J. Parallel Distrib. Comput. 95 (2016) 15–28

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels

Keqin Li

Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

HIGHLIGHTS

- Address energy and time constrained task scheduling with discrete speed levels.
- Prove the NP-hardness even on a uniprocessor computer with only two speed levels.
- Develop algorithms with two components: task scheduling and speed determination.
- Derive worst-case asymptotic performance bounds and average-case asymptotic per- formance bounds.
- Perform extensive simulations to verify the analytical results.

ARTICLE INFO

Article history:
Received 18 September 2015
Received in revised form
25 December 2015
Accepted 26 February 2016
Available online 4 March 2016

Keywords:
Discrete speed levels
Energy consumption
List scheduling
List placement
Performance analysis
Power-aware scheduling
Simulation
Task scheduling

ABSTRACT

Energy and time constrained task scheduling on multiprocessor computers with discrete clock frequency and supply voltage and execution speed and power levels is addressed as combinatorial optimization problems. It is proved that the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint are NP-hard even on a uniprocessor computer with only two speed levels. A class of algorithms is developed to solve the above two problems. These algorithms include two components, namely, a list scheduling algorithm for task scheduling and a list placement algorithm for speed determination. A worst-case asymptotic performance bound and an average-case asymptotic performance bound are derived for our algorithms on uniprocessor computers, and a worst-case asymptotic performance bound is derived for our algorithms on multiprocessor computers. Extensive simulations are performed to verify our analytical results. It is found that our algorithms produce solutions very close to optimal and are practically very useful.

1. Introduction

Modern high-performance processors can consume significant amount of energy. An idle processor (e.g., Intel Core i7-975 3.33 GHz, DDR3-1066, 1 MB L2, 8 MB L3) may require 83 W of power. The peak power consumption can be as high as 210 W (e.g., AMD FX 8350 4.0 GHz, DDR3-1866, 8 MB L2, 8 MB L3) [36]. As the scale of multi-core and many-core systems increases to the order of 10^6 cores, the power consumption also increases dramatically. As of June 2015, the average power of the worlds ten fastest supercomputers is 6.4454 MW [37], costing 56,461,704 kWh of electricity and 6,137,387 USD (assuming 10.87 cents per Kilowatthour [34]) per year. On July 29, 2015, President Obama signed an executive order—creating a National Strategic Computing Initiative with the objective of accelerating delivery of a capable exascale computing system, which is able to perform a quintillion, i.e., 10^{18}, floating point operations per second [38]. Since it is widely believed that power consumption of any computing system should not exceed 20 MW, we are facing the challenge of developing an exascale system with 50,000 MFLOPS/Watt; however, the current (as of June 2015) technology can only achieve 7031.58 MFLOPS/Watt [35]. Therefore, energy efficiency is listed as the number one challenge of the top ten exascale system research challenges [33].

According to Moore's law, power consumption in computer systems has increased at an exponential speed for decades [31]. Power density in high-performance computer systems will soon reach that of a nuclear reactor [79]. Such increased energy consumption causes severe economic, ecological, and technical problems [17,20,25,75]. Power conservation is critical in many computation and communication environments and has attracted extensive research activities. Reducing processor energy

E-mail address: lik@newpaltz.edu.

http://dx.doi.org/10.1016/j.jpdc.2016.02.006
0743-7315/

consumption has been an important and pressing research issue in recent years. There has been increasing interest and importance in developing high-performance and energy-efficient computing systems [16,18,19]. There exists an explosive body of literature on power-aware computing and communication. The reader is referred to [1,8,9,77–79,91] for comprehensive surveys.

Power consumption in computing systems can be reduced by thermal-aware hardware and software design at various levels. Software techniques for power reduction are supported by a mechanism called *dynamic voltage scaling* (equivalently, dynamic frequency scaling, dynamic speed scaling, dynamic power scaling) [28]. A power-aware algorithm can change supply voltage and frequency at appropriate times to optimize a combined consideration of performance and energy consumption. There are many existing technologies and commercial processors that support dynamic voltage (frequency, speed, power) scaling [29,30,32].

Dynamic power management at the operating system level refers to supply voltage and clock frequency adjustment schemes implemented while tasks are running. These energy conservation techniques explore the opportunities for tuning the energy-delay tradeoff [76]. Since the pioneering work in [80,82], power-aware task scheduling on processors with variable voltages and speeds has been extensively studied, including scheduling tasks with arrival times and deadlines on a uniprocessor computer with minimum energy consumption [4–6,12,42,45,57,60,59,83], scheduling independent or precedence constrained tasks on uniprocessor or multiprocessor computers in real-time applications [3,23,26,27,40,44,46,61,63,66,70,71,73,74,81,87–90], dealing with the energy-delay tradeoff [7,11,14,22,43,48,58,72,86,92], developing high-performance and energy-efficient computing systems [10,16,18,19], improving system level power management [15,39,47,62,67], and conducting other studies [2,65,68,85]. In [50,52,54,53,51,55,56], we addressed energy and time constrained power allocation and task scheduling on multiprocessors with dynamically variable voltage and frequency and speed and power as combinatorial optimization problems.

Much existing research assume that a task can be supplied with any power and a processor can be set at any speed, that is, clock frequency and supply voltage and execution speed and power supply can be changed continuously in any range. However, the currently available processors have only a few *discrete* clock frequency and supply voltage and execution speed and power levels [41,69]. Much existing research also assume that clock frequency and supply voltage and execution speed and power supply can be changed in any range. However, discrete settings also imply that clock frequency and supply voltage and execution speed and power supply can only be change in certain *bounded* range. The constraints of discrete and bounded settings certainly make our optimization problems more difficult to solve. However, power-aware task scheduling algorithms developed with such constraints, though more complicated, will be more practically useful.

Task scheduling on processors with discrete speed levels has been investigated by a number of researchers. For instances, it was shown that an optimal preemptive schedule with minimum energy consumption on a uniprocessor computer can be found in polynomial time [45,60,59]. Processors with discrete speed levels were also considered in real-time multiprocessor systems [64,88]. However, to the best of the author's knowledge, energy and time constrained nonpreemptive task scheduling on multiprocessor computers with discrete speed levels has not been well studied analytically, although experimental studies have been conducted by many researchers. The motivation of this paper is to make investigation towards this direction. We find that addressing discrete and bounded speed levels simultaneously does not yield analytically tractable algorithms and manageable and meaningful results. Therefore, we will concentrate on discrete speed levels which are assumed to be enough to accommodate the needs of our algorithms.

The main contributions of the present paper are as follows. First, we prove that the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint are NP-hard even on a uniprocessor computer with only two speed levels (Proposition 1 in Section 3.1 and Proposition 2 in 4.1). Second, we develop a class of algorithms to solve the above two problems. These algorithms include two components, namely, a list scheduling algorithm for task scheduling and a list placement algorithm for speed determination (Sections 3.2 and 4.2). Third, we derive a worst-case asymptotic performance bound and an average-case asymptotic performance bound for our algorithms on uniprocessor computers (Theorem 1 and Corollary 1 in Section 3.1 and Theorem 3 and Corollary 2 in Section 4.1), and a worst-case asymptotic performance bound on multiprocessor computers (Theorem 2 in Section 3.2 and Theorem 4 in Section 4.2). Fourth, we perform extensive simulations to verify our analytical results. We found that our algorithms produce solutions very close to optimal and are practically very useful (Section 5).

2. The model of power consumption

Power dissipation and circuit delay in digital CMOS circuits can be accurately modeled by simple equations, even for complex microprocessor circuits. CMOS circuits have dynamic, static, and short-circuit power dissipation; however, the dominant component in a well designed circuit is dynamic power consumption p (i.e., the switching component of power), which is approximately $p = aCV^2f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency [13]. The supply voltage and the clock frequency are related in such a way that $f \propto V^\gamma$ and equivalently $V \propto f^{1/\gamma}$, where $0 < \gamma \leq 1$ [84]. The processor execution speed s is linearly proportional to the clock frequency, namely, $s \propto f$. For ease of discussion, we will assume that $V = bf^{1/\gamma}$ and $s = cf$, where b and c are some constants.

We consider processors with discrete clock frequency and supply voltage and execution speed and power levels. Assume that a processor has d discrete levels of clock frequency $f_1, f_2, \ldots, f_d$, which result in d discrete levels of supply voltage $V_1, V_2, \ldots, V_d$, and d discrete levels of processor speed $s_1, s_2, \ldots, s_d$, and d discrete levels of power supply $p_1, p_2, \ldots, p_d$, where $V_j = bf_j^{1/\gamma}$, $s_j = cf_j$, and $p_j = aCV_j^2f_j = ab^2Cf_j^{1+2/\gamma} = ab^2Cf_j^\alpha$, where $\alpha = 1 + 2/\gamma \geq 3$, for all $1 \leq j \leq d$.

Assume that we are given n independent tasks to be executed on m identical processors. Let r_i represent the execution requirement (i.e., the number of CPU cycles or the number of instructions) of task i. If task i is executed with processor speed s_j, the execution time of task i is $t_i = r_i/s_j$, and the energy consumed to execute task i is calculated as $e_i = p_jt_i = r_i(p_j/s_j) = r_i(aCV_j^2f_j/s_j) = (aC/c)r_iV_j^2 = (aC/c)r_i(bf_j^{1/\gamma})^2 = (ab^2C/c)r_if_j^{\alpha-1} = (ab^2C/c)r_i(s_j/c)^{\alpha-1} = (ab^2C/c^\alpha)r_is_j^{\alpha-1}$, where $1 \leq i \leq n$ and $1 \leq j \leq d$. Notice that the constants a, b, c, and C only generate effect of scaling. Thus, we will simply assume that $e_i = r_is_j^{\alpha-1}$.

Throughout the paper, we assume that $s_1 < s_2 < \cdots < s_d$, where $d \geq 2$. We also define $\phi_j = s_{j+1}/s_j$, for all $1 \leq j \leq d-1$. The quantity ϕ_j measures the relative gap between two successive speed levels s_j and s_{j+1}. As mentioned earlier, we assume that the number of speed levels is large enough to accommodate the needs of our algorithms.

3. Energy constrained scheduling

Given n tasks with task execution requirements $r_1, r_2, \ldots, r_n$, the problem of *minimizing schedule length with energy consumption constraint* E on a multiprocessor computer with m processors is to determine the execution speeds $s_{j_1}, s_{j_2}, \ldots, s_{j_n}$ and a nonpreemptive schedule of the n tasks on the m processors such that the schedule length is minimized and the total energy consumed does not exceed E.

3.1. Scheduling on uniprocessor computers

We first consider a uniprocessor computer with $m = 1$ processor. It is clear that on a uniprocessor computer with energy constraint E, the problem of minimizing schedule length with energy consumption constraint is simply to decide the execution speeds $s_{j_1}, s_{j_2}, \ldots, s_{j_n}$, such that the schedule length

$$t_1 + t_2 + \cdots + t_n = \frac{r_1}{s_{j_1}} + \frac{r_2}{s_{j_2}} + \cdots + \frac{r_n}{s_{j_n}}$$

is minimized and the total energy consumed does not exceed E, i.e.,

$$e_1 + e_2 + \cdots + e_n = r_1 s_{j_1}^{\alpha-1} + r_2 s_{j_2}^{\alpha-1} + \cdots + r_n s_{j_n}^{\alpha-1} \le E.$$

Since tasks can be executed in any order on one processor, there is no issue of scheduling. The determination of the execution speeds is the only problem.

Let $R = r_1 + r_2 + \cdots + r_n$ be the total execution requirement of the n tasks. We define $E_j = Rs_j^{\alpha-1}$, which is the amount of energy required for all the tasks to be executed at speed s_j, where $1 \le j \le d$. It is reasonable to assume that the given energy budget E is in the range $E_1 \le E < E_d$. If $E < E_1$, there is no feasible schedule to meet the energy constraint. If $E \ge E_d$, we simply execute all tasks at speed s_d and achieve the shortest possible schedule length.

Our first result shows that even on a uniprocessor computer with only $d = 2$ speed levels, the problem of minimizing schedule length with energy consumption constraint is NP-hard.

