J. Parallel Distrib. Comput. 123 (2019) 13–25

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Optimal task execution speed setting and lower bound for delay and energy minimization

Keqin Li

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

HIGHLIGHTS

- Investigate scheduling a set of independent sequential tasks on identical processors.
- Find the optimal task execution speed setting analytically for delay and energy minimization.
- Establish lower bound for the minimum schedule length with a given energy consumption constraint.
- Establish lower bound for the minimum energy consumption with a given schedule length constraint.
- Perform experimental study on the performance of list scheduling algorithms.

ARTICLE INFO

Article history:
Received 22 November 2017
Received in revised form 7 July 2018
Accepted 3 September 2018
Available online xxxx

Keywords:
Delay minimization
Energy minimization
Energy-efficient task scheduling
List scheduling
Lower bound
Optimal speed setting
Static power consumption

ABSTRACT

The current technology trend reveals that static power consumption is growing at a faster rate than dynamic power consumption. In this paper, energy-efficient task scheduling is studied when static power consumption is a significant part of energy consumption which cannot be ignored. The problems of scheduling a set of independent sequential tasks on identical processors so that the schedule length is minimized for a given energy consumption constraint or the energy consumption is minimized for a given schedule length constraint are investigated. For a given schedule, the optimal task execution speed setting for delay and energy minimization is found analytically. Lower bounds for the minimum schedule length of a set of tasks with a given energy consumption constraint and the minimum energy consumption of a set of tasks with a given schedule length constraint are established. Our lower bounds are applicable to sequential or parallel, and independent or precedence constrained tasks, on processors with discrete or continuous speed levels, and bounded or unbounded speed ranges. The significance of these lower bounds is that they can be used to evaluate the performance of any heuristic algorithms when compared with optimal algorithms. Experimental study on the performance of list scheduling algorithms is performed and it is shown that their performance is very close to the optimal. To the best of the author's knowledge, this is the first paper that provides such analytical results for energy-efficient task scheduling with both dynamic and static power consumptions.

1. Introduction

Reducing processor energy consumption has been a significant research issue in the last two decades, and a huge body of literature has been published [1,29,40]. Processor power consumption includes two components, i.e., dynamic power consumption and static power consumption. It was believed that dynamic power consumption is the dominant part of processor energy consumption, and some research ignored static power consumption [15,39].

However, as transistors become smaller and faster, static power dissipation (i.e., the power due to leakage current in the absence of any switching activity) has become increasingly significant. Because leakage current flows from every transistor that is powered on, with increasing die size and integration, static power will become a significant part of processor power consumption. Static power dissipation is equal to the product of the supply voltage and the leakage current. While the rate of reduction of supply voltage is decreasing, leakage current is increasing exponentially [5]. The current technology trend reveals that static power consumption is growing at a faster rate than dynamic power consumption. Leakage current increases about 7.5 times and leakage power increases about 5.0 times every generation, while active power remains roughly constant [4]. In just a few processor generations, the curves will intersect. Technology scaling is increasing both the absolute and relative contributions of static power dissipation [28]. Static power consumption has noticeable influence on energy consumption and energy-delay product (EDP) [24]. It was demonstrated that if static power consumption is tuned during designing and

E-mail address: lik@newpaltz.edu.

https://doi.org/10.1016/j.jpdc.2018.09.003
0743-7315/© 2018 Elsevier Inc. All rights reserved.

manufacturing, it is possible to save up to 35% reduction in energy consumption and achieve up to 20% improvement in the EDP [23].

One major challenge in the study of energy-efficient task scheduling algorithms is lack of performance analysis and comparison between a heuristic solution and an optimal solution, as traditionally conducted in scheduling theory [8] and other areas of approximation algorithms for NP-hard problems [12]. The main weakness of most existing researches is that they only compare the performance of heuristic algorithms among the algorithms themselves, not with an optimal algorithm [3,10,14,25,30–35,38,41]. Furthermore, there is little analytical result on the worst-case or average-case performance ratio, although some attempt has been made without consideration of static power dissipation [15,16,19,20,26]. This is essentially due to the sophistication of energy-efficient task scheduling algorithms and the apparent lack of the understanding of optimal solutions.

