}$, namely $X_2=0$, i.e., $(X_1,X_2)=(\frac{KT_1}{C_1},0)$, and $X_2=\frac{KT_2}{C_2}$, i.e, $(X_1,X_2)=(0,\frac{KT_2}{C_2})$. Notice that $[1,(K-\frac{C_1}{T_1})\frac{T_2}{C_2}]$ is the range for X_2, according to the restrictions.

Consider $f(X_2)$:

1. On the open interval $(-\infty,0)$, f is decreasing from ∞ to 0.

2. On the open interval $(0,\frac{KT_2}{C_2})$, f has a minimum at

$$X_2^L=\frac{K}{(\frac{T_1}{T_2})^{1/3}\frac{C_1}{T_1}+\frac{C_2}{T_2}}\le\frac{KT_2}{C_2}\le\frac{T_2}{C_2}$$

i.e.,

$$(X_1^L,X_2^L)=(\frac{K}{\frac{C_1}{T_1}+(\frac{T_2}{T_1})^{1/3}\frac{C_2}{T_2}},\frac{K}{(\frac{T_1}{T_2})^{1/3}\frac{C_1}{T_1}+\frac{C_2}{T_2}})$$

by the method of Lagrange Multiplier.

3. On the open interval $(\frac{KT_2}{C_2},\infty)$, f is decreasing from ∞ to 0.

Notice that the interval $[1,(K-\frac{C_1}{T_1})\frac{T_2}{C_2}]$ is a subset of the open interval $(0,\frac{KT_2}{C_2})$. Recall $1\le X_1^L\le\frac{T_1}{C_1}$, $X_2^L\le\frac{T_2}{C_2}$, $\frac{C_1X_1}{T_1}+\frac{C_2X_2}{T_2}=K$, and $X_1^L\ge X_2^L$. So we only have the following two cases:

Case 1: $X_2^L \geq 1$. Then (X_1^L, X_2^L) provides the min.

Note: Since $\frac{C_1 X_1^L}{T_1} + \frac{C_2 X_2^L}{T_2} = K$, $X_1^L \geq 1$ implies $\frac{C_2 X_2^L}{T_2} = K - \frac{C_1 X_1^L}{T_1} \leq K - \frac{C_1}{T_1}$ so that

$$X_2^L \leq (K - \frac{C_1}{T_1})\frac{T_2}{C_2}.$$

Similarly if $X_2^L \geq 1$, then

$$X_1^L \leq (K - \frac{C_2}{T_2})\frac{T_1}{C_1}.$$

Case 2: $X_2^L < 1$. The minimum must be obtained on the boundary ∂R. Now

$$\begin{aligned} \frac{d}{dX_2} f(X_2) &= \frac{2C_1 \frac{C_2}{T_2}}{(K - \frac{C_2 X_2}{T_2})^3 (\frac{T_1}{C_1})^2} - \frac{2C_2}{X_2^3} \\ &= \frac{2C_2}{(K - \frac{C_2 X_2}{T_2})^3 X_2^3} \left[\frac{C_1^3}{T_1^2 T_2} - (K - \frac{C_2 X_2}{T_2})^3 \right] \end{aligned}$$

Recall that the range for X_2 is $[1, (K - \frac{C_1}{T_1})\frac{T_2}{C_2}]$. For $1 \leq X_2 < \frac{T_2}{C_2}$, since $X_2^L < 1$, i.e.,

$$\frac{C_1}{T_1}\left(\frac{T_1}{T_2}\right)^{1/3} + \frac{C_2}{T_2} > K$$

so that

$$\frac{C_1}{T_1}\left(\frac{T_1}{T_2}\right)^{1/3} + \frac{C_2 X_2}{T_2} \geq \frac{C_1}{T_1}\left(\frac{T_1}{T_2}\right)^{1/3} + \frac{C_2}{T_2} > K.$$

Hence

$$\frac{C_1^3}{T_1^2 T_2} - (K - \frac{C_2 X_2}{T_2})^3 > 0$$

so that $\frac{d}{dX_2} f(X_2) > 0$. In other words, f is increasing on the interval. Hence the minimum occurs at $X_2 = 1$.