Proposition 1. *The problem of minimizing schedule length with energy consumption constraint is NP-hard on a uniprocessor computer with two speed levels.*

Proof. Consider a uniprocessor computer with $d = 2$ speed levels s_1 and s_2. Assume that the energy constraint E is $E_1 \le E < E_2$. It is clear that some tasks are executed at speed s_1, while others are executed at speed s_2. Let $S \subseteq \{1, 2, \ldots, n\}$ be the set of tasks which are executed at speed s_2, and

$$R' = \sum_{i \in S} r_i$$

be the total execution requirement of tasks in S. The schedule length is

$$\sum_{i \notin S} \frac{r_i}{s_1} + \sum_{i \in S} \frac{r_i}{s_2} = \frac{R - R'}{s_1} + \frac{R'}{s_2} = \frac{R}{s_1} - R'\left(\frac{1}{s_1} - \frac{1}{s_2}\right).$$

The actual amount of energy consumed is

$$\begin{aligned}\sum_{i \notin S} r_i s_1^{\alpha-1} + \sum_{i \in S} r_i s_2^{\alpha-1} &= (R - R')s_1^{\alpha-1} + R's_2^{\alpha-1} \\ &= Rs_1^{\alpha-1} + R'(s_2^{\alpha-1} - s_1^{\alpha-1}) \\ &= E_1 + R'(s_2^{\alpha-1} - s_1^{\alpha-1}).\end{aligned}$$

Our problem of determination of the execution speeds is equivalent to the problem of choosing S such that

$$E_1 + R'(s_2^{\alpha-1} - s_1^{\alpha-1}) \le E,$$

that is,

$$R' \le \frac{E - E_1}{s_2^{\alpha-1} - s_1^{\alpha-1}} = M,$$

and that R' is maximized so that the schedule length is minimized.

The above problem is exactly the *maximum subset sum* problem [49]. Assume that there are n objects (i.e., tasks) of sizes $r_1, r_2, \ldots, r_n$ and a bag of capacity M. The maximum subset problem is to find a subset S of objects to be placed into the bag such that the total size of the objects packed into the bag is as large as possible but does not exceed the capacity M of the bag. Notice that the solution of the maximum subset problem can be used to decide whether there is a subset of objects whose total size is exactly M, i.e., the *subset sum* problem, which is known to be NP-hard [21, p. 223]. Therefore, the maximum subset sum problem is NP-hard, so is our problem of minimizing schedule length with energy consumption constraint. ■

Our strategy for energy constrained task scheduling on processors with discrete clock frequency and supply voltage and execution speed and power levels is to simulate the efficient algorithms on processors with continuous clock frequency and supply voltage and execution speed and power levels. It has been known from [50] that if processors have continuous clock frequency and supply voltage and execution speed and power levels, the schedule length is minimized when all tasks are executed at the same speed $s = (E/R)^{1/(\alpha-1)}$. It is clear that such an optimal speed s is not available in our d discrete speed levels, unless $E = E_j$ for some $1 \le j \le d$. Assume that $E_j \le E < E_{j+1}$, where $1 \le j \le d - 1$. This implies that $s_j \le s < s_{j+1}$. Our strategy is to achieve close-to-optimal performance by using speed levels s_j and s_{j+1}. Hence, although there are d speed levels available, we only focus on two speed levels.

Our scheduling algorithm works as follows. All the n tasks are executed at speed s_j or s_{j+1}. If all tasks are executed at speed s_j, the energy consumption is E_j, and there is extra energy $E - E_j$ to accommodate some tasks to be executed at speed s_{j+1}. We choose a subset $S \subseteq \{1, 2, \ldots, n\}$ of tasks that are executed at speed s_{j+1}. Tasks not in S are executed at speed s_j. Let

$$R' = \sum_{i \in S} r_i.$$

The subset S of tasks are chosen such that R' is as large as possible under the condition that

$$R' \le M = \frac{R\Delta}{\phi_j^{\alpha-1} - 1},$$

where Δ is defined such that $E = E_j(1 + \Delta) = Rs_j^{\alpha-1}(1 + \Delta)$, with $0 \le \Delta < \phi_j^{\alpha-1} - 1$. The bound M will be explained shortly.

The problem of finding S is equivalent to the maximum subset sum problem, where the n objects are the n tasks with sizes $r_1, r_2, \ldots, r_n$, and the bag capacity M is the space translated from the extra energy $E - E_j$ to accommodate tasks to be executed at speed s_{j+1}. The problem can be solved by using a simple greedy algorithm called *list placement* (LP) [49] which works as follows. Initially, the available space of the bag is M. We scan the list of objects one after another. For each object i, we put the object into the bag if r_i is no greater than the currently available space of the bag. After object i is packed into the bag, the available space of the bag is reduced by r_i.

Algorithm LP arranges tasks in a random order. The LP algorithm has many variations, depending on the strategy used in the initial ordering of the tasks. We mention two of them here.

- *Largest object first* (LOF): This algorithm is the same as the LP algorithm, except that the objects are arranged such that $r_1 \ge r_2 \ge \cdots \ge r_n$.

- *Smallest object first* (SOF): This algorithm is the same as the LP algorithm, except that the objects are arranged such that $r_1 \le r_2 \le \cdots \le r_n$.

We call algorithm LP and its variations simply as list placement algorithms.

Let A be any algorithm which solves the schedule length minimization problem. We use T_A to denote the length of the schedule produced by algorithm A and T_{OPT} the optimal schedule length. The *performance ratio* of an algorithm A is defined as $\beta_A = T_A/T_{\text{OPT}}$ and the *asymptotic performance ratio* of algorithm A is $\beta_A^\infty = \lim_{R/r^*\to\infty} T_A/T_{\text{OPT}}$, where $r^* = \{r_1, r_2, \ldots, r_n\}$ is the maximum task execution requirement. If $\beta_A \le B$, we call B a *performance bound* of algorithm A. If $\beta_A^\infty \le B$, we call B an *asymptotic performance bound* of algorithm A.

The following theorem gives an asymptotic performance bound for list placement algorithms.

Theorem 1. *For any list placement algorithm A, the asymptotic performance ratio of algorithm A is $\beta_A^\infty \le B$, where the asymptotic performance bound is*

$$B = (1+\Delta)^{1/(\alpha-1)}(1 - K_j\Delta),$$

with

$$K_j = \frac{\phi_j - 1}{\phi_j(\phi_j^{\alpha-1} - 1)}.$$

Proof. The schedule length of algorithm A is

$$\begin{aligned} T_A &= \sum_{i\notin S}\frac{r_i}{s_j} + \sum_{i\in S}\frac{r_i}{s_{j+1}} \\ &= \frac{R - R'}{s_j} + \frac{R'}{s_{j+1}} \\ &= \frac{R}{s_j} - R'\left(\frac{1}{s_j} - \frac{1}{s_{j+1}}\right). \end{aligned}$$

Let E_A denote the actual amount of energy consumed by algorithm A. Then, we get

$$\begin{aligned} E_A &= \sum_{i\notin S} r_i s_j^{\alpha-1} + \sum_{i\in S} r_i s_{j+1}^{\alpha-1} \\ &= (R - R')s_j^{\alpha-1} + R's_{j+1}^{\alpha-1} \\ &= Rs_j^{\alpha-1} + R'(s_{j+1}^{\alpha-1} - s_j^{\alpha-1}) \\ &= E_j + R'(s_{j+1}^{\alpha-1} - s_j^{\alpha-1}). \end{aligned}$$

The problem of choosing S is actually to choose a subset of tasks such that $E_A \le E$, i.e.,

$$E_j + R'(s_{j+1}^{\alpha-1} - s_j^{\alpha-1}) \le E,$$

which implies that

$$\begin{aligned} R' \le \frac{E - E_j}{s_{j+1}^{\alpha-1} - s_j^{\alpha-1}} &= \frac{E_j\Delta}{s_{j+1}^{\alpha-1} - s_j^{\alpha-1}} = \frac{Rs_j^{\alpha-1}\Delta}{s_{j+1}^{\alpha-1} - s_j^{\alpha-1}} \\ &= \frac{R\Delta}{\phi_j^{\alpha-1} - 1} = M, \end{aligned}$$

and that R' is maximized so that the schedule length T_A is minimized. This is exactly the maximum subset sum problem.

It is clear that by using any list placement algorithm A, we always get

$$R' > \frac{R\Delta}{\phi_j^{\alpha-1} - 1} - r^*,$$

where $r^* = \{r_1, r_2, \ldots, r_n\}$ is the maximum task execution requirement; otherwise, if $R' \le M - r^*$, our bag has more room to accommodate more tasks. This implies that

$$\begin{aligned} T_A &< \frac{R}{s_j} - \left(\frac{R\Delta}{\phi_j^{\alpha-1} - 1} - r^*\right)\left(\frac{1}{s_j} - \frac{1}{s_{j+1}}\right) \\ &= R\left(\frac{1}{s_j} - \frac{\Delta}{\phi_j^{\alpha-1} - 1}\left(\frac{1}{s_j} - \frac{1}{s_{j+1}}\right)\right) + r^*\left(\frac{1}{s_j} - \frac{1}{s_{j+1}}\right). \end{aligned}$$

It has been known from [50] that if processors have continuous clock frequency and supply voltage and execution speed and power levels, the optimal schedule length is

$$\tilde{T}_{\text{OPT}} = \frac{R^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}}.$$

Such an optimal schedule length $\tilde{T}_{\text{OPT}}$ gives a lower bound for the optimal schedule length T_{OPT} when processors have discrete clock frequency and supply voltage and execution speed and power levels, i.e.,

$$T_{\text{OPT}} \ge \tilde{T}_{\text{OPT}} = \frac{R}{(E/R)^{1/(\alpha-1)}} = \frac{R}{s_j(1+\Delta)^{1/(\alpha-1)}}.$$

Hence, we get the asymptotic performance ratio of algorithm A as

$$\begin{aligned} \beta_A^\infty &= \lim_{R/r^*\to\infty}\frac{T_A}{T_{\text{OPT}}} \\ &\le \lim_{R/r^*\to\infty}\frac{T_A}{\tilde{T}_{\text{OPT}}} \\ &= s_j(1+\Delta)^{1/(\alpha-1)}\left(\frac{1}{s_j} - \frac{\Delta}{\phi_j^{\alpha-1} - 1}\left(\frac{1}{s_j} - \frac{1}{s_{j+1}}\right)\right) \\ &= (1+\Delta)^{1/(\alpha-1)}\left(1 - \frac{\Delta}{\phi_j^{\alpha-1} - 1}\left(1 - \frac{1}{\phi_j}\right)\right) \\ &= (1+\Delta)^{1/(\alpha-1)}(1 - K_j\Delta), \end{aligned}$$

where

$$K_j = \frac{\phi_j - 1}{\phi_j(\phi_j^{\alpha-1} - 1)}.$$

The theorem is proven. ■

We notice that for a given α and a given gap ϕ_j, the asymptotic performance bound B in Theorem 1 can be viewed as a function of Δ, i.e.,

$$B(\Delta) = (1+\Delta)^{1/(\alpha-1)}(1 - K_j\Delta).$$

If we can show that $\beta_A^\infty \le B_{\text{worst}}$, where B_{worst} is independent of Δ, then B_{worst} is called a *worst-case asymptotic performance bound* for algorithm A. When Δ is a random variable, β_A^∞ also becomes a random variable. If we can show that $\overline{\beta_A^\infty} \le B_{\text{average}}$, where B_{average} is independent of Δ, then B_{average} is called an *average-case asymptotic performance bound* for algorithm A. (Notation: We use $\bar{x}$ to represent the expectation of a random variable x.)

The following corollary gives a worst-case asymptotic performance bound an average-case asymptotic performance bound for list placement algorithms.