To tackle the above challenge and weakness, one effective approach has been developed in [15,16], i.e., establishing a lower bound for the optimal solution and comparing a heuristic solution with the lower bound. The advantages of a lower bound are two fold. First, it is easy to obtain based on just a few parameters, and thus can be easily incorporated into any scheduler in a real system. Second, we can still assess the performance of a heuristic algorithm when compared with an optimal algorithm even we do not know the optimal solution. If the ratio of a heuristic solution to the lower bound is close to one, the performance of a heuristic algorithm is close to the optimal. Even though a performance ratio cannot be derived, it can be obtained experimentally by simulations or numerically by calculations. This method has been successful in studying the performance of various energy-efficient algorithms in scheduling sequential or parallel tasks, and independent or precedence constrained tasks [15–21]. However, such an effort has been effective only when the static power consumption is ignored.

The motivation of this paper is to make further progress towards this direction when static power consumption is a significant part of energy consumption which cannot be ignored. The main contributions of the paper are as follows.

- We investigate the problems of scheduling a set of independent sequential tasks on identical processors so that the schedule length is minimized for a given energy consumption constraint or the energy consumption is minimized for a given schedule length constraint. In particular, for a given schedule, we are able to find the optimal task execution speed setting analytically for delay and energy minimization.
- We establish lower bounds for the minimum schedule length of a set of tasks with a given energy consumption constraint and the minimum energy consumption of a set of tasks with a given schedule length constraint. Our lower bounds are applicable to sequential or parallel, and independent or precedence constrained tasks, on processors with discrete or continuous speed levels, and bounded or unbounded speed ranges. The significance of these lower bounds is that they can be used to evaluate the performance of any heuristic algorithms when compared with optimal algorithms.
- We perform experimental study on the performance of list scheduling algorithms and show that their performance is very close to the optimal.

To the best of the author's knowledge, this is the first paper that provides such analytical results for energy-efficient task scheduling with both dynamic and static power consumptions. All researchers in this area can benefit from our work in the sense that they can compare the performance of their heuristic algorithms with the lower bounds derived in this paper.

The paper is organized as follows. In Section 2, we present preliminary information, including the power consumption model and problem definitions. In Section 3, we develop lower bound for delay minimization, i.e., minimizing schedule length with energy consumption constraint. In Section 4, we develop lower bound for energy minimization, i.e., minimizing energy consumption with schedule length constraint. In Section 5, we demonstrate experimental data for some heuristic algorithms. In Section 6, we extend our lower bounds to parallel tasks and precedence constrained tasks and other power consumption models. In Section 7, we conclude the paper.

2. Background information

Assume that we are given n independent sequential tasks to be executed on m identical processors. Each task can be executed on any of the m processors. There is no precedence constraint (i.e., dependency) nor communication cost among the tasks. (Note: Extensions of our results to parallel tasks and precedence constrained tasks and other situations are discussed in Section 6.) Let r_i represent the execution requirement (measured in the number of processor cycles or the number of instructions) of task i, where $1 \le i \le n$. The processors can be either computing cores in the same node, or computing cores across different nodes, as long as the cores are homogeneous.

We use the following power consumption model adopted by many researchers [6,9,22,27,36,37]. It is well known that dynamic power consumption p (i.e., the switching component of power) can be accurately modeled by a simple equation, i.e., $p = aCV^2f$, where a is an activity factor, C is the loading capacitance, V is the supply voltage, and f is the clock frequency. In the ideal case, the supply voltage and the clock frequency are related in such a way that $V \propto f^{\phi}$ for some constant $\phi > 0$. The processor execution speed s is usually linearly proportional to the clock frequency, namely, $s \propto f$. For ease of discussion, we will assume that $V = bf^{\phi}$ and $s = cf$, where b and c are some constants. Hence, we know that the dynamic power consumption is $p = aCV^2f = ab^2Cf^{2\phi+1} = (ab^2C/c^{2\phi+1})s^{2\phi+1} = \xi s^{\alpha}$, where $\xi = ab^2C/c^{2\phi+1}$ and $\alpha = 2\phi + 1$. Let p_i represent the dynamic power (measured in watts) consumed to execute task i, which is $p_i = \xi s_i^{\alpha}$, where s_i is the execution speed of task i (measured in GHz or the number of billion instructions per second). Let ψ be the static power consumption (measured in watts). Therefore, the total power consumption is $\xi s_i^{\alpha} + \psi = \xi(s_i^{\alpha} + \psi/\xi)$. Since ξ is a constant which only creates scaling effect, for ease of discussion, we will assume that $\xi = 1$ and simply say that $p = \psi/\xi$ is static power consumption. Hence, the power required to execute task i is $p_i + p = s_i^{\alpha} + p$.