The following is a typical picture for $n = 2$ case.

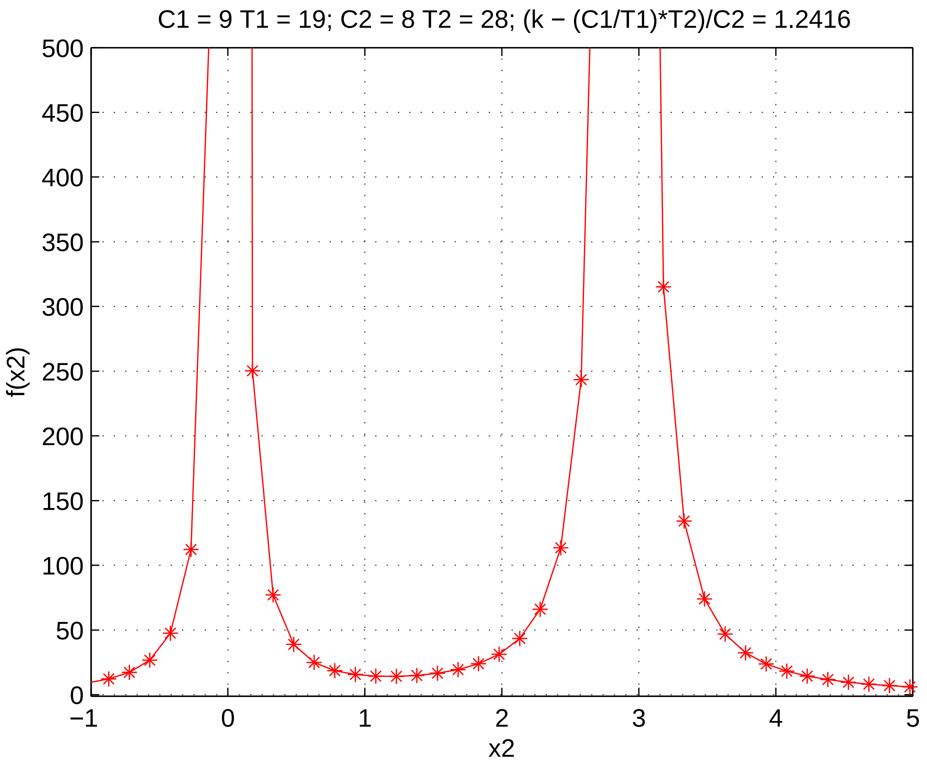

Figure 4.1: Plot of $f(X_2)$

4.2 Scaling Factor Assignment Algorithm

On the basis of above proofs, we restate our solution as follows:

1. Sort time period T_i in descending order.

2. Compute scaling factor X_i by the method of Lagrange multiplier ($X_1 \geq \cdots \geq X_n$). If all of them are greater than or equal to 1, then they are the solution. Then we are done. Otherwise, there are some X's less than 1.

3. Then find those X_i less than or equal to 1 and set them to 1, i.e., $X_r = 1$, $r = i, i+1, \ldots, n$.

4. Repeat 2 and 3 until all $X_i \geq 1$.

Table 4.1 is the ScaleFactors pseudocode according to the above algorithm.

```
1   Procedure ScaleFactors
2   Input: task set $t = \{< C_i, T_i >\}$ of $n$ elements
3   Output: Scale factor set $\{X_i\}$ of $n$ elements

4   Sort $t$ in descending $T_i$
5   $K \leftarrow n(2^{1/n} - 1)$
6   $r \leftarrow n + 1$
7   repeat
8       $denom \leftarrow 0, K' \leftarrow K$
9       for $j \leftarrow 0$ to $r - 1$ do $denom \leftarrow denom + T_j^{1/3} * C_j/T_j$
10      for $q \leftarrow r$ to $n$ do $K' \leftarrow K' - C_q/T_q$
11      $trunc \leftarrow$ false
12      for $i \leftarrow 1$ to $r - 1$ do
13          $X_i \leftarrow T_i^{1/3} * K'/denom$
14          if $(X_i \leq 1)$ then
15              $r \leftarrow i + 1, trunc \leftarrow$ true
16              break
17          end if
18      end for
19  until (not $trunc$) or $(r = 1)$
20  for $i \leftarrow r$ to $n$ do $X_i \leftarrow 1$
21  end ScaleFactors
```