Corollary 1. *For any list placement algorithm A, a worst-case asymptotic performance bound for algorithm A is $\beta_A^\infty \le B_{\text{worst}}$, where*

$$B_{\text{worst}} = \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K_j + 1)^{\alpha/(\alpha-1)}}{K_j^{1/(\alpha-1)}}.$$

If Δ is a random variable uniformly distributed in $[0, \phi_j^{\alpha-1} - 1)$, an average-case asymptotic performance bound for algorithm A is $\overline{\beta_A^\infty} \le B_{\text{average}}$, where

$$B_{\text{average}} = \frac{\alpha-1}{\alpha} \cdot \frac{1}{\phi_j^{\alpha-1}-1} \left((K_j+1)\phi_j^\alpha - \frac{\alpha}{2\alpha-1} \cdot K_j \phi_j^{2\alpha-1} - \frac{\alpha-1}{2\alpha-1} \cdot K_j - 1 \right).$$

Proof. Notice that

$$\frac{\partial B(\Delta)}{\partial \Delta} = (1+\Delta)^{1/(\alpha-1)} \left(\frac{1-K_j\Delta}{(\alpha-1)(1+\Delta)} - K_j \right).$$

To maximize $B(\Delta)$, we need $\partial B(\Delta)/\partial\Delta = 0$, that is,

$$\frac{1-K_j\Delta}{(\alpha-1)(1+\Delta)} - K_j = 0.$$

Consequently, when

$$\Delta = \frac{1-(\alpha-1)K_j}{\alpha K_j},$$

$B(\Delta)$ reaches its maximum value of

$$B_{\text{worst}} = \left(\frac{(\alpha-1)(K_j+1)}{\alpha} \right) \left(\frac{K_j+1}{\alpha K_j} \right)^{1/(\alpha-1)} = \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K_j+1)^{\alpha/(\alpha-1)}}{K_j^{1/(\alpha-1)}}.$$

Thus, we have proved that B_{worst} is a worst-case asymptotic performance bound for algorithm A, i.e.,

$$\beta_A^\infty \le B(\Delta) \le B_{\text{worst}} = \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K_j+1)^{\alpha/(\alpha-1)}}{K_j^{1/(\alpha-1)}}.$$

As for the average-case asymptotic performance bound for algorithm A, we notice that

$$\int B(\Delta)d\Delta = \int (1+\Delta)^{1/(\alpha-1)}(1-K_j\Delta)d\Delta = \frac{\alpha-1}{\alpha} \left((1+\Delta)^{\alpha/(\alpha-1)}(1-K_j\Delta) + (1+\Delta)^{(2\alpha-1)/(\alpha-1)} \cdot \frac{\alpha-1}{2\alpha-1} \cdot K_j \right).$$

Consequently, if Δ is a random variable uniformly distributed in $[0, \phi_j^{\alpha-1} - 1)$, we have

$$\begin{aligned}
\overline{B(\Delta)} &= \frac{1}{\phi_j^{\alpha-1}-1} \int_0^{\phi_j^{\alpha-1}-1} B(\Delta)d\Delta \\
&= \frac{1}{\phi_j^{\alpha-1}-1} \int_0^{\phi_j^{\alpha-1}-1} (1+\Delta)^{1/(\alpha-1)}(1-K_j\Delta)d\Delta \\
&= \frac{1}{\phi_j^{\alpha-1}-1} \cdot \frac{\alpha-1}{\alpha} \left((1+\Delta)^{\alpha/(\alpha-1)}(1-K_j\Delta) + (1+\Delta)^{(2\alpha-1)/(\alpha-1)} \cdot \frac{\alpha-1}{2\alpha-1} \cdot K_j \right) \Bigg|_0^{\phi_j^{\alpha-1}-1} \\
&= \frac{\alpha-1}{\alpha} \cdot \frac{1}{\phi_j^{\alpha-1}-1} \left((K_j+1)\phi_j^\alpha - \frac{\alpha}{2\alpha-1} \cdot K_j \phi_j^{2\alpha-1} - \frac{\alpha-1}{2\alpha-1} \cdot K_j - 1 \right).
\end{aligned}$$

Table 1
Numerical data for the asymptotic performance bound in Theorem 1 ($\alpha = 3$).

z	$\phi_j = 1.4$	$\phi_j = 1.8$	$\phi_j = 2.2$	$\phi_j = 2.6$	$\phi_j = 3.0$
0	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000
1	1.0090942	1.0310805	1.0620115	1.0999808	1.1437754
2	1.0169888	1.0571744	1.1122661	1.1781337	1.2521981
3	1.0237402	1.0787977	1.1526748	1.2392577	1.3349157
4	1.0293997	1.0963662	1.1846076	1.2864196	1.3974580
5	1.0340134	1.1102219	1.2090909	1.3217346	1.4433757
6	1.0376237	1.1206506	1.2269201	1.3467429	1.4751271
7	1.0402692	1.1278946	1.2387285	1.3626129	1.4945085
8	1.0419855	1.1321615	1.2450316	1.3702593	1.5028861
9	1.0428055	1.1336313	1.2462569	1.3704168	1.5013327
10	1.0427593	1.1324615	1.2427642	1.3636880	1.4907120
11	1.0418751	1.1287909	1.2348603	1.3505758	1.4717337
12	1.0401789	1.1227427	1.2228102	1.3315059	1.4449913
13	1.0376952	1.1144271	1.2068447	1.3068435	1.4109886
14	1.0344467	1.1039431	1.1871666	1.2769053	1.3701581
15	1.0304546	1.0913804	1.1639559	1.2419683	1.3228757
16	1.0257390	1.0768202	1.1373729	1.2022771	1.2694706
17	1.0203185	1.0603368	1.1075617	1.1580494	1.2102341
18	1.0142109	1.0419981	1.0746522	1.1094798	1.1454257
19	1.0074328	1.0218666	1.0387625	1.0567437	1.0752777

Table 2
Numerical data for the worst-case asymptotic performance bound in Corollary 1.

ϕ	$\alpha = 3.0$	$\alpha = 3.5$	$\alpha = 4.0$	$\alpha = 4.5$	$\alpha = 5.0$
1.1	1.0034093	1.0039773	1.0045449	1.0051120	1.0056786
1.2	1.0125029	1.0145833	1.0166590	1.0187285	1.0207905
1.3	1.0259738	1.0302884	1.0345826	1.0388508	1.0430874
1.4	1.0428902	1.0499995	1.0570544	1.0640399	1.0709420
1.5	1.0625693	1.0729152	1.0831474	1.0932359	1.1031535
1.6	1.0844994	1.0984341	1.1121654	1.1256425	1.1388205
1.7	1.1082893	1.1260959	1.1435760	1.1606516	1.1772569
1.8	1.1336343	1.1555428	1.1769653	1.1977922	1.2179343
1.9	1.1602939	1.1864920	1.2120073	1.2366945	1.2604409
2.0	1.1880752	1.2187171	1.2484416	1.2770653	1.3044527
2.1	1.2168219	1.2520348	1.2860583	1.3186693	1.3497126
2.2	1.2464060	1.2862951	1.3246861	1.3613164	1.3960145
2.3	1.2767212	1.3213739	1.3641843	1.4048515	1.4431924
2.4	1.3076790	1.3571680	1.4044363	1.4491469	1.4911110
2.5	1.3392048	1.3935906	1.4453445	1.4940971	1.5396598
2.6	1.3712350	1.4305681	1.4868271	1.5396140	1.5887478
2.7	1.4037157	1.4680380	1.5288146	1.5856239	1.6382993
2.8	1.4366003	1.5059466	1.5712480	1.6320645	1.6882515
2.9	1.4698485	1.5442479	1.6140768	1.6788832	1.7385516
3.0	1.5034254	1.5829016	1.6572575	1.7260349	1.7891549

Thus, we have proved that $\overline{B(\Delta)}$ is an average-case asymptotic performance bound for algorithm A, i.e.,

$$\overline{\beta_A^\infty} \le \overline{B(\Delta)} = \frac{\alpha-1}{\alpha} \cdot \frac{1}{\phi_j^{\alpha-1}-1} \left((K_j+1)\phi_j^\alpha - \frac{\alpha}{2\alpha-1} \cdot K_j \phi_j^{2\alpha-1} - \frac{\alpha-1}{2\alpha-1} \cdot K_j - 1 \right).$$

This proves the corollary. ■

In Table 1, we demonstrate numerical data for the asymptotic performance bound B in Theorem 1. Assume that $\alpha = 3$. For $\phi_j = 1.4, 1.8, 2.2, 2.6, 3.0$, we show the asymptotic performance bound for the asymptotic performance ratio β_A^∞, i.e., $B = (1+\Delta)^{1/(\alpha-1)}(1-K_j\Delta)$, for $\Delta = 0.05z(\phi_j^{\alpha-1}-1)$, where $0 \le z \le 19$. We observe that the asymptotic performance bound B is an increasing function of Δ until B reaches its maximum value, and then B becomes a decreasing function of Δ as Δ further increases. This means that for $\Delta \in [0, \phi_j^{\alpha-1}-1)$, when Δ is close to the boundaries of the interval, it is easier to get close-to-optimal performance. However, When Δ is in the middle of the interval, it is more difficult to get close-to-optimal performance.

In Table 2, we demonstrate numerical data for the worst-case asymptotic performance bound B_{worst} in Corollary 1. For

Table 3
Numerical data for the average-case asymptotic performance bound in Corollary 1.

ϕ	$\alpha = 3.0$	$\alpha = 3.5$	$\alpha = 4.0$	$\alpha = 4.5$	$\alpha = 5.0$
1.1	1.0022731	1.0026518	1.0030303	1.0034084	1.0037862
1.2	1.0083379	1.0097262	1.0111111	1.0124913	1.0138659
1.3	1.0173273	1.0202093	1.0230765	1.0259243	1.0287480
1.4	1.0286243	1.0333788	1.0380934	1.0427559	1.0473547
1.5	1.0417778	1.0487058	1.0555501	1.0622856	1.0688898
1.6	1.0564497	1.0657938	1.0749870	1.0839864	1.0927542
1.7	1.0723817	1.0843391	1.0960521	1.1074546	1.1184904
1.8	1.0893726	1.1041050	1.1184710	1.1323763	1.1457441
1.9	1.1072633	1.1249043	1.1420267	1.1585041	1.1742381
2.0	1.1259259	1.1465865	1.1665452	1.1856406	1.2037531
2.1	1.1452568	1.1690293	1.1918854	1.2136267	1.2341143
2.2	1.1651705	1.1921322	1.2179319	1.2423331	1.2651812
2.3	1.1855961	1.2158118	1.2445890	1.2716539	1.2968399
2.4	1.2064744	1.2399983	1.2717771	1.3015015	1.3289976
2.5	1.2277551	1.2646325	1.2994294	1.3318029	1.3615783
2.6	1.2493954	1.2896644	1.3274894	1.3624975	1.3945191
2.7	1.2713586	1.3150510	1.3559090	1.3935338	1.4277680
2.8	1.2936130	1.3407551	1.3846470	1.4248685	1.4612816
2.9	1.3161311	1.3667449	1.4136681	1.4564647	1.4950234
3.0	1.3388889	1.3929924	1.4429417	1.4882910	1.5289630

$\alpha = 3.0, 3.5, 4.0, 4.5, 5.0$, and $\phi = 1.1, 1.2, 1.3, \ldots, 3.0$, we show the worst-case asymptotic performance bound B_{worst} for the asymptotic performance ratio β_A^∞. We observe that the worst-case asymptotic performance bound B_{worst} is an increasing function of α and ϕ. As the gap between two successive speed levels becomes larger, it is more difficult to get close-to-optimal performance. Since the typical values of α and ϕ are $\alpha = 3$ and $\phi < 2$ [41], the worst-case asymptotic performance bound B_{worst} is very close to optimal.

In Table 3, we demonstrate numerical data for the average-case asymptotic performance bound B_{average} in Corollary 1. For $\alpha = 3.0, 3.5, 4.0, 4.5, 5.0$, and $\phi = 1.1, 1.2, 1.3, \ldots, 3.0$, we show the average-case asymptotic performance bound B_{average} for the asymptotic performance ratio β_A^∞. We observe that the average-case asymptotic performance bound B_{average} is lower than the worst-case asymptotic performance bound B_{worst} and predicts the performance our scheduling algorithms more accurately in real applications.

3.2. Scheduling on multiprocessor computers

Now, we consider a multiprocessor computer with $m > 1$ processors. We observe that the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer with $m > 1$ processors consists of two components, namely, scheduling tasks and determining speeds. Scheduling the tasks is essentially to partition the n tasks into m groups $R_1, R_2, \ldots, R_m$, such that each processor executes one group of tasks. (Notation: R_k stands for a group of tasks as well as the total execution requirement of tasks in group k.) Once a partition $R_1, R_2, \ldots, R_m$ (i.e., a schedule) is given, execution speeds which attempt to minimize the schedule length within energy consumption constraint can be determined by using the LP algorithms.

The classic *list scheduling* (LS) algorithm [24] and its variations have been proposed and analyzed to solve the partitioning problem (i.e., the scheduling problem). The task execution times are simply $r_1, r_2, \ldots, r_n$, and tasks are assigned to the m processors (i.e., groups) by using the following method. Initially, task k is scheduled on processor (or group) k, where $1 \le k \le m$, and tasks $1, 2, \ldots, m$ are removed from the list simultaneously. Upon the completion of a task k, the first unscheduled task in the list, i.e., task $m + 1$, is removed from the list and scheduled to be executed on processor k. This process repeats until all tasks in the list are finished. Algorithm LS has many variations, depending on the strategy used in the initial ordering of the tasks. We mention two of them here.

- *Largest requirement first* (LRF): This algorithm is the same as the LS algorithm, except that the tasks are arranged such that $r_1 \ge r_2 \ge \cdots \ge r_n$.
- *Smallest requirement first* (SRF): This algorithm is the same as the LS algorithm, except that the tasks are arranged such that $r_1 \le r_2 \le \cdots \le r_n$.

We call algorithm LS and its variations simply as list scheduling algorithms.

We use the notation A_1–A_2 to represent a power-aware task scheduling algorithm on multiprocessor computers. Algorithm A_1–A_2 works as follows. First, a list scheduling algorithm A_1 is used to produce a partition $R_1, R_2, \ldots, R_m$. Second, for each processor k, a list placement algorithm A_2 is used to produce a schedule of tasks in R_k on processor k by using

$$E_k = \left(\frac{R_k^\alpha}{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha}\right) E$$

amount of energy, where $1 \le k \le m$. The above energy E_k will be explained shortly.

The following result gives a worst-case asymptotic performance bound for algorithm A_1–A_2.