The execution time (measured in seconds) of task i is $t_i = r_i/s_i$. The energy (measured in joule) consumed to execute task i is $e_i = (p_i+p)t_i = (p_i+p)r_i/s_i = r_i(s_i^{\alpha}+p)/s_i = r_i(s_i^{\alpha-1}+p/s_i)$. We observe that

$$\frac{\partial e_i}{\partial s_i} = r_i\left((\alpha-1)s_i^{\alpha-2} - \frac{p}{s_i^2}\right).$$

Hence, when $\partial e_i/\partial s_i = 0$, that is,

$$(\alpha-1)s_i^{\alpha-2} = \frac{p}{s_i^2},$$

which implies that when

$$s_i = s^* = \left(\frac{p}{\alpha-1}\right)^{1/\alpha},$$

e_i has its minimum value of

$$e_i^* = r_i\left((s^*)^{\alpha-1} + \frac{p}{s^*}\right),$$

which is actually

$$e_i^* = r_i p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}}.$$

It is clear that when $0 < s_i \le s^*$, we have $\infty > e_i \ge e_i^*$ and e_i is a decreasing function of s_i; and when $s_i \ge s^*$, we have $e_i \ge e_i^*$ and e_i is an increasing function of s_i. Therefore, for any $s_i \in (0, s^*]$, there is $s_i \in [s^*, \infty)$, such that e_i has the same value. This means that a slower speed in $(0, s^*]$ can be replaced by a faster speed in $[s^*, \infty)$ that leads to reduced execution time without any extra energy consumption. Hence, we assume that $s_i \in [s^*, \infty)$ and $e_i \ge e_i^*$ for all task i.

We can now define the problems to be addressed in this paper.

Given a set of n independent sequential tasks, m identical processors, and energy constraint E, the problem of minimizing schedule length with energy consumption constraint is to find execution speeds $s_1, s_2, \dots, s_n$ of the n tasks and a nonpreemptive schedule of the n tasks on the m processors such that the schedule length is minimized and that the energy consumption does not exceed E.

Given a set of n independent sequential tasks, m identical processors, and time constraint T, the problem of minimizing energy consumption with schedule length constraint is to find execution speeds $s_1, s_2, \dots, s_n$ of the n tasks and a nonpreemptive schedule of the n tasks on the m processors such that the energy consumption is minimized and that the schedule length does not exceed T.

3. Related work

When there is no static power consumption, i.e., $p = 0$, lower bounds were established for the minimum schedule length of a set of tasks with a given energy consumption constraint and the minimum energy consumption of a set of tasks with a given schedule length constraint.

Let $R = r_1 + r_2 + \cdots + r_n$ be the total execution requirement of n independent sequential tasks. For the problem of minimizing schedule length with energy consumption constraint, it was proved in [15] that for a set of independent sequential tasks with total execution requirement R and energy constraint E on m identical processors, we have

$$T^* \ge \left(\frac{m}{E} \left(\frac{R}{m} \right)^{\alpha} \right)^{1/(\alpha-1)}$$

for the optimal schedule length T^*. For the problem of minimizing energy consumption with schedule length constraint, it was proved in [15] that for a set of independent sequential tasks with total execution requirement R and time constraint T on m identical processors, we have

$$E^* \ge m \left(\frac{R}{m} \right)^{\alpha} \frac{1}{T^{\alpha-1}}$$

for the optimal energy consumption E^*.

The above lower bounds have been extended to independent parallel tasks. Let W denote the total amount of work to be performed for n independent parallel tasks (see Section 6 for definition of W). For the problem of minimizing schedule length with energy consumption constraint, it was proved in [16] that for a set of independent parallel tasks with total amount of work W and energy constraint E on m identical processors, we have

$$T^* \ge \left(\frac{m}{E} \left(\frac{W}{m} \right)^{\alpha} \right)^{1/(\alpha-1)}$$

for the optimal schedule length T^*. For the problem of minimizing energy consumption with schedule length constraint, it was proved in [16] that for a set of independent parallel tasks with total amount of work W and time constraint T on m identical processors, we have

$$E^* \ge m \left(\frac{W}{m} \right)^{\alpha} \frac{1}{T^{\alpha-1}}$$

for the optimal energy consumption E^*. It is clear that these lower bounds include the lower bounds for sequential tasks as special cases.