Table 4.1: Procedure ScaleFactors Pseudocode

Let us analyze the ScaleFactors algorithm.

- Sorting task set t of n elements can take *O(*$n \lg n$*)* time if we use Quicksort algorithm.

- Initializing variable K and r at the beginning takes *O(*1*)* time, respectively.

- There are three nested loops. The body of the first and second inner loops, controlled by counter j and q, are executed together n times, for $j = 0, \dots, r-1$ and $q = r, \dots, n$; that is, *O(n)* times.

- Similarly, the body of the third inner loop, which is controlled by counter i, is executed $r - 1$ times, depending on the current value of the variable r. Since r is initialized with $n + 1$, and is never greater than $n + 1$, which implies that in each iteration, the body of the for loop executes $n+1$ times at most. Therefore, the statements in the second inner loop also takes *O(n)* times.

- The outer loop is not terminated until the variable *trunc* is ***false*** or r is set to be 1. The best case occurs when the variable *trunc* never be assigned ***true*** or $r = 1$, which takes *O(1)* time. However, the worst case occurs when $r = n - 1$ in the prior iteration each time and takes *O(n)* times.

In sum, the running time of ScaleFactors algorithm is given $O(n \lg n) + O(1) + O(n^2) = O(n^2)$.

However, since our goal is to find $(\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_{r-1})$ for some $r = 1, \ldots, n$, if we can get the first X less than 1 sooner in each iteration, we can improve the complexity of the algorithm. Using divide-and-conquer paradigm in Table 4.2 to arrive at $\hat{X}_{r-1}$ takes $O(n \lg n)$ run-time and the complexity will be reduced to *O(*$n \lg n$*)*.

```
Procedure ScaleFactorsDnC
Input: task set $t = \{< C_i, T_i >\}$ of $n$ elements
Output: Scale factor set $\{X_i\}$ of $n$ elements

Sort $t$ in descending $T_i$
$K \leftarrow n(2^{1/n} - 1)$
$first \leftarrow 0, last \leftarrow n$
while $(first <= last)$ do
    $r \leftarrow last$
    $mid \leftarrow (first + last)/2$
    $denom \leftarrow 0, K' \leftarrow K$
    for $j \leftarrow 0$ to $r - 1$ do $denom \leftarrow denom + T_j^{1/3} * C_j/T_j$
    for $q \leftarrow r$ to $n$ do $K' \leftarrow K' - C_q/T_q$
    $X_{mid} \leftarrow T_{mid}^{1/3} * K'/denom$
    if $(X_{mid} > 1)$ then
        $first \leftarrow mid + 1$
    else
        $last \leftarrow mid - 1$
    end if
end while
for $i \leftarrow 1$ to $r - 1$ do $X_i \leftarrow T_i^{1/3} * K'/denom$
for $i \leftarrow r$ to $n$ do $X_i \leftarrow 1$
end ScaleFactorsDnC
```