Theorem 2. *For any list scheduling algorithm A_1 and any list placement algorithm A_2, the asymptotic performance ratio of algorithm A_1–A_2 is $\beta_{A_1-A_2}^\infty \le B$, where the worst-case asymptotic performance bound is*

$$B = \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K+1)^{\alpha/(\alpha-1)}}{K^{1/(\alpha-1)}},$$

and

$$K = \frac{\phi - 1}{\phi(\phi^{\alpha-1} - 1)},$$

with $\phi = \max\{\phi_1, \phi_2, \ldots, \phi_d\}$.

Proof. It has been known from [50] that for a given partition $R_1, R_2, \ldots, R_m$ of the n tasks into m groups generated by any list scheduling algorithm A_1, if processors have continuous clock frequency and supply voltage and execution speed and power levels, the optimal energy allocation is to allocate

$$\tilde{E}_k = \left(\frac{R_k^\alpha}{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha}\right) E$$

amount of energy to processor k, where $1 \le k \le m$. Furthermore, if we use $\tilde{T}_k$ to denote the optimal schedule length of R_k on processor k, then we have $\tilde{T}_1 = \tilde{T}_2 = \cdots = \tilde{T}_m = \tilde{T}$, where

$$\tilde{T}_k = \frac{R_k^{\alpha/(\alpha-1)}}{\tilde{E}_k^{1/(\alpha-1)}},$$

for all $1 \le k \le m$, and

$$\tilde{T} = \left(\frac{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha}{E}\right)^{1/(\alpha-1)}.$$

Let $T_{A_2,k}$ denote the length of the schedule produced by algorithm A_2 on processor k with energy constraint $E_k = \tilde{E}_k$, where $1 \le k \le m$. In the proof of Theorem 1, we have already shown that for list placement algorithm A_2, we have

$$\lim_{R_k/r^* \to \infty} \frac{T_{A_2,k}}{\tilde{T}_k} = (1 + \Delta_k)^{1/(\alpha-1)}(1 - K_{j_k}\Delta_k),$$

where we assume that $E_{k,j_k} \le E_k < E_{k,j_k+1}$, with $E_{k,j_k} = R_k s_{j_k}^{\alpha-1}$, and $1 \le j_k \le d-1$, and $E_k = E_{k,j_k}(1 + \Delta_k)$, where $0 \le \Delta_k < \phi_{j_k}^{\alpha-1} - 1$, for all $1 \le k \le m$. Since each processor k has its own j_k and Δ_k,

where $1 \le k \le m$, we need to get rid of these j_k's and Δ_k's to reach a manageable and meaningful result.

First, from Corollary 1, we have

$$\lim_{R_k/r^* \to \infty} \frac{T_{A_2,k}}{\tilde{T}_k} \le \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K_{j_k} + 1)^{\alpha/(\alpha-1)}}{K_{j_k}^{1/(\alpha-1)}},$$

for all $1 \le k \le m$. Thus, there is no more Δ_k.

Next, let us view the worst-case asymptotic performance bound B_{worst} in Corollary 1 as a function of K_j, i.e.,

$$B_{\text{worst}}(K_j) = \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K_j + 1)^{\alpha/(\alpha-1)}}{K_j^{1/(\alpha-1)}}.$$

It can be verified that $B_{\text{worst}}(K_j)$ is a decreasing function of K_j. This can be seen from the fact that

$$\frac{\partial B_{\text{worst}}(K_j)}{\partial K_j} = \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \left(1 + \frac{1}{K_j}\right)^{1/(\alpha-1)} \left(1 - \frac{1}{(\alpha - 1)K_j}\right) < 0,$$

where

$$K_j = \frac{\phi_j - 1}{\phi_j(\phi_j^{\alpha-1} - 1)} = \frac{1}{\phi_j(\phi_j^{\alpha-2} + \phi_j^{\alpha-3} + \cdots + \phi_j^0)} < \frac{1}{\alpha - 1},$$

since $\phi_j > 1$ and $\alpha \ge 3$. Hence, we get

$$B_{\text{worst}}(K_j) \le \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K + 1)^{\alpha/(\alpha-1)}}{K^{1/(\alpha-1)}},$$

where $K = \min\{K_1, K_2, \ldots, K_d\}$, and

$$K = \frac{\phi - 1}{\phi(\phi^{\alpha-1} - 1)},$$

with $\phi = \max\{\phi_1, \phi_2, \ldots, \phi_d\}$, since K is a decreasing function of ϕ. The above discussion leads to

$$\lim_{R_k/r^* \to \infty} \frac{T_{A_2,k}}{\tilde{T}_k} \le \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K + 1)^{\alpha/(\alpha-1)}}{K^{1/(\alpha-1)}},$$

for all $1 \le k \le m$. Thus, there is no more j_k.

Notice that the length of the schedule produced by algorithm A_1–A_2 is

$$T_{A_1-A_2} = \max\{T_{A_2,1}, T_{A_2,2}, \ldots, T_{A_2,m}\}.$$

Therefore, we obtain

$$\begin{aligned}
&\lim_{R/r^* \to \infty} \frac{T_{A_1-A_2}}{\tilde{T}} \\
&= \lim_{R/r^* \to \infty} \frac{\max\{T_{A_2,1}, T_{A_2,2}, \ldots, T_{A_2,m}\}}{\tilde{T}} \\
&= \lim_{R/r^* \to \infty} \max\left\{\frac{T_{A_2,1}}{\tilde{T}}, \frac{T_{A_2,2}}{\tilde{T}}, \ldots, \frac{T_{A_2,m}}{\tilde{T}}\right\} \\
&= \max\left\{\lim_{R/r^* \to \infty} \frac{T_{A_2,1}}{\tilde{T}}, \lim_{R/r^* \to \infty} \frac{T_{A_2,2}}{\tilde{T}}, \ldots, \lim_{R/r^* \to \infty} \frac{T_{A_2,m}}{\tilde{T}}\right\} \\
&= \max\left\{\lim_{R_1/r^* \to \infty} \frac{T_{A_2,1}}{\tilde{T}_1}, \lim_{R_2/r^* \to \infty} \frac{T_{A_2,2}}{\tilde{T}_2}, \ldots, \lim_{R_m/r^* \to \infty} \frac{T_{A_2,m}}{\tilde{T}_m}\right\} \\
&\le \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K + 1)^{\alpha/(\alpha-1)}}{K^{1/(\alpha-1)}}.
\end{aligned}$$

Let T_{OPT} denote the optimal schedule length when processors have discrete clock frequency and supply voltage and execution speed and power levels, and $\tilde{T}_{\text{OPT}}$ denote the optimal schedule length when processors have continuous clock frequency and supply voltage and execution speed and power levels. Clearly, we have $T_{\text{OPT}} \ge \tilde{T}_{\text{OPT}}$. It has been proven in [50] that for any list scheduling algorithm A_1, as $R/r^* \to \infty$, we have $\tilde{T}/\tilde{T}_{\text{OPT}} \to 1$. Consequently, the asymptotic performance ratio of algorithm A_1–A_2 is

$$\begin{aligned}
\beta^{\infty}_{A_1-A_2} &= \lim_{R/r^* \to \infty} \frac{T_{A_1-A_2}}{T_{\text{OPT}}} \\
&\le \lim_{R/r^* \to \infty} \frac{T_{A_1-A_2}}{\tilde{T}_{\text{OPT}}} \\
&= \left(\lim_{R/r^* \to \infty} \frac{T_{A_1-A_2}}{\tilde{T}}\right)\left(\lim_{R/r^* \to \infty} \frac{\tilde{T}}{\tilde{T}_{\text{OPT}}}\right) \\
&\le \frac{\alpha - 1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K + 1)^{\alpha/(\alpha-1)}}{K^{1/(\alpha-1)}}.
\end{aligned}$$

This proves the theorem. ■

4. Time constrained scheduling

Given n tasks with task execution requirements $r_1, r_2, \ldots, r_n$, the problem of *minimizing energy consumption with schedule length constraint* T on a multiprocessor computer with m processors is to determine the execution speeds $s_{j_1}, s_{j_2}, \ldots, s_{j_n}$ and a nonpreemptive schedule of the n tasks on the m processors such that the total energy consumption is minimized and the schedule length does not exceed T.

4.1. Scheduling on uniprocessor computers

We first consider a uniprocessor computer with $m = 1$ processor. It is clear that on a uniprocessor computer with time constraint T, the problem of minimizing energy consumption with schedule length constraint is simply to decide the execution speeds $s_{j_1}, s_{j_2}, \ldots, s_{j_n}$, such that the total energy consumption

$$e_1 + e_2 + \cdots + e_n = r_1 s_{j_1}^{\alpha-1} + r_2 s_{j_2}^{\alpha-1} + \cdots + r_n s_{j_n}^{\alpha-1}$$

is minimized and the schedule length does not exceed T, i.e.,

$$t_1 + t_2 + \cdots + t_n = \frac{r_1}{s_{j_1}} + \frac{r_2}{s_{j_2}} + \cdots + \frac{r_n}{s_{j_n}} \le T.$$

We define $T_j = R/s_j$, which is the total execution time of all the tasks when they are executed at speed s_j, where $1 \le j \le d$. It is reasonable to assume that the given time deadline T is in the range $T_1 > T \ge T_d$. If $T < T_d$, there is no feasible schedule to meet the time constraint. If $T \ge T_1$, we simply execute all tasks at speed s_1 and consume the minimum possible energy.

Again, even on a uniprocessor computer with only $d = 2$ speed levels, the problem of minimizing energy consumption with schedule length constraint is NP-hard.

Proposition 2. *The problem of minimizing energy consumption with schedule length constraint is NP-hard on a uniprocessor computer with two speed levels.*

Proof. Consider a uniprocessor computer with $d = 2$ speed levels s_1 and s_2. Assume that the time constraint T is $T_1 > T \ge T_2$. It is clear that some tasks are executed at speed s_1, while others are executed at speed s_2. Let $S \subseteq \{1, 2, \ldots, n\}$ be the set of tasks which are executed at speed s_2, and

$$R' = \sum_{i \in S} r_i$$

be the total execution requirement of tasks in S. Similar to the proof of Proposition 1, the amount of energy consumed is

$$R s_1^{\alpha-1} + R'(s_2^{\alpha-1} - s_1^{\alpha-1}).$$

The actual schedule length is

$$\frac{R}{s_1} - R'\left(\frac{1}{s_1} - \frac{1}{s_2}\right) = T_1 - R'\left(\frac{1}{s_1} - \frac{1}{s_2}\right).$$

Our problem of determination of the execution speeds is equivalent to the problem of choosing S such that

$$T_1 - R'\left(\frac{1}{s_1} - \frac{1}{s_2}\right) \le T,$$

that is,

$$R' \ge \frac{T_1 - T}{1/s_1 - 1/s_2} = M,$$

and that R' is minimized so that the energy consumed is minimized.

The above problem is exactly the *minimum subset sum* problem. Assume that there are n objects (i.e., tasks) of sizes $r_1, r_2, \ldots, r_n$ and a bag of capacity M. The minimum subset problem is to find a subset S of objects to fill the bag such that the total size of the objects filling the bag is as small as possible but is at least as large as the capacity M of the bag. Again, since the solution of the minimum subset problem can be used to decide whether there is a subset of objects whose total size is exactly M (i.e., the subset sum problem), the minimum subset sum problem is NP-hard, so is our problem of minimizing energy consumption with schedule length constraint. ■

It has been known from [50] that if processors have continuous clock frequency and supply voltage and execution speed and power levels, the energy consumption is minimized when all tasks are executed at the same speed $s = R/T$. Assume that $T_j > T \ge T_{j+1}$, where $1 \le j \le d-1$. This implies that $s_j < s \le s_{j+1}$. Our strategy is to achieve close-to-optimal performance by using speed levels s_j and s_{j+1}.

Our scheduling algorithm works as follows. All the n tasks are executed at speed s_j or s_{j+1}. If all tasks are executed at speed s_j, the total execution time is T_j, and there is extra time $T_j - T$ to be reduced by selecting some tasks to be executed at speed s_{j+1}. We choose a subset $S \subseteq \{1, 2, \ldots, n\}$ of tasks that are executed at speed s_{j+1}. Tasks not in S are executed at speed s_j. Let

$$R' = \sum_{i \in S} r_i.$$

The subset S of tasks are chosen such that R' is as small as possible under the condition that

$$R' \ge M = \frac{R\Delta}{1 - 1/\phi_j},$$

where Δ is defined such that $T = T_j(1-\Delta) = (R/s_j)(1-\Delta)$, with $0 < \Delta \le 1 - 1/\phi_j$, and $\phi_j = s_{j+1}/s_j$ for all $1 \le j \le d-1$. The bound M will be explained shortly.