Our lower bounds have also been extended to heterogeneous processors which are specified by $\alpha_1, \alpha_2, \dots, \alpha_m$, i.e., each processor k has its own α_k in the power consumption model, where $1 \le k \le m$. For the problem of minimizing schedule length with energy consumption constraint, it was proved in [20] that for a set of independent sequential tasks with total execution requirement R and energy constraint E on m heterogeneous processors, we have $T^* \ge T$, where T and the partition $R_1, R_2, \dots, R_m$ that results in T can be obtained by solving the $m+1$ equations, i.e., the constraint

$$R_1 + R_2 + \cdots + R_m = R,$$

and m equations

$$T = \left(\alpha_k R_k^{\alpha_k - 1} \left(\sum_{j=1}^{m} \frac{R_j}{\alpha_j} \right) \frac{1}{E} \right)^{1/(\alpha_k-1)},$$

for all $1 \le k \le m$. For the problem of minimizing energy consumption with schedule length constraint, it was proved in [20] that for a set of independent sequential tasks with total execution requirement R and time constraint T on m heterogeneous processors, we have $E^* \ge E$, where E and the partition $R_1, R_2, \dots, R_m$ that results in E are

$$E = T \sum_{k=1}^{m} \left(\frac{\phi}{\alpha_k} \right)^{\alpha_k/(\alpha_k-1)},$$

and

$$R_k = \left(\frac{\phi}{\alpha_k} \right)^{1/(\alpha_k-1)} T,$$

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

$$\sum_{k=1}^{m} \left(\frac{\phi}{\alpha_k} \right)^{1/(\alpha_k-1)} = \frac{R}{T}.$$

It is clear that these lower bounds include the lower bounds for identical processors as special cases.

It is worth mentioning that our lower bounds for independent tasks can be applied to precedence constrained tasks, and on processors with discrete or bounded speed levels.

In this paper, all the above lower bounds on identical processors are extended to include static power consumption $p \ne 0$ for all kinds of tasks (sequential or parallel, and independent or precedence constrained) on all kinds of processors (with discrete or continuous speed levels, and bounded or unbounded speed ranges).

Performance of energy-efficient task scheduling algorithms has been compared with optimal solutions without lower bounds to the optimal solutions. For the problem of minimizing schedule length with energy consumption constraint, it was proved in [26] that for a set of independent sequential tasks, there is a polynomial time approximation scheme, and that for a set of precedence constrained sequential tasks, there is an $O(\log^{1+2/\alpha} m)$-approximation algorithm, where the approximation ratio can be arbitrarily large.

We would like to mention that consideration of static power consumption is different from sleep state management [2,7,13], where the main concern is the energy cost to transition from the sleep state to the operating state. Although static power consumption may also be considered, the preemptive scheduling model in [2,7,13] is substantially different from our nonpreemptive scheduling model.

4. Delay minimization

4.1. Uniprocessor systems

It is clear that on a uniprocessor system with energy constraint E, where

$$E \geq \sum_{i=1}^{n} e_i^* = \left(\sum_{i=1}^{n} r_i\right) p^{1-1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

the problem of minimizing schedule length with energy consumption constraint is simply to find the execution speeds $s_1, s_2, \ldots, s_n$, such that the schedule length

$$T(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} \frac{r_i}{s_i}$$

is minimized and the total energy consumed $e_1+e_2+\cdots+e_n$ does not exceed E, i.e.,

$$F(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} r_i \left(s_i^{\alpha-1} + \frac{p}{s_i}\right) \leq E.$$

Notice that both the schedule length $T(s_1, s_2, \ldots, s_n)$ and the energy consumption $F(s_1, s_2, \ldots, s_n)$ are viewed as functions of the task execution speeds $s_1, s_2, \ldots, s_n$.