Table 4.2: Procedure ScaleFactorsDnC Pseudocode

4.3 Revised Scaling Factor Assignment Algorithm

If we look back into the mathematical form of our solution from (4.11), we would find out that $\hat{X}_i > 1$ only when $i \leq r-1$. In other words, $\hat{X}_i \leq 1$ when $i > r-1$ according to Proposition 4.7. Therefore, we can also get $\hat{X}_{r-1}$ more efficiently if we compute X's backwards. We adjust above algorithm and list its pseudocode in Table 4.3.

```
Procedure ScaleFactorsBackward
Input: task set $t = < C_i, T_i >$ of $n$ elements
Output: Scale factor set $X_i$ of $n$ elements

Sort time periods in descending order.
$K \leftarrow n(2^{1/n} - 1), denom \leftarrow 0$
for i $\leftarrow$ 1 to $n$ do
    $denom \leftarrow denom + T_i^{1/3} * C_i/T_i$
end for
for i $\leftarrow$ $n$ down to 1 do
    $X_i = T_i^{1/3} * K/denom$
    if $X_i \leq 1$ then
        $X_i \leftarrow 1$
        $K \leftarrow K - C_i/T_i$
        $denom \leftarrow denom - T_i^{1/3} * C_i/T_i$
        $r \leftarrow i$
    end if
end for
end ScaleFactorsBackward
```

Table 4.3: Procedure ScaleFactorsBackward Pseudocode

- As well, sorting task set t of n elements can take $O(n \lg n)$ time if we use Quicksort algorithm.

- Initializing variables K and *denom* takes $O(1)$ time.

- The body of the first inner loop, controlled by counter i, is executed n times, for $i = 0, \ldots, n-1$, that is, $O(n)$ times.

- Similarly, the body of the second inner loop, which is controlled by counter i, is executed n times, for $i = n-1, \ldots, 0$. Therefore, the statements in the second inner loop takes $O(n)$ times.

The ScaleFactorsBackwards then takes $O(n \lg n) + O(1) + O(n) + O(n) = O(n \lg n)$ times.

If we only consider the complexity of computing $\hat{X}_i$, the ScaleFactorsBackwards takes only $O(n)$ times to arrive at $\hat{X}_r$, yet the ScaleFactors takes $O(n^2)$ and Scale-FactorsDnC takes $O(n \lg n)$ times. Therefore, the ScaleFactorsBackwards algorithm is proven to have lowest complexity than the other two.

Chapter 5

Result and Analysis

Consider the following task set:

Task	Computation Time, C	Time Period, T	Utilization, U	Priority
a	3	8	0.38	3
b	3	10	0.30	2
c	1	14	0.07	1

Table 5.1: Example Task Set A

Table 5.1 contains three tasks that are given priorities through RM algorithm. Computation time and time period are measured in the number of clock cycles per second. Note that priority is presented in integer in descending order. Namely, the higher the integer, the greater the priority.

Task a is given the highest priority since it has the shortest time period. Their combined CPU utilization is 0.75 and passes RM test, which equals 0.78 from (3.1).

Figure 5.1 is the time-line for task set A before scaling. Y-coordinate represents clock frequency of the processor, and x-coordinate means task computation time. a_2 is the deadline of task a; likewise, b_2, c_2 are the deadlines of tasks b and c. Notice that all the three tasks finish their execution before their deadlines.

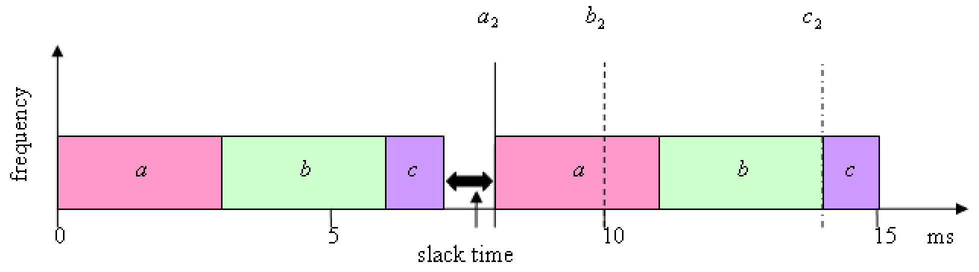

Figure 5.1: Time-line for Task Set A before scaling, where computation times are specified at the maximum CPU frequency

After scaling all tasks computation time according to each corresponding scaling factor from our solution, table 5.2 and table 5.3 show the results of energy reduction.