The problem of finding S is equivalent to the minimum subset sum problem, where the n objects are the n tasks with sizes $r_1, r_2, \ldots, r_n$, and a bag of capacity M is the space translated from the required time reduction $T_j - T$ to be filled by tasks which are executed at speed s_{j+1}. The problem can be solved by using the LP algorithm slightly modified as follows. Initially, the content of the bag is zero. We scan the list of objects one after another. For each object i, we fill the bag with the object if the bag is still not full. After object i is packed into the bag, the content of the bag is increased by r_i. The modified LP algorithm also has variations such as SOF and LOF.

Let A be any algorithm which solves the energy consumption minimization problem. We use E_A to denote the amount of energy consumed by algorithm A and E_{OPT} the minimum energy consumption. The *performance ratio* of an algorithm A is defined as $\beta_A = E_A/E_{\text{OPT}}$ and the *asymptotic performance ratio* of algorithm A is $\beta_A^\infty = \lim_{R/r^* \to \infty} E_A/E_{\text{OPT}}$, where $r^* = \{r_1, r_2, \ldots, r_n\}$ is the maximum task execution requirement. If $\beta_A^\infty \le B$, we call B an *asymptotic performance bound* for algorithm A.

The following theorem gives an asymptotic performance bound for list placement algorithms.

Theorem 3. *For any list placement algorithm A, the asymptotic performance ratio of algorithm A is $\beta_A^\infty \le B$, where the asymptotic performance bound is*

$$B = (1-\Delta)^{\alpha-1}\left(1 + \frac{\Delta}{K_j}\right),$$

with

$$K_j = \frac{\phi_j - 1}{\phi_j(\phi_j^{\alpha-1} - 1)}.$$

Proof. Similar to the proof of Theorem 1, we know that the amount of energy consumed by algorithm A is

$$E_A = Rs_j^{\alpha-1} + R'(s_{j+1}^{\alpha-1} - s_j^{\alpha-1}),$$

and the actual schedule length of algorithm A is

$$T_A = T_j - R'\left(\frac{1}{s_j} - \frac{1}{s_{j+1}}\right).$$

The problem of choosing S is actually to choose a subset of tasks such that $T_A \le T$, i.e.,

$$T_j - R'\left(\frac{1}{s_j} - \frac{1}{s_{j+1}}\right) \le T,$$

which implies that

$$R' \ge \frac{T_j - T}{1/s_j - 1/s_{j+1}} = \frac{T_j\Delta}{1/s_j - 1/s_{j+1}} = \frac{(R/s_j)\Delta}{1/s_j - 1/s_{j+1}} = \frac{R\Delta}{1 - 1/\phi_j} = M,$$

and that R' is minimized so that the energy consumption E_A is minimized. This is exactly the minimum subset sum problem.

It is clear that by using any list placement algorithm A, we always get

$$R' < \frac{R\Delta}{1 - 1/\phi_j} + r^*,$$

where $r^* = \{r_1, r_2, \ldots, r_n\}$ is the maximum task execution requirement; otherwise, if $R' \ge M + r^*$, we can take one object away while still fill the bag. This implies that

$$E_A < Rs_j^{\alpha-1} + \left(\frac{R\Delta}{1 - 1/\phi_j} + r^*\right)(s_{j+1}^{\alpha-1} - s_j^{\alpha-1})$$
$$= R\left(s_j^{\alpha-1} + \frac{\Delta}{1 - 1/\phi_j}(s_{j+1}^{\alpha-1} - s_j^{\alpha-1})\right) + r^*(s_{j+1}^{\alpha-1} - s_j^{\alpha-1}).$$

It has been known from [50] that if processors have continuous clock frequency and supply voltage and execution speed and power levels, the minimum energy consumption is

$$\tilde{E}_{\text{OPT}} = \frac{R^\alpha}{T^{\alpha-1}}.$$

Such minimum energy consumption $\tilde{E}_{\text{OPT}}$ gives a lower bound for the minimum energy consumption E_{OPT} when processors have discrete clock frequency and supply voltage and execution speed and power levels, i.e.,

$$E_{\text{OPT}} \ge \tilde{E}_{\text{OPT}} = R(R/T)^{\alpha-1} = R\left(\frac{s_j}{1-\Delta}\right)^{\alpha-1}.$$

Hence, we get the asymptotic performance ratio of algorithm A as

$$\begin{aligned}\beta_A^{\infty} &= \lim_{R/r^*\to\infty} \frac{E_A}{E_{\mathrm{OPT}}} \\ &\le \lim_{R/r^*\to\infty} \frac{E_A}{\tilde{E}_{\mathrm{OPT}}} \\ &= \left(\frac{1-\Delta}{s_j}\right)^{\alpha-1}\left(s_j^{\alpha-1} + \frac{\Delta}{1-1/\phi_j}(s_{j+1}^{\alpha-1} - s_j^{\alpha-1})\right) \\ &= (1-\Delta)^{\alpha-1}\left(1 + \frac{\Delta}{1-1/\phi_j}(\phi_j^{\alpha-1} - 1)\right) \\ &= (1-\Delta)^{\alpha-1}\left(1 + \frac{\Delta}{K_j}\right),\end{aligned}$$

where

$$K_j = \frac{\phi_j - 1}{\phi_j(\phi_j^{\alpha-1} - 1)}.$$

The theorem is proven. ■

The following corollary gives a worst-case asymptotic performance bound an average-case asymptotic performance bound for list placement algorithms.

Corollary 2. *For any list placement algorithm A, a worst-case asymptotic performance bound for algorithm A is $\beta_A^{\infty} \le B_{\mathrm{worst}}$, where*

$$B_{\mathrm{worst}} = \frac{(\alpha-1)^{\alpha-1}}{\alpha^{\alpha}} \cdot \frac{(K_j+1)^{\alpha}}{K_j}.$$

If Δ is a random variable uniformly distributed in $(0, 1-1/\phi_j]$, an average-case asymptotic performance bound for algorithm A is $\overline{\beta_A^{\infty}} \le B_{\mathrm{average}}$, where

$$B_{\mathrm{average}} = \frac{1}{\alpha}\left(1 + \frac{1}{(\alpha+1)K_j} \cdot \frac{\phi_j}{\phi_j - 1}\left(1 - \frac{1}{\phi_j^{\alpha+1}}\right)\right).$$

Proof. If we view the asymptotic performance bound B in Theorem 3 as a function of Δ, i.e.,

$$B(\Delta) = (1-\Delta)^{\alpha-1}\left(1 + \frac{\Delta}{K_j}\right),$$

then we obtain

$$\frac{\partial B(\Delta)}{\partial \Delta} = (1-\Delta)^{\alpha-2} \cdot \frac{1 - (\alpha-1)K_j - \alpha\Delta}{K_j}.$$

To maximize $B(\Delta)$, we need $\partial B(\Delta)/\partial\Delta = 0$, that is,

$$1 - (\alpha-1)K_j - \alpha\Delta = 0.$$

Consequently, when

$$\Delta = \frac{1-(\alpha-1)K_j}{\alpha},$$

$B(\Delta)$ reaches its maximum value of

$$\begin{aligned}B_{\mathrm{worst}} &= \left(\frac{(\alpha-1)(K_j+1)}{\alpha}\right)^{\alpha-1}\left(\frac{K_j+1}{\alpha K_j}\right) \\ &= \frac{(\alpha-1)^{\alpha-1}}{\alpha^{\alpha}} \cdot \frac{(K_j+1)^{\alpha}}{K_j}.\end{aligned}$$

Table 4
Numerical data for the asymptotic performance bound in Theorem 3 ($\alpha = 3$).

z	$\phi_j = 1.4$	$\phi_j = 1.8$	$\phi_j = 2.2$	$\phi_j = 2.6$	$\phi_j = 3.0$
1	1.0182710	1.0631269	1.1278684	1.2099579	1.3082222
2	1.0342661	1.1176178	1.2371359	1.3879991	1.5680000
3	1.0480441	1.1638044	1.3286592	1.5357595	1.7820000
4	1.0596637	1.2020188	1.4032952	1.6548753	1.9528889
5	1.0691837	1.2325926	1.4619008	1.7469822	2.0833333
6	1.0766629	1.2558578	1.5053329	1.8137164	2.1760000
7	1.0821600	1.2721462	1.5344483	1.8567138	2.2335556
8	1.0857339	1.2817896	1.5501038	1.8776104	2.2586667
9	1.0874433	1.2851200	1.5531564	1.8780421	2.2540000
10	1.0873469	1.2824691	1.5444628	1.8596450	2.2222222
11	1.0855037	1.2741689	1.5248800	1.8240549	2.1660000
12	1.0819722	1.2605511	1.4952648	1.7729079	2.0880000
13	1.0768114	1.2419477	1.4564740	1.7078400	1.9908889
14	1.0700800	1.2186904	1.4093646	1.6304871	1.8773333
15	1.0618367	1.1911111	1.3547934	1.5424852	1.7500000
16	1.0521404	1.1595417	1.2936172	1.4454703	1.6115556
17	1.0410498	1.1243141	1.2266929	1.3410783	1.4646667
18	1.0286237	1.0857600	1.1548774	1.2309453	1.3120000
19	1.0149208	1.0442114	1.0790274	1.1167072	1.1562222
20	1.0000000	1.0000000	1.0000000	1.0000000	1.0000000

As for the average-case asymptotic performance bound for algorithm A, we notice that

$$\begin{aligned}\int B(\Delta)d\Delta &= \int (1-\Delta)^{\alpha-1}\left(1 + \frac{\Delta}{K_j}\right)d\Delta \\ &= -\frac{1}{\alpha}\left((1-\Delta)^{\alpha}\left(1 + \frac{\Delta}{K_j}\right) + \frac{1}{(\alpha+1)K_j}(1-\Delta)^{\alpha+1}\right).\end{aligned}$$

Consequently, if Δ is a random variable uniformly distributed in $(0, 1-1/\phi_j]$, we have

$$\begin{aligned}\overline{B(\Delta)} &= \frac{1}{1-1/\phi_j}\int_0^{1-1/\phi_j} B(\Delta)d\Delta \\ &= \frac{1}{1-1/\phi_j}\int_0^{1-1/\phi_j} (1-\Delta)^{\alpha-1}\left(1 + \frac{\Delta}{K_j}\right)d\Delta \\ &= \frac{1}{\alpha} \cdot \frac{\phi_j}{\phi_j - 1}\left((1-\Delta)^{\alpha}\left(1 + \frac{\Delta}{K_j}\right)\right. \\ &\quad \left. + \frac{1}{(\alpha+1)K_j}(1-\Delta)^{\alpha+1}\right)\Bigg|_{1-1/\phi_j}^{0} \\ &= \frac{1}{\alpha}\left(1 + \frac{1}{(\alpha+1)K_j} \cdot \frac{\phi_j}{\phi_j - 1}\left(1 - \frac{1}{\phi_j^{\alpha+1}}\right)\right).\end{aligned}$$

Thus, $\overline{B(\Delta)}$ is an average-case asymptotic performance bound for algorithm A. ■

In Table 4, we demonstrate numerical data for the asymptotic performance bound B in Theorem 3. Assume that $\alpha = 3$. For $\phi_j = 1.4, 1.8, 2.2, 2.6, 3.0$, we show the asymptotic performance bound for the asymptotic performance ratio β_A^{∞}, i.e., $B = (1-\Delta)^{\alpha-1}(1 + \Delta/K_j)$, for $\Delta = 0.05z(1-1/\phi_j)$, where $1 \le z \le 20$. We observe that the asymptotic performance bound B is an increasing function of Δ until B reaches its maximum value, and then B becomes a decreasing function of Δ as Δ further increases.

In Table 5, we demonstrate numerical data for the worst-case asymptotic performance bound B_{worst} in Corollary 2. For $\alpha = 3.0, 3.5, 4.0, 4.5, 5.0$, and $\phi = 1.1, 1.2, 1.3, \ldots, 3.0$, we show the worst-case asymptotic performance bound B_{worst} for the asymptotic performance ratio β_A^{∞}. We observe that the worst-case asymptotic performance bound B_{worst} is very close to optimal for $\alpha = 3$ and $\phi < 2$.

In Table 6, we demonstrate numerical data for the average-case asymptotic performance bound B_{average} in Corollary 2. For

Table 5
Numerical data for the worst-case asymptotic performance bound in Corollary 2.