We can minimize $T(s_1, s_2, \ldots, s_n)$ subject to the constraint $F(s_1, s_2, \ldots, s_n) = E$ by using the Lagrange multiplier system:

$$\nabla T(s_1, s_2, \ldots, s_n) = \lambda \nabla F(s_1, s_2, \ldots, s_n),$$

where λ is a Lagrange multiplier. Since

$$\frac{\partial T(s_1, s_2, \ldots, s_n)}{\partial s_i} = \lambda \frac{\partial F(s_1, s_2, \ldots, s_n)}{\partial s_i},$$

that is,

$$-\frac{r_i}{s_i^2} = \lambda r_i \left((\alpha-1)s_i^{\alpha-2} - \frac{p}{s_i^2}\right),$$

where $\lambda \leq 0$, we have

$$s_i = s = \left(\frac{1}{\alpha-1}\left(p - \frac{1}{\lambda}\right)\right)^{1/\alpha},$$

for all $1 \leq i \leq n$. Substituting the above s_i into the constraint $F(s_1, s_2, \ldots, s_n) = E$, we get

$$R\left(s^{\alpha-1} + \frac{p}{s}\right) = E,$$

where $R = r_1 + r_2 + \cdots + r_n$ is the total execution requirement of the n tasks.

The last equation can be rewritten as

$$s^\alpha - \left(\frac{E}{R}\right) s + p = 0.$$

Let us consider the function $y = s^\alpha - (E/R)s$. Since

$$\frac{\partial y}{\partial s} = \alpha s^{\alpha-1} - \frac{E}{R},$$

we know that when

$$s = \bar{s} = \left(\frac{E}{\alpha R}\right)^{1/(\alpha-1)},$$

y gets its minimum value of

$$y^* = \left(\frac{E}{\alpha R}\right)^{\alpha/(\alpha-1)} - \frac{E}{R}\left(\frac{E}{\alpha R}\right)^{1/(\alpha-1)},$$

which is

$$y^* = -\frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}}\left(\frac{E}{R}\right)^{\alpha/(\alpha-1)}.$$

It is clear that when $0 < s \leq \bar{s}$, y is a decreasing function of s; and when $s \geq \bar{s}$, y is an increasing function of s. Furthermore, $y \leq 0$ when $0 \leq s \leq (E/R)^{1/(\alpha-1)}$, and $y = 0$ when $s = 0$ and $s = (E/R)^{1/(\alpha-1)}$. To have a solution of s, we must have $-p \geq y^*$, i.e.,

$$p \leq \frac{\alpha-1}{\alpha^{\alpha/(\alpha-1)}}\left(\frac{E}{R}\right)^{\alpha/(\alpha-1)},$$

which implies that

$$E \geq Rp^{1-1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}.$$

Notice that the above condition is consistent with the requirement that

$$E \geq \sum_{i=1}^{n} e_i^*.$$

Furthermore, if the above condition is satisfied, there might be two solutions to the equation $s^\alpha - (E/R)s + p = 0$, one in the interval $(0, \bar{s}]$ and another in the interval $[\bar{s}, (E/R)^{1/(\alpha-1)})$. We will certainly take the faster speed in $[\bar{s}, (E/R)^{1/(\alpha-1)})$. It is observed that increasing E reduces y^* and increases the solution. Also, a reduced p increases the solution.

It is also easy to see that $\bar{s} \geq s^*$, i.e.,

$$\left(\frac{E}{\alpha R}\right)^{1/(\alpha-1)} \geq \left(\frac{p}{\alpha-1}\right)^{1/\alpha}.$$

Once the identical speed s is found, the schedule length is simply $T = R/s$.

The above discussion is summarized in the following theorem which gives the optimal speed setting and the optimal schedule length.

Theorem 1. *On a uniprocessor system, for a given energy consumption constraint E satisfying*

$$E \geq Rp^{1-1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

the schedule length is minimized when all tasks are executed with the same speed s which satisfies

$$s^\alpha - \left(\frac{E}{R}\right) s + p = 0,$$

where

$$s \in \left[\left(\frac{E}{\alpha R}\right)^{1/(\alpha-1)}, \left(\frac{E}{R}\right)^{1/(\alpha-1)}\right).$$

The optimal schedule length is $T = R/s$. Equivalently, T can be obtained by solving the equation $pT^\alpha - ET^{\alpha-1} + R^\alpha = 0$.

When $p = 0$, we have $T = R^{\alpha/(\alpha-1)}/E^{1/(\alpha-1)}$, which was obtained in [15].