Task	Computation Time, C	Utilization, U	CPU Frequency	Energy
a	3	0.38	1.0	
b	3	0.30	1.0	7
c	1	0.07	1.0	

Table 5.2: Energy Consumption before Scaling for Task Set A

We suppose the maximum CPU frequency is 1.0. Energy in Tables 5.2 and 5.3 are calculated from (4.1) $\times$ CPU frequency.

Task	Computation Time, C	Utilization, U	CPU Frequency, f	Minimum Energy
a	3	0.38	1	
b	3.21	0.32	0.94	6.35
c	1.19	0.08	0.84	

Table 5.3: Energy Consumption after Scaling for Task Set A

As we see in Table 5.3, the CPU frequency and computation time of each task change individually. For instance, CPU frequency and computation time of task a remain the same after scaling, yet in task b, CPU frequency is slightly reduced to 0.94 and its computation time extends from 3 till 3.21. The CPU frequency of task c is reduced even more. This phenomenon just illustrates the algorithm suggested in Chapter 4, the longer the time period of a task, the greater its scaling factor.

The combined CPU utilization after scaling just reaches 0.78 (the upper bound of RM test), but the scaled task set saves 17.79% energy.

The scaled time-line is depicted in Figure 5.2. Note that not only both CPU frequencies of tasks *b* and *c* are reduced, but slack time is also shortened. However, each task still complete its execution before deadline. Therefore, we prove that the task set can still be scheduled by RM algorithm.

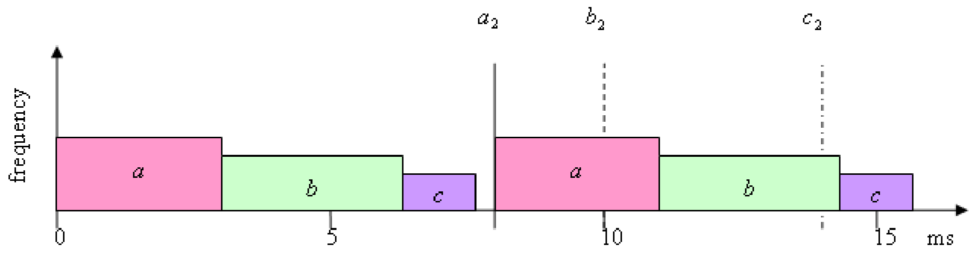

Figure 5.2: Scaled Time-line for Task Set A

Consider another case:

Task	Computation Time, C	Time Period, T	Utilization, U	Priority
a	2	14	0.14	1
b	1	10	0.10	3
c	3	12	0.25	2

Table 5.4: Example Task Set B

Task *b* in table 5.4 is given the highest priority and task *a* the lowest. Their combined CPU utilization is 0.49, much less than 0.78.

The time-lime scheduled by RM algorithm for task set B is as follows:

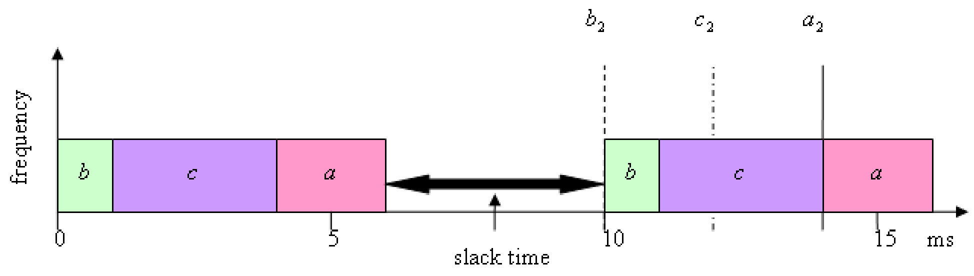

Figure 5.3: Time-line for Task Set B

The slack time in task set B is much longer than in task set A, since its combined CPU utilization is much less than in task set A. Task *b* has the greatest priority and executes first. Again, we assume that every task executes at the maximum CPU frequency 1.0.

Table 5.5 and Table 5.6 show the result of power reduction after scaling. The scaled task set saves almost 60.22 % energy.