ϕ	$\alpha = 3.0$	$\alpha = 3.5$	$\alpha = 4.0$	$\alpha = 4.5$	$\alpha = 5.0$
1.1	1.0068302	1.0099729	1.0136967	1.0180067	1.0229086
1.2	1.0251621	1.0368580	1.0508142	1.0670988	1.0857918
1.3	1.0526222	1.0774496	1.1073771	1.1427102	1.1838120
1.4	1.0876199	1.1297250	1.1811146	1.2426612	1.3154182
1.5	1.1290535	1.1923768	1.2707575	1.3661506	1.4809616
1.6	1.1761391	1.2645470	1.3756506	1.5132186	1.6819808
1.7	1.2283052	1.3456697	1.4955298	1.6844440	1.9208126
1.8	1.2851268	1.4353742	1.6303882	1.8807677	2.2003687
1.9	1.3462819	1.5334232	1.7803924	2.1033837	2.5240035
2.0	1.4115226	1.6396730	1.9458293	2.3536713	2.8954322
2.1	1.4806556	1.7540460	2.1270708	2.6331503	3.3186786
2.2	1.5535279	1.8765127	2.3245501	2.9434511	3.7980420
2.3	1.6300171	2.0070783	2.5387453	3.2862943	4.3380733
2.4	1.7100245	2.1457744	2.7701680	3.6634767	4.9435609
2.5	1.7934694	2.2926513	3.0193547	4.0768604	5.6195187
2.6	1.8802855	2.4477741	3.2868613	4.5283654	6.3711794
2.7	1.9704178	2.6112186	3.5732586	5.0199642	7.2039887
2.8	2.0638205	2.7830689	3.8791289	5.5536773	8.1236012
2.9	2.1604548	2.9634153	4.2050636	6.1315698	9.1358780
3.0	2.2602881	3.1523527	4.5516614	6.7557489	10.2468838

Table 6
Numerical data for the average-case asymptotic performance bound in Corollary 2.

ϕ	$\alpha = 3.0$	$\alpha = 3.5$	$\alpha = 4.0$	$\alpha = 4.5$	$\alpha = 5.0$
1.1	1.0045523	1.0066454	1.0091240	1.0119907	1.0152482
1.2	1.0167593	1.0245288	1.0337796	1.0445452	1.0568642
1.3	1.0350148	1.0514457	1.0711645	1.0943204	1.1210877
1.4	1.0582313	1.0859697	1.1195848	1.1595001	1.2062175
1.5	1.0856481	1.1271513	1.1780093	1.2391647	1.3117541
1.6	1.1167187	1.1743452	1.2458195	1.3329477	1.4379448
1.7	1.1510409	1.2271062	1.3226645	1.4408394	1.5855306
1.8	1.1883128	1.2851246	1.4083709	1.5630694	1.7555981
1.9	1.2283033	1.3481852	1.5028879	1.7000344	1.9494903
2.0	1.2708333	1.4161393	1.6062500	1.8522514	2.1687500
2.1	1.3157615	1.4888864	1.7185533	2.0203276	2.4150837
2.2	1.3629752	1.5663614	1.8399387	2.2049392	2.6903378
2.3	1.4123834	1.6485258	1.9705805	2.4068177	2.9964817
2.4	1.4639120	1.7353606	2.1106780	2.6267395	3.3355972
2.5	1.5175000	1.8268622	2.2604500	2.8655186	3.7098700
2.6	1.5730966	1.9230384	2.4201300	3.1240012	4.1215841
2.7	1.6306596	2.0239062	2.5899632	3.4030620	4.5731177
2.8	1.6901531	2.1294895	2.7702042	3.7036010	5.0669401
2.9	1.7515468	2.2398178	2.9611148	4.0265408	5.6056090
3.0	1.8148148	2.3549248	3.1629630	4.3728256	6.1917695

$\alpha = 3.0, 3.5, 4.0, 4.5, 5.0$, and $\phi = 1.1, 1.2, 1.3, \ldots, 3.0$, we show the average-case asymptotic performance bound B_{average} for the asymptotic performance ratio β_A^∞. We observe that the average-case asymptotic performance bound B_{average} is noticeably lower than the worst-case asymptotic performance bound B_{worst}, especially for large α and ϕ.

4.2. Scheduling on multiprocessor computers

Now, we consider a multiprocessor computer with $m > 1$ processors. Again, our scheduling algorithm A_1–A_2 to solve the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer with $m > 1$ processors contains two components, namely, a list scheduling algorithm A_1 which is used to produce a partition $R_1, R_2, \ldots, R_m$, and a modified list placement algorithm A_2 which is used to produce a schedule of tasks in R_k on processor k with time constraint T, where $1 \le k \le m$.

The following result gives a worst-case asymptotic performance bound for algorithm A_1–A_2.

Theorem 4. *For any list scheduling algorithm A_1 and any list placement algorithm A_2, the asymptotic performance ratio of algorithm A_1–A_2 is $\beta_{A_1-A_2}^\infty \le B$, where the worst-case asymptotic performance bound is*

$$B = \frac{(\alpha-1)^{\alpha-1}}{\alpha^\alpha} \cdot \frac{(K+1)^\alpha}{K},$$

and

$$K = \frac{\phi - 1}{\phi(\phi^{\alpha-1} - 1)},$$

with $\phi = \max\{\phi_1, \phi_2, \ldots, \phi_d\}$.

Proof. It has been known from [50] that for a given partition $R_1, R_2, \ldots, R_m$ of the n tasks into m groups generated by any list scheduling algorithm A_1, if processors have continuous clock frequency and supply voltage and execution speed and power levels, the minimum energy consumption on processor k is

$$\tilde{E}_k = \frac{R_k^\alpha}{T^{\alpha-1}},$$

where $1 \le k \le m$, and the minimum energy consumed by all the n tasks to meet the time deadline T is

$$\tilde{E} = \frac{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha}{T^{\alpha-1}}.$$

Let $E_{A_2,k}$ denote the energy consumed by algorithm A_2 on processor k with time constraint T, where $1 \le k \le m$. In the proof of Theorem 3, we have already shown that for any list placement algorithm A_2, we have

$$\lim_{R_k/r^* \to \infty} \frac{E_{A_2,k}}{\tilde{E}_k} = (1-\Delta_k)^{\alpha-1}\left(1 + \frac{\Delta_k}{K_{j_k}}\right),$$

where we assume that $T_{k,j_k} > T \ge T_{k,j_k+1}$, with $T_{k,j_k} = R_k/s_{j_k}$, and $1 \le j_k \le d-1$, and $T = T_{k,j_k}(1-\Delta_k) = (R_k/s_{j_k})(1-\Delta_k)$, where $0 < \Delta_k \le 1 - 1/\phi_{j_k}$, for all $1 \le k \le m$. Furthermore, by Corollary 2 and a similar argument in the proof of Theorem 2, we know that

$$\lim_{R_k/r^* \to \infty} \frac{E_{A_2,k}}{\tilde{E}_k} \le \frac{(\alpha-1)^{\alpha-1}}{\alpha^\alpha} \cdot \frac{(K+1)^\alpha}{K},$$

for all $1 \le k \le m$, where

$$K = \min\{K_1, K_2, \ldots, K_d\} = \frac{\phi - 1}{\phi(\phi^{\alpha-1} - 1)},$$

with $\phi = \max\{\phi_1, \phi_2, \ldots, \phi_d\}$.

Notice that the total amount of energy consumed by algorithm A_1–A_2 is

$$E_{A_1-A_2} = E_{A_2,1} + E_{A_2,2} + \cdots + E_{A_2,m}.$$

Since

$$\begin{aligned}
\lim_{R/r^* \to \infty} \frac{E_{A_2,k}}{\tilde{E}} &= \frac{\tilde{E}_k}{\tilde{E}} \lim_{R_k/r^* \to \infty} \frac{E_{A_2,k}}{\tilde{E}_k} \\
&= \frac{R_k^\alpha}{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha} \lim_{R_k/r^* \to \infty} \frac{E_{A_2,k}}{\tilde{E}_k} \\
&\le \frac{R_k^\alpha}{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha} \cdot \frac{(\alpha-1)^{\alpha-1}}{\alpha^\alpha} \cdot \frac{(K+1)^\alpha}{K},
\end{aligned}$$

for all $1 \le k \le m$, we have

$$\begin{aligned}
\lim_{R/r^* \to \infty} \frac{E_{A_1-A_2}}{\tilde{E}} &= \lim_{R/r^* \to \infty} \frac{E_{A_2,1} + E_{A_2,2} + \cdots + E_{A_2,m}}{\tilde{E}} \\
&= \sum_{k=1}^m \lim_{R/r^* \to \infty} \frac{E_{A_2,k}}{\tilde{E}} \\
&\le \sum_{k=1}^m \frac{R_k^\alpha}{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha}
\end{aligned}$$

$$\cdot \frac{(\alpha-1)^{\alpha-1}}{\alpha^{\alpha}} \cdot \frac{(K+1)^{\alpha}}{K} = \frac{(\alpha-1)^{\alpha-1}}{\alpha^{\alpha}} \cdot \frac{(K+1)^{\alpha}}{K}.$$

Let E_{OPT} denote the minimum energy consumption when processors have discrete clock frequency and supply voltage and execution speed and power levels, and $\tilde{E}_{\text{OPT}}$ denote the minimum energy consumption when processors have continuous clock frequency and supply voltage and execution speed and power levels. Clearly, we have $E_{\text{OPT}} \geq \tilde{E}_{\text{OPT}}$. It has been proven in [50] that for any list scheduling algorithm A_1, as $R/r^* \to \infty$, we have $\tilde{E}/\tilde{E}_{\text{OPT}} \to 1$. Consequently, the asymptotic performance ratio of algorithm A_1–A_2 is

$$\begin{aligned}\beta^{\infty}_{A_1-A_2} &= \lim_{R/r^*\to\infty} \frac{E_{A_1-A_2}}{E_{\text{OPT}}} \\ &\leq \lim_{R/r^*\to\infty} \frac{E_{A_1-A_2}}{\tilde{E}_{\text{OPT}}} \\ &= \left(\lim_{R/r^*\to\infty} \frac{E_{A_1-A_2}}{\tilde{E}}\right)\left(\lim_{R/r^*\to\infty} \frac{\tilde{E}}{\tilde{E}_{\text{OPT}}}\right) \\ &\leq \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}} \cdot \frac{(K+1)^{\alpha/(\alpha-1)}}{K^{1/(\alpha-1)}}.\end{aligned}$$

This proves the theorem. ■

5. Simulation results

In this section, we present simulation results for the nine algorithms developed in this paper, which are SRF-SOF, SRF-LP, SRF-LOF, LS-SOF, LS-LP, LS-LOF, LRF-SOF, LRF-LP, and LRF-LOF. Our experimental performance evaluation is based on two performance measures, namely, normalized schedule length and normalized energy consumption.

The *normalized schedule length* NSL_A of an algorithm A that solves the problem of minimizing schedule length with energy consumption constraint is defined as

$$\text{NSL}_A = \frac{T_A}{((m/E)(R/m)^{\alpha})^{1/(\alpha-1)}},$$

where the denominator is a lower bound for the optimal schedule length even when processors have continuous clock frequency and supply voltage and execution speed and power levels [50].

We notice that NSL_A serves as a performance bound for the performance ratio $\beta_A = T_A/T_{\text{OPT}}$ of any algorithm A that solves the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer. When the r_i's and E are random variables, T_A, T_{OPT}, β_A, and the NSL_A all become random variables. It is clear that for the problem of minimizing schedule length with energy consumption constraint, we have $\bar{\beta}_A \leq \overline{\text{NSL}}_A$, i.e., the expected performance ratio is no greater than the expected normalized schedule length.

The *normalized energy consumption* NEC_A of an algorithm A that solves the problem of minimizing energy consumption with schedule length constraint is defined as

$$\text{NEC}_A = \frac{E_A}{R^{\alpha}/(mT)^{\alpha-1}},$$

where the denominator is a lower bound for the minimum energy consumption even when processors have continuous clock frequency and supply voltage and execution speed and power levels [50].

It is noticed that NEC_A is a performance bound for the performance ratio $\beta_A = E_A/E_{\text{OPT}}$ of any algorithm A that solves the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer. It is also clear that for the problem of minimizing energy consumption with schedule length constraint, we have $\bar{\beta}_A \leq \overline{\text{NEC}}_A$, i.e., the expected performance ratio is no greater than the expected normalized schedule length.

Notice that for a given power allocation and task scheduling algorithm A, the expected normalized schedule length $\overline{\text{NSL}}_A$ and the expected normalized energy consumption $\overline{\text{NEC}}_A$ are determined by m, n, α, $s_1, s_2, \ldots, s_d$, the probability distribution of the r_i's, and the probability distribution of Δ. In our simulations, the number of processors is set as $m = 10$. The number of tasks is in the range $n = 10, 20, 30, \ldots, 200$. The parameter α is set as 3. As mentioned earlier, we assume that the number of speed levels is large enough to accommodate the needs our algorithms. Furthermore, we assume that $\phi_1 = \phi_2 = \cdots = \phi_{d-1} = \phi = 2$. For convenience, we assume that $s_j = \phi^j$ for $j = 0, \pm1, \pm2, \ldots$, such that the value of d is sufficiently large. The r_i's are independent and identically distributed random variables with a uniform distribution in [0, 1]. For the expected normalized schedule length $\overline{\text{NSL}}_A$, the energy constraint E is set as $E = Rs_j^{\alpha-1}(1+\Delta)$, where j is any value and Δ has a uniform distribution in $[0, \phi^{\alpha-1}-1)$. For the expected normalized energy consumption $\overline{\text{NEC}}_A$, the time constraint T is set as $T = ((R/m)/s_j)(1-\Delta)$ (R/m is approximately R_k), where j is any value and Δ has a uniform distribution in $(0, 1-1/\phi]$.