Closed form solutions to s can be found for special cases of α. When $\alpha = 2$, we have

$$s^2 - \left(\frac{E}{R}\right) s + p = 0,$$

and

$$s = \frac{1}{2}\left(E/R + \sqrt{(E/R)^2 - 4p}\right).$$

When $\alpha = 3$, we have

$$s^3 - \left(\frac{E}{R}\right) s + p = 0.$$

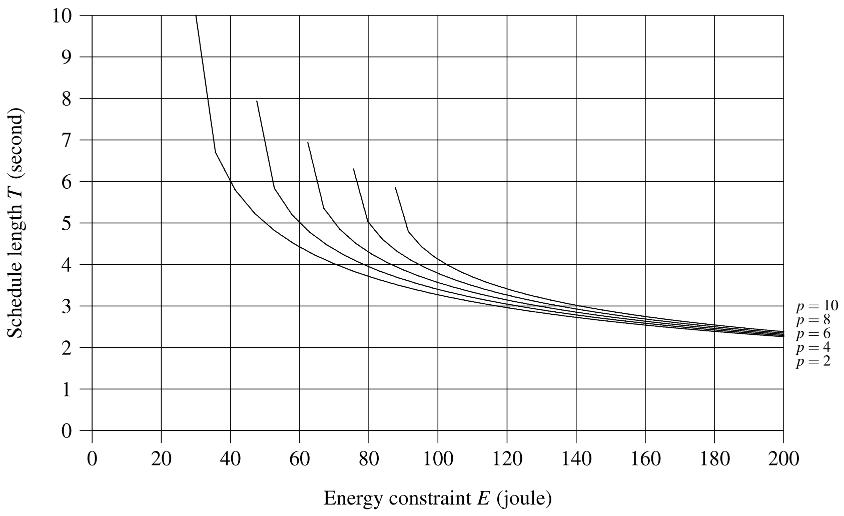

Fig. 1. Schedule length T vs. energy constraint E (uniprocessor).

The discriminant of the above cubic equation is

$$\Delta = -4\left(\frac{E}{3R}\right)^3 + p^2.$$

Since

$$E \ge 3\left(\frac{p}{2}\right)^{2/3} R,$$

we have $\Delta \le 0$, and the equation has three real roots ([42], p. 82). The one which we are seeking is

$$s = 2\sqrt[3]{r}\cos\frac{\theta}{3} = 2\sqrt{\frac{E}{3R}}\cos\frac{\theta}{3},$$

where

$$r = \sqrt{\left(\frac{E}{3R}\right)^3},$$

and

$$\theta = \cos^{-1}\left(-\frac{p}{2r}\right),$$

and $2r \ge p$. (Note: The other two solutions are obtained by replacing θ by $\theta + 2\pi$, which gives $s < 0$, and θ by $\theta + 4\pi$, which gives $s > 0$ but too small to be in the desired interval.) For fixed α and p, s approaches $(E/R)^{1/(\alpha-1)}$ as E/R gets large, which was obtained in [15] for $p = 0$.

In Fig. 1, we show T obtained in Theorem 1 as a function of E, where $\alpha = 3$, $R = 10$, and $p = 2, 4, 6, 8, 10$. Notice that for a given p, there is a minimum energy requirement in Theorem 1, for which, we get the maximum schedule length R/s^*. It is clear that as E increases, T decreases accordingly.

4.2. Multiprocessor systems

Let us consider a multiprocessor system with m processors. A schedule of a set of n tasks is essentially a partition of the set into m groups, such that all the tasks in group k are executed on processor k, where $1 \le k \le m$. Let R_k denote group k as well as the total execution requirement of the tasks in group k. For a given schedule $(R_1, R_2, \ldots, R_m)$ of the n tasks on m processors, we are seeking task execution speeds that minimize the schedule length.

Let E_k be the energy consumed by all the tasks in group k. We observe that by fixing E_k and adjusting the execution speeds of the tasks in group k to the same speed s_k which satisfies

$$s_k^\alpha - \left(\frac{E_k}{R_k}\right)s_k + p = 0,$$

according to Theorem 1, the total execution time of the tasks in group k can be minimized to $T_k = R_k/s_k$. Therefore, the problem of finding execution speeds $s_1, s_2, \ldots, s_n$ that minimize the schedule length is equivalent to finding $E_1, E_2, \ldots, E_m$ that minimize the schedule length. It is clear that if E is large enough, the schedule length is minimized when all the m processors complete their execution of the m groups of tasks at the same time T, that is, $T_1 = T_2 = \cdots = T_m = T$. Since