Task	Computation Time, C	Utilization, U	CPU Frequency, f	Energy
a	2	0.14	1.0	
b	1	0.10	1.0	6
c	3	0.25	1.0	

Table 5.5: Energy Consumption before Scaling for Task Set B

Task	Computation Time, C	Utilization, U	CPU Frequency, f	Minimum Energy
a	3.32	0.24	0.60	
b	1.58	0.16	0.67	2.39
c	4.45	0.37	0.63	

Table 5.6: Energy Consumption after Scaling for Task Set B

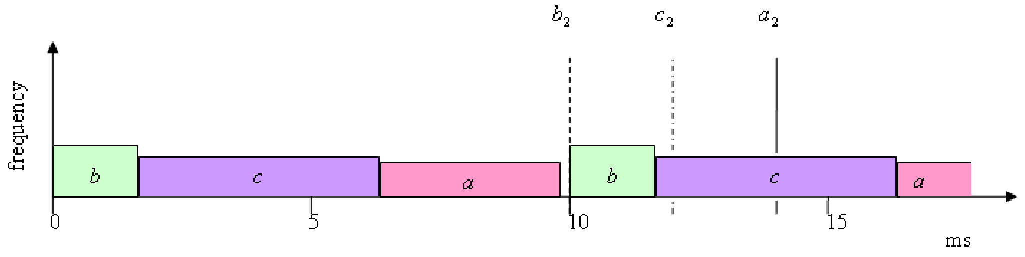

Figure 5.4: Scaled Time-line for Task Set B

The time-line for task set B after CPU frequency scaling is in Figure 5.4.

The results show that the percentage of power saving varies dramatically because this approach is independent of task sets. In general, the more difference between CPU combined utilization and the upper bound of RM test, the more power reduction.

We should also consider the effect of the overhead of voltage switching because voltage switching results in overhead in that CPU cannot execute any task [15]. We could further incorporate our approach with different dynamic reclaiming algorithms. We will extend this work in the real life examples as our future work.

Chapter 6

Summary

In this thesis, we derive a minimum energy function for RM task sets and solve this problem. Different from other approaches, our solution goes straightforward and takes account of RM test united with power dissipation equation in CMOS circuits. By using the method of Lagrange Multiplier, we are able to abtain each scaling factor for corresponding task execution time.

We assume the processor can vary its supply voltage dynamically and ignore the threshold voltage and voltage switching overhead. We also assume that every slack time can be fully filled by extending each task's WCET according to their specified scaling factors.

For a task set containing n tasks, the scaling factors $\hat{X}_i$ can be obtained by iteratively using the following formula to test whether the scaling factor $\hat{X}_i$ is less than or equal to 1,

$$\hat{X}_i = \frac{T_i^{1/3} K'}{\sum_{j=1}^{r-1} T_j^{1/3} \frac{C_j}{T_j}}, \qquad i = 1, \ldots, r-1, \qquad \text{for some } r = 1, \ldots, n$$

$$K' := n(2^{1/n} - 1) - \sum_{i=r}^{n} \frac{C_i}{T_i}.$$

The experiment results show that minimum power consumption can be exactly accomplished in the presence of all $\hat{X}_i$ greater than or equal to 1. At the same time, the RM task set is still guaranteed to meet all deadlines. Our solution is proven not to violate tasks deadlines.

Bibliography

[1] A. Burns and A. Wellings, *Real Time Systems and Programming Languages: Ada 95, Real-Time Java and Real-Time C/POSIX*, 3rd ed., Addison Wesley, 2001.

[2] A. P. Chandrakasan, S. Sheng and R. W. Brodersen. "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 473-484, Apr. 1992.

[3] A. Chilambuchelvan, S. Saravanan and J. R. P. Perinbam. "A Simulation Software Development for Performance Analysis of DVS Algorithm for Low Power Embedded System," *ARPN Journal of Engineering and Applied Sciences*, vol. 2, pp. 27-31, Aug. 2007.