In Tables 7 and 8, we show our simulation results. For each combination of n and algorithm $A \in \{$ SRF-SOF, SRF-LP, SRF-LOF, LS-SOF, LS-LP, LS-LOF, LRF-SOF, LRF-LP, LRF-LOF$\}$, we generate 5000 sets of n tasks, produce their schedules by using algorithm A, calculate their NSL_A (or NEC_A), and report the average of NSL_A (or NEC_A), which is the experimental value of $\overline{\text{NSL}}_A$ (or $\overline{\text{NEC}}_A$). The 99% confidence interval (CI) of all the data in the same table is also given. We observe the following facts.

- The performance of all these algorithms improves as n increases. The expected normalized schedule length $\overline{\text{NSL}}_A$ (the expected normalized energy consumption $\overline{\text{NEC}}_A$, respectively) decreases as n increases, i.e., R/r^* increases, and eventually approaches the average-case asymptotic performance bound B_{average} whose exact value is analytically not available. (Notice that Tables 3 and 6 are only for uniprocessor computers.)
- For the expected normalized schedule length $\overline{\text{NSL}}_A$, the speed of convergence depends on algorithm A. It is clear that algorithm LRF leads to faster speed of convergence than LS and SRF, and algorithm LOF leads to faster speed of convergence than LP and SOF. Algorithm LRF-LOF performs the best among all these nine algorithms.
- For the expected normalized energy consumption $\overline{\text{NEC}}_A$, the speeds of convergence of all these algorithms are about the same.

Notice that for multiprocessor computers, there is no average-case performance analysis similar to that of Corollary 1. The reason is that after the energy allocation according to the proof of Theorem 2, it is impossible to have a uniform distribution for Δ_k, where $1 \leq k \leq m$. Even though we know the distribution of E, we do not know the distribution of E_k. Similarly, for multiprocessor computers, there is no average-case performance analysis similar to that of Corollary 2. It is impossible to provide T such that Δ_k in the proof of Theorem 4 has a uniform distribution for all $1 \leq k \leq m$.

6. Concluding remarks

We have addressed energy and time constrained task scheduling on multiprocessor computers with discrete clock frequency and supply voltage and execution speed and power levels. We

Table 7
Simulation results for the expected NSL (99% CI = ±0.548%).

n	SRF-SOF	SRF-LP	SRF-LOF	LS-SOF	LS-LP	LS-LOF	LRF-SOF	LRF-LP	LRF-LOF
10	2.6627937	2.6531092	2.6651033	2.6506262	2.6514494	2.6621895	2.6606938	2.6504537	2.6590911
20	1.9073127	1.8607263	1.7933438	1.8379552	1.8165155	1.7889429	1.6230521	1.6127421	1.5995227
30	1.5780350	1.5125698	1.3969578	1.5464532	1.5119938	1.4675845	1.4195489	1.4037548	1.3348711
40	1.4509819	1.3871110	1.2896858	1.4382028	1.3980687	1.3480303	1.3372592	1.3149214	1.2395246
50	1.3854113	1.3223044	1.2473871	1.3729157	1.3305447	1.2825567	1.2849894	1.2672456	1.1977052
60	1.3403008	1.2815090	1.2222159	1.3315909	1.2857138	1.2427048	1.2523527	1.2376041	1.1767710
70	1.3094400	1.2554527	1.2059391	1.3009057	1.2583646	1.2184541	1.2299720	1.2184488	1.1617892
80	1.2876245	1.2353795	1.1957224	1.2795055	1.2384573	1.2027791	1.2162788	1.2040222	1.1545888
90	1.2706625	1.2211495	1.1857643	1.2631078	1.2218291	1.1893480	1.2057288	1.1922161	1.1481900
100	1.2571234	1.2099720	1.1791448	1.2480708	1.2099745	1.1808149	1.1956781	1.1843063	1.1426318
110	1.2453771	1.2000902	1.1748636	1.2389628	1.2006062	1.1734898	1.1889156	1.1768732	1.1389374
120	1.2349509	1.1925040	1.1709599	1.2272486	1.1925177	1.1690677	1.1819901	1.1710082	1.1382297
130	1.2275895	1.1870073	1.1670815	1.2195753	1.1862007	1.1646078	1.1763297	1.1667280	1.1367946
140	1.2191585	1.1809062	1.1635122	1.2136861	1.1802097	1.1599714	1.1721043	1.1621538	1.1334180
150	1.2130956	1.1777033	1.1617430	1.2076149	1.1755992	1.1587206	1.1691728	1.1588820	1.1340065
160	1.2096720	1.1737784	1.1591710	1.2014031	1.1719383	1.1555972	1.1655089	1.1549036	1.1310444
170	1.2030821	1.1700972	1.1568725	1.1979739	1.1688821	1.1535698	1.1639257	1.1535769	1.1327832
180	1.2001909	1.1663476	1.1553638	1.1937645	1.1660122	1.1505158	1.1616800	1.1514544	1.1316846
190	1.1954581	1.1643199	1.1535240	1.1917164	1.1626359	1.1504688	1.1589337	1.1494556	1.1318445
200	1.1929093	1.1632193	1.1528457	1.1874242	1.1601478	1.1478626	1.1568164	1.1463886	1.1292879

Table 8
Simulation results for the expected NEC (99% CI = ±1.415%).

n	SRF-SOF	SRF-LP	SRF-LOF	LS-SOF	LS-LP	LS-LOF	LRF-SOF	LRF-LP	LRF-LOF
10	4.4035621	4.4268208	4.4273344	4.4118209	4.4345747	4.3931841	4.4046844	4.4119929	4.4097261
20	2.2082212	2.2293548	2.2487686	2.0943158	2.1063697	2.1172441	1.8949531	1.9231441	1.9520099
30	1.7926217	1.8131297	1.8277041	1.7435763	1.7652009	1.7837458	1.6562990	1.7032158	1.7233051
40	1.6292817	1.6415019	1.6636407	1.6036847	1.6172160	1.6357266	1.5460615	1.5794236	1.6096659
50	1.5387299	1.5523530	1.5717763	1.5224977	1.5391586	1.5528148	1.4887687	1.5151398	1.5461654
60	1.4860855	1.4967544	1.5129503	1.4748531	1.4909989	1.5037466	1.4533972	1.4728249	1.4934220
70	1.4484483	1.4652334	1.4740507	1.4407390	1.4572177	1.4679866	1.4277042	1.4463159	1.4672424
80	1.4222806	1.4371573	1.4479055	1.4161548	1.4306426	1.4399494	1.3992306	1.4213839	1.4426356
90	1.4061707	1.4145159	1.4275958	1.4006336	1.4083174	1.4247135	1.3898933	1.4050381	1.4217626
100	1.3885308	1.3988607	1.4121022	1.3851371	1.3979123	1.4071813	1.3786933	1.3900597	1.4035455
110	1.3777139	1.3851678	1.3973236	1.3766578	1.3833436	1.3931982	1.3659468	1.3797409	1.3931678
120	1.3659509	1.3751669	1.3864855	1.3637074	1.3734461	1.3859370	1.3614404	1.3710160	1.3848408
130	1.3553616	1.3678461	1.3782386	1.3552720	1.3650039	1.3764768	1.3501051	1.3624730	1.3713060
140	1.3519278	1.3600004	1.3678148	1.3471772	1.3602783	1.3685951	1.3445761	1.3573125	1.3665716
150	1.3432114	1.3536814	1.3591455	1.3454590	1.3525593	1.3616739	1.3409090	1.3504305	1.3592882
160	1.3398802	1.3484211	1.3568213	1.3391282	1.3453472	1.3536476	1.3352940	1.3453661	1.3563311
170	1.3384126	1.3433291	1.3517670	1.3390302	1.3449947	1.3496369	1.3323613	1.3376515	1.3500502
180	1.3328860	1.3371342	1.3469822	1.3334055	1.3377823	1.3439984	1.3261517	1.3350548	1.3420405
190	1.3301356	1.3388385	1.3423936	1.3283203	1.3348592	1.3410373	1.3235238	1.3328611	1.3389523
200	1.3256016	1.3312320	1.3370636	1.3223971	1.3342259	1.3380277	1.3225960	1.3309612	1.3355932

considered the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint. We proved that both problems are NP-hard even on a uniprocessor computer with only two speed levels. We developed a type of algorithms to solve the above problems. Such an algorithm contains two components, namely, a list scheduling algorithm for task scheduling and a list placement algorithm for speed determination. We derived a worst-case asymptotic performance bound and an average-case asymptotic performance bound for our algorithms on uniprocessor computers, and a worst-case asymptotic performance bound on multiprocessor computers. We also conducted extensive simulations to verify our analytical results. We found that our algorithms produce near-optimal solutions and are practically very useful.

References

[1] S. Albers, Energy-efficient algorithms, Commun. ACM 53 (5) (2010) 86–96.
[2] J. Augustine, S. Irani, C. Swamy, Optimal power-down strategies, SIAM J. Comput. 37 (5) (2008) 1499–1516.
[3] H. Aydin, R. Melhem, D. Mossé, P. Mejía-Alvarez, Power-aware scheduling for periodic real-time tasks, IEEE Trans. Comput. 53 (5) (2004) 584–600.
[4] E. Bampis, C. Dürr, F. Kacem, I. Milis, Speed scaling with power down scheduling for agreeable deadlines, Sustain. Comput.: Inform. Syst. 2 (4) (2012) 184–189.
[5] N. Bansal, T. Kimbrel, K. Pruhs, Speed scaling to manage energy and temperature, J. ACM 54 (1) (2007) article no. 3.
[6] P. Baptiste, M. Chrobak, C. Dürr, Polynomial-time algorithms for minimum energy scheduling, ACM Trans. Algorithms 8 (3) (2012) article no. 26.
[7] J.A. Barnett, Dynamic task-level voltage scheduling optimizations, IEEE Trans. Comput. 54 (5) (2005) 508–520.
[8] A. Beloglazov, R. Buyya, Y.C. Lee, A. Zomaya, A taxonomy and survey of energy-efficient data centers and cloud computing systems, Adv. Comput. 82 (2011) 47–111.
[9] L. Benini, A. Bogliolo, G. De Micheli, A survey of design techniques for system-level dynamic power management, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 8 (3) (2000) 299–316.
[10] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M.Q. Dang, K. Pentikousis, Energy-efficient cloud computing, Comput. J. 53 (7) (2010) 1045–1051.
[11] D.P. Bunde, Power-aware scheduling for makespan and flow, in: Proceedings of the 18th ACM Symposium on Parallelism in Algorithms and Architectures, 2006, pp. 190–196.
[12] H.-L. Chan, W.-T. Chan, T.-W. Lam, L.-K. Lee, K.-S. Mak, P.W.H. Wong, Energy efficient online deadline scheduling, in: Proceedings of the 18th ACM–SIAM Symposium on Discrete Algorithms, 2007, pp. 795–804.
[13] A.P. Chandrakasan, S. Sheng, R.W. Brodersen, Low-power CMOS digital design, IEEE J. Solid-State Circuits 27 (4) (1992) 473–484.
[14] S. Cho, R.G. Melhem, On the interplay of parallelization, program performance, and energy consumption, IEEE Trans. Parallel Distrib. Syst. 21 (3) (2010) 342–353.
[15] V. Devadas, H. Aydin, On the interplay of voltage/frequency scaling and device power management for frame-based real-time embedded applications, IEEE Trans. Comput. 61 (1) (2012) 31–44.