$$E_k = \sum_{j \in R_k} e_j = \left(\sum_{j \in R_k} r_j\right)\left(s_k^{\alpha-1} + \frac{p}{s_k}\right) = R_k\left(s_k^{\alpha-1} + \frac{p}{s_k}\right)$$
$$= \frac{R_k^\alpha}{T^{\alpha-1}} + pT,$$

where $s_k = R_k/T$, we have

$$E = \sum_{k=1}^{m} E_k = \frac{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha}{T^{\alpha-1}} + mpT,$$

which implies that

$$mpT^\alpha - ET^{\alpha-1} + \sum_{k=1}^{m} R_k^\alpha = 0.$$

Since

$$\frac{E_k - pT}{E - mpT} = \frac{R_k^\alpha}{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha},$$

we get

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

for all $1 \le k \le m$.

To solve the equation of T, let us consider the function

$$y = mpT^\alpha - ET^{\alpha-1} = T^{\alpha-1}(mpT - E).$$

Since

$$\frac{\partial y}{\partial T} = \alpha mpT^{\alpha-1} - (\alpha - 1)ET^{\alpha-2} = T^{\alpha-2}(\alpha mpT - (\alpha - 1)E),$$

we know that when

$$T = \bar{T} = \left(1 - \frac{1}{\alpha}\right)\frac{E}{mp},$$

y gets its minimum value of

$$y^* = \left(\frac{\alpha-1}{\alpha}\cdot\frac{E}{mp}\right)^{\alpha-1}\left(mp\cdot\frac{\alpha-1}{\alpha}\cdot\frac{E}{mp}-E\right) = -\frac{(\alpha-1)^{\alpha-1}}{\alpha^\alpha}\cdot\frac{E^\alpha}{(mp)^{\alpha-1}}.$$

It is clear that when $0 < T \le \bar{T}$, y is a decreasing function of T; and when $T \ge \bar{T}$, y is an increasing function of T. Furthermore, $y \le 0$ when $0 \le T \le E/(mp)$, and $y = 0$ when $T = 0$ and $T = E/(mp)$. To have a solution of T, we must have

$$-\sum_{k=1}^{m} R_k^\alpha \ge y^*,$$

that is,

$$\sum_{k=1}^{m} R_k^\alpha \le \frac{(\alpha-1)^{\alpha-1}}{\alpha^\alpha}\cdot\frac{E^\alpha}{(mp)^{\alpha-1}},$$

which gives rise to

$$E \ge m^{1-1/\alpha}\left(\sum_{k=1}^{m} R_k^\alpha\right)^{1/\alpha} p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}}.$$

Furthermore, if the above condition is satisfied, there might be two solutions to the equation

$$mpT^\alpha - ET^{\alpha-1} + \sum_{k=1}^{m} R_k^\alpha = 0,$$

one in the interval $(0, \bar{T}]$ and another in the interval $[\bar{T}, E/(mp))$. We will certainly take the shorter time in $(0, \bar{T}]$. It is observed that increasing E reduces y^* and decreases the solution. Also, a reduced $R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha$ decreases the solution.

Notice that the above condition for E is stronger than the one derived from Theorem 1, i.e.,

$$E = \sum_{k=1}^{m} E_k \ge \left(\sum_{k=1}^{m} R_k\right) p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

where we notice that

$$m^{1-1/\alpha}\left(\sum_{k=1}^{m} R_k^\alpha\right)^{1/\alpha} \ge \sum_{k=1}^{m} R_k,$$

that is,

$$\frac{1}{m}\sum_{k=1}^{m} R_k^\alpha \ge \left(\frac{1}{m}\sum_{k=1}^{m} R_k\right)^\alpha,$$

which is clearly true due to the convexity of the function x^α.

Thus, we have proved the following theorem.

Theorem 2. *For a given schedule $(R_1, R_2, \ldots, R_m)$ of n tasks on a multiprocessor system with m processors, if*

$$E \ge m^{1-1/\alpha}\left(\sum_{k=1}^{m} R_k^\alpha\right)^{1/\alpha} p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

we can achieve the schedule length T which satisfies

$$mpT^\alpha - ET^{\alpha-1} + \sum_{k=1}^{m} R_k^\alpha = 0,$$

where

$$T \in \left(0, \left(1-\frac{1}{\alpha}\right)\frac{E}{mp}\right],$$

by allocating the following energy to group k,

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

for all $1 \le k \le m$. The execution speed of tasks in group k is $s_k = R_k/T$, for all $1 \le k \le m$.