[4] F. Gruian, "Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors," In Proc. the 2001 International Symposium on Low Power Electronics and Design, 2001, pp. 46-51.

[5] A. Hakan, R. Melhern, D. Mosse, P. M. Alvarez, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," *Real-Time Systems*, pp. 225-232, Jun. 2001.

[6] R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[7] W. Kim, D. Shin, H. S. Yun, J. Kim, and S. L. Min, "Performance Comparison of Dynamic Voltage Scaling Algorithms for Hard Real-Time Systems," In Proc. the 8th IEEE Real-Time and Embedded Technology and Applications Symposium, 2002, pp. 219-218.

[8] Y. H. Tsai, K. Wang, and J. M. Chen, "A Deferred-Workload-based Inter-Task Dynamic Voltage Scaling Algorithm for Portable Multimedia Devices," In Proc. the 2007 international conference on Wireless communications and mobile computing, 2007, pp. 677-682.

[9] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of the ACM (JACM), vol. 20, pp. 46-61, Jan. 1973.

[10] Y. Liu and A. K. Mok, “An Integrated Approach for Applying Dynamic Voltage Scaling to Hard Real-Time Systems,” In Proc. the 9th IEEE Real-Time and Embedded Technology and Applications Symposium, 2003, pp. 116-123.

[11] A. Manzak and C. Chakrabarti, “Variable Voltage Task Scheduling Algorithms for Minimizing Energy/Power,” In Proc. the 2001 International Symposium on Low Power Electronics and Design, 2001, pp.279-282.

[12] P. Pillai and K. G. Shin, “Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating System,” In Proc. the 18th ACM Symposium on Operating Systems Principles, 2001, pp. 89-102.

[13] D. Shin, J. Kim and S. Lee. “Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications,” IEEE Design and Test of Computers, vol. 18, pp. 20-30, Mar. 2001

[14] Y. Shin and K. Choi, “Power Conscious Fixed Priority Scheduling for Hard Real-Time Systems,” In Proc. the 36th ACM/IEEE Conference on Design Automation, 1999, pp. 134-139.

[15] V. Swaminathan and K. Chakrabarty, “Investigating the Effect of Voltage-Switching on Low-Energy Task Scheduling in Hard Real-Time Systems,” In Proc. the 2001 Conference on Asia South Pacific Design Automation, 2001, pp. 251 - 254.

[16] O. S. Unsal and I. Koren, “System-Level Power-Aware Design Techniques in Real-Time Systems,” In Proc. the IEEE, 2003, pp. 1055 - 1069.

[17] M. Weiser, B. Welch, A. Demers, and S. Shenker, ”Scheduling for Reduced CPU Energy,” In Proc. the 1st USENIX conference on Operating Systems Design and Implementation, 1994, pp. 13-23.

[18] X. L. Zhong and C. Z. Xu, “Energy-Aware Modeling and Scheduling of Real-Time Tasks for Dynamic Voltage Scaling,” In Proc. the 26th IEEE International Real-Time Systems Symposium, 2005.

Appendices

```
function y = MinEnergy(T,C)
. . .
    [T,NDX] = sort(T,'descend'); % sort T in descending order
    C = C(NDX); % sort C accordingly but not necessarily in descending order
    unsort(NDX) = 1:n; % see http://blogs.mathworks.com/loren/?p=104
        % inverse sort
    K = n*(2^(1/n)−1); % upper bound of RM
    C1 = C;
    T1 = T;
    i = 1;
    while i <= n
        D = sum((T1.^(1/3)).*(C1./T1)); % Denominator
        X(i) = (T1(i))^(1/3)*K/D;
        if X(i) <= 1
            X(i) = 1;
            K = K − sum((C1(i:n))./(T1(i:n)));
            T1 = T1(1:i-1); C1 = C1(1:i-1);
            n = i - 1; i = 1;
        else
            i = i + 1;
        end
    end
    unsort;
    X=X(unsort) % print the real X
    end
end
```