[16] D. Donofrio, L. Oliker, J. Shalf, M.F. Wehner, C. Rowen, J. Krueger, S. Kamil, M. Mohiyuddin, Energy-efficient computing for extreme-scale science, Computer 42 (11) (2009) 62–71.
[17] W.-C. Feng, The importance of being low power in high performance computing, CTWatchQuarterly 1 (3) (2005) Los Alamos National Laboratory.
[18] W.-c. Feng, K.W. Cameron, The green500 list: encouraging sustainable supercomputing, Computer 40 (12) (2007) 50–55.
[19] V.W. Freeh, D.K. Lowenthal, F. Pan, N. Kappiah, R. Springer, B.L. Rountree, M.E. Femal, Analyzing the energy–time trade-off in high-performance computing applications, IEEE Trans. Parallel Distrib. Syst. 18 (6) (2007) 835–848.
[20] A. Gara, et al., Overview of the Blue Gene/L system architecture, IBM J. Res. Dev. 49 (2–3) (2005) 195–212.
[21] M.R. Garey, D.S. Johnson, Computers and Intractability—A Guide to the Theory of NP-Completeness, W. H. Freeman, New York, 1979.
[22] S.K. Garg, C.S. Yeo, A. Anandasivam, R. Buyya, Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers, J. Parallel Distrib. Comput. 71 (6) (2011) 732–749.
[23] M.E.T. Gerards, Algorithmic power management—energy minimization under real-time constraints (Ph.D. thesis), University of Twente, Netherlands, 2014.
[24] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (2) (1969) 416–429.
[25] S.L. Graham, M. Snir, C.A. Patterson (Eds.), Getting up to speed: the future of supercomputing, in: Committee on the Future of Supercomputing, in: National Research Council, National Academies Press, 2005.
[26] J.-J. Han, X. Wu, D. Zhu, H. Jin, L.T. Yang, J.-L. Gaudiot, Synchronization-aware energy management for VFI-based multicore real-time systems, IEEE Trans. Comput. 61 (12) (2012) 1682–1696.
[27] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, M.B. Srivastava, Power optimization of variable-voltage core-based systems, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 18 (12) (1999) 1702–1714.
[28] http://en.wikipedia.org/wiki/Dynamic_voltage_scaling.
[29] http://en.wikipedia.org/wiki/LongHaul.
[30] http://en.wikipedia.org/wiki/LongRun.
[31] http://en.wikipedia.org/wiki/Moore's_law.
[32] http://en.wikipedia.org/wiki/SpeedStep.
[33] http://science.energy.gov/~/media/ascr/ascac/pdf/meetings/20140210/Top10reportFEB14.pdf.
[34] http://www.eia.gov/electricity/monthly/epm_table_grapher.cfm?t=epmt_5_6_a.
[35] http://www.green500.org/lists/green201506.
[36] http://www.tomshardware.com/reviews/cpu-performance-comparison,3370-17.html.
[37] http://www.top500.org/lists/2015/06/.
[38] https://www.whitehouse.gov/the-press-office/2015/07/29/executive-order-creating-national-strategic-computing-initiative.
[39] F. Hu, J.J. Evans, Power and environment aware control of Beowulf clusters, Cluster Comput. 12 (3) (2009) 299–308.
[40] C. Im, S. Ha, H. Kim, Dynamic voltage scheduling with buffers in low-power multimedia applications, ACM Trans. Embedded Comput. Syst. 3 (4) (2004) 686–705.
[41] Intel, Enhanced Intel SpeedStep Technology for the Intel Pentium M Processor—White Paper, March 2004.
[42] S. Irani, S. Shukla, R. Gupta, Algorithms for power savings, ACM Trans. Algorithms 3 (4) (2007) article no. 41.
[43] S.U. Khan, I. Ahmad, A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids, IEEE Trans. Parallel Distrib. Syst. 20 (3) (2009) 346–360.
[44] C.M. Krishna, Y.-H. Lee, Voltage-clock-scaling adaptive scheduling techniques for low power in hard real-time systems, IEEE Trans. Comput. 52 (12) (2003) 1586–1593.
[45] W.-C. Kwon, T. Kim, Optimal voltage allocation techniques for dynamically variable voltage processors, ACM Trans. Embedded Comput. Syst. 4 (1) (2005) 211–230.
[46] Y.-H. Lee, C.M. Krishna, Voltage-clock scaling for low energy consumption in fixed-priority real-time systems, Real-Time Syst. 24 (3) (2003) 303–317.
[47] W.-K. Lee, S.-W. Lee, W.-O. Siew, Hybrid model for dynamic power management, IEEE Trans. Consum. Electron. 55 (2) (2009) 656–664.
[48] Y.C. Lee, A.Y. Zomaya, Energy conscious scheduling for distributed computing systems under different operating conditions, IEEE Trans. Parallel Distrib. Syst. 22 (8) (2011) 1374–1381.
[49] K. Li, Average-case performance analysis of an approximation algorithm for maximum subset sum using recurrence relations, Comput. Math. Appl. 36 (6) (1998) 63–75.
[50] K. Li, Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed, IEEE Trans. Parallel Distrib. Syst. 19 (11) (2008) 1484–1497.
[51] K. Li, Algorithms and analysis of energy-efficient scheduling of parallel tasks, in: I. Ahmad, S. Ranka (Eds.), Handbook of Energy-Aware and Green Computing, Vol. 1, CRC Press/Taylor & Francis Group, 2012, pp. 331–360. (Chapter 15).
[52] K. Li, Energy efficient scheduling of parallel tasks on multiprocessor computers, J. Supercomput. 60 (2) (2012) 223–247.
[53] K. Li, Power allocation and task scheduling on multiprocessor computers with energy and time constraints, in: A.Y. Zomaya, Y.C. Lee (Eds.), Energy-Efficient Distributed Computing Systems, John Wiley & Sons, 2012, pp. 1–37. (Chapter 1).
[54] K. Li, Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers, IEEE Trans. Comput. 61 (12) (2012) 1668–1681.
[55] K. Li, Energy-efficient and high-performance processing of large-scale parallel applications in data centers, in: S.U. Khan, A.Y. Zomaya (Eds.), Data Centers, Springer, 2015, pp. 1–33. (Chapter 1).
[56] K. Li, Power and performance management for parallel computations in clouds and data centers, J. Comput. System Sci. 82 (2016) 174–190.
[57] M. Li, B.J. Liu, F.F. Yao, Min-energy voltage allocation for tree-structured tasks, J. Comb. Optim. 11 (2006) 305–319.
[58] K. Li, X. Tang, K. Li, Energy-efficient stochastic task scheduling on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 2867–2876.
[59] M. Li, F.F. Yao, An efficient algorithm for computing optimal discrete voltage schedules, SIAM J. Comput. 35 (3) (2006) 658–671.
[60] M. Li, A.C. Yao, F.F. Yao, Discrete and continuous min-energy schedules for variable voltage processors, Proc. Natl. Acad. Sci. USA 103 (11) (2006) 3983–3987.
[61] J.R. Lorch, A.J. Smith, PACE: a new approach to dynamic voltage scaling, IEEE Trans. Comput. 53 (7) (2004) 856–869.
[62] G. Lovász, F. Niedermeier, H. de Meer, Performance tradeoffs of energy-aware virtual machine consolidation, Cluster Comput. 16 (3) (2013) 481–496.
[63] R.N. Mahapatra, W. Zhao, An energy-efficient slack distribution technique for multimode distributed real-time embedded systems, IEEE Trans. Parallel Distrib. Syst. 16 (7) (2005) 650–662.
[64] M. Marinoni, G. Buttazzo, Elastic DVS management in processors with discrete voltage/frequency modes, IEEE Trans. Ind. Inf. 3 (1) (2007) 51–62.
[65] J. Mei, K. Li, K. Li, Energy-aware task scheduling in heterogeneous computing environments, Cluster Comput. 17 (2) (2014) 537–550.
[66] B.C. Mochocki, X.S. Hu, G. Quan, A unified approach to variable voltage scheduling for nonideal DVS processors, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 23 (9) (2004) 1370–1377.
[67] V.A. Patil, V. Chaudhary, Rack aware scheduling in HPC data centers: an energy conservation strategy, Cluster Comput. 16 (3) (2013) 559–573.
[68] K. Pruhs, R. van Stee, P. Uthaisombut, Speed scaling of tasks with precedence constraints, in: T. Erlebach, G. Persinao (Eds.), Approximation and Online Algorithms, in: Lecture Notes in Computer Science, vol. 3879, Springer-Verlag, Berlin, Heidelberg, 2006, pp. 307–319.
[69] G. Qu, What is the limit of energy saving by dynamic voltage scaling, in: Proceedings of the International Conference on Computer-Aided Design, 2001, pp. 560–563.
[70] G. Quan, X.S. Hu, Energy efficient DVS schedule for fixed-priority real-time systems, ACM Trans. Embedded Comput. Syst. 6 (4) (2007) Article no. 29.
[71] N.B. Rizvandi, J. Taheri, A.Y. Zomaya, Some observations on optimal frequency selection in DVFS-based energy consumption minimization, J. Parallel Distrib. Comput. 71 (8) (2011) 1154–1164.
[72] C. Rusu, R. Melhem, D. Mossé, Maximizing rewards for real-time applications with energy constraints, ACM Trans. Embedded Comput. Syst. 2 (4) (2003) 537–559.
[73] D. Shin, J. Kim, Power-aware scheduling of conditional task graphs in real-time multiprocessor systems, in: Proceedings of the International Symposium on Low Power Electronics and Design, 2003, pp. 408–413.
[74] D. Shin, J. Kim, S. Lee, Intra-task voltage scheduling for low-energy hard real-time applications, IEEE Des. Test Comput. 18 (2) (2001) 20–30.
[75] M.B. Srivastava, A.P. Chandrakasan, R.W. Rroderson, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 4 (1) (1996) 42–55.
[76] M.R. Stan, K. Skadron, Guest editors' introduction: power-aware computing, IEEE Comput. 36 (12) (2003) 35–38.
[77] O.S. Unsal, I. Koren, System-level power-aware design techniques in real-time systems, Proc. IEEE 91 (7) (2003) 1055–1069.
[78] G.L. Valentini, W. Lassonde, S.U. Khan, N. Min-Allah, S.A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A.Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J.E. Pecero, D. Kliazovich, P. Bouvry, An overview of energy efficiency techniques in cluster computing systems, Cluster Comput. 16 (1) (2013) 3–15.
[79] V. Venkatachalam, M. Franz, Power reduction techniques for microprocessor systems, ACM Comput. Surv. 37 (3) (2005) 195–237.
[80] M. Weiser, B. Welch, A. Demers, S. Shenker, Scheduling for reduced CPU energy, in: Proceedings of the 1st USENIX Symposium on Operating Systems Design and Implementation, 1994, pp. 13–23.
[81] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, R. Lauwereins, Energy-aware runtime scheduling for embedded-multiprocessor SOCs, IEEE Des. Test Comput. 18 (5) (2001) 46–58.
[82] F. Yao, A. Demers, S. Shenker, A scheduling model for reduced CPU energy, in: Proceedings of the 36th IEEE Symposium on Foundations of Computer Science, 1995, pp. 374–382.
[83] H.-S. Yun, J. Kim, On energy-optimal voltage scheduling for fixed-priority hard real-time systems, ACM Trans. Embedded Comput. Syst. 2 (3) (2003) 393–430.
[84] B. Zhai, D. Blaauw, D. Sylvester, K. Flautner, Theoretical and practical limits of dynamic voltage scaling, in: Proceedings of the 41st Design Automation Conference, 2004, pp. 868–873.

[85] L.M. Zhang, K. Li, D.C.-T. Lo, Y. Zhang, Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds, Sustain. Comput.: Inform. Syst. 3 (2) (2013) 109–118.
[86] L. Zhang, K. Li, Y. Xu, F. Zhang, K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, Inform. Sci. 319 (2015) 113–131.
[87] X. Zhong, C.-Z. Xu, Energy-aware modeling and scheduling for dynamic voltage scaling with statistical real-time guarantee, IEEE Trans. Comput. 56 (3) (2007) 358–372.
[88] D. Zhu, R. Melhem, B.R. Childers, Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems, IEEE Trans. Parallel Distrib. Syst. 14 (7) (2003) 686–700.
[89] D. Zhu, D. Mossé, R. Melhem, Power-aware scheduling for AND/OR graphs in real-time systems, IEEE Trans. Parallel Distrib. Syst. 15 (9) (2004) 849–864.
[90] J. Zhuo, C. Chakrabarti, Energy-efficient dynamic task scheduling algorithms for DVS systems, ACM Trans. Embedded Comput. Syst. 7 (2) (2008) Article no. 17.
[91] S. Zhuravlev, J.C. Saez, S. Blagodurov, A. Fedorova, M. Prieto, Survey of energy-cognizant scheduling techniques, IEEE Trans. Parallel Distrib. Syst. 24 (7) (2013) 1447–1464.
[92] Z. Zong, A. Manzanares, X. Ruan, X. Qin, EAD and PEBD: two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters, IEEE Trans. Comput. 60 (3) (2011) 360–374.

Keqin Li is a SUNY Distinguished Professor of computer science in the State University of New York. He is also a Distinguished Professor of Chinese National Recruitment Program of Global Experts (1000 Plan) at Hunan University, China. He was an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China, during 2011–2014. His current research interests include parallel computing and high-performance computing, distributed computing, energy-efficient computing and communication, heterogeneous computing systems, cloud computing, big data computing, CPU–GPU hybrid and cooperative computing, multicore computing, storage and file systems, wireless communication networks, sensor networks, peer-to-peer file sharing systems, mobile computing, service computing, Internet of things and cyber–physical systems. He has published over 400 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently or has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *Journal of Parallel and Distributed Computing*, *International Journal of Parallel, Emergent and Distributed Systems*, *International Journal of High Performance Computing and Networking, Optimization Letters, and International Journal of Big Data Intelligence*. He is an IEEE Fellow.