When $m = 1$, Theorem 2 is equivalent to Theorem 1, since the equation $pT^\alpha - ET^{\alpha-1} + R^\alpha = 0$ can be obtained from $s^\alpha - (E/R)s + p = 0$ by letting $s = R/T$.

When $p = 0$, we have

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

which was obtained in [15].

The speed setting obtained by Theorem 2 is based on the conditions that E is sufficiently large and that $T_1 = T_2 = \cdots = T_m$. When E is not sufficient to guarantee $T_1 = T_2 = \cdots = T_m$, we need to take a new approach. Furthermore, if E is insufficient (i.e., $E < E^{(m)}$, where $E^{(m)}$ is defined below), the condition $T_1 = T_2 = \cdots = T_m$ does not necessarily yield the minimum schedule length. For instance, if R_k is too small, we might have $s_k = R_k/T < s^*$ and $T > R_k/s^*$. In this case, we would rather set $s_k = s^*$, which not only reduces T_k, but also reduces E_k. The saved energy can be allocated to other groups to reduce the schedule length. The principle is, we should keep $s_k \ge s^*$ and $T_k \le R_k/s^*$, for all $1 \le k \le m$.

Without loss of generality, we assume that $R_1 \ge R_2 \ge \cdots \ge R_m$. Let us define $E^{(1)}$ to be the amount of energy just enough to run all the tasks at the minimum speed s^*, i.e.,

$$E^{(1)} = \sum_{i=1}^{n} e_i^* = Rp^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}}.$$

If $E = E^{(1)}$, all tasks have the same minimum execution speed s^*, which gives $T_k = R_k/s^*$, and $T_1 \ge T_2 \ge \cdots \ge T_m$. If $E > E^{(1)}$, we first allocate the extra energy $E - E^{(1)}$ to group 1, such that T_1 can be reduced. If $T_1 = T_2$, and there is still extra energy, we then allocate the extra energy to groups 1 and 2, such that T_1 and T_2 can be reduced. If $T_1 = T_2 = T_3$, and there is still extra energy, we continue to allocate the extra energy to groups 1, 2, and 3, such that T_1, T_2, and T_3 can be reduced. Let us define $E^{(k)}$ to be the amount of energy just enough to have $T_1 = T_2 = \cdots = T_k$, and tasks in groups $k, k+1, \ldots, m$ are executed with the minimum speed s^*, where $1 \le k \le m$. It is clear that $T_{k'} = T_k$, i.e., $R_{k'}/s_{k'} = R_k/s^*$, and $s_{k'} = (R_{k'}/R_k)s^*$, for all $1 \le k' < k$. Hence, we get

$$E^{(k)} = \sum_{k'=1}^{k-1} E_{k'} + \sum_{k'=k}^{m} E_{k'} = \sum_{k'=1}^{k-1} R_{k'}\left(s_{k'}^{\alpha-1} + \frac{p}{s_{k'}}\right) + \sum_{k'=k}^{m} R_{k'}\left((s^*)^{\alpha-1} + \frac{p}{s^*}\right),$$

which is actually

$$E^{(k)} = \sum_{k'=1}^{k-1}\left(\frac{R_{k'}^\alpha}{R_k^{\alpha-1}}(s^*)^{\alpha-1} + p\frac{R_k}{s^*}\right) + \left(\sum_{k'=k}^{m} R_{k'}\right)p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

or,

$$E^{(k)} = \frac{1}{R_k^{\alpha-1}}\left(\sum_{k'=1}^{k-1} R_{k'}^\alpha\right)(s^*)^{\alpha-1} + (k-1)p\frac{R_k}{s^*} + \left(\sum_{k'=k}^{m} R_{k'}\right)p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

where $1 \le k \le m$. (Notice that $E^{(1)} \le E^{(2)} \le \cdots \le E^{(m)}$. Also, if $R_k = R_{k+1}$, then $E^{(k)} = E^{(k+1)}$.)

Assume that k is the largest integer such that $E \ge E^{(k)}$, where $1 \le k \le m$, i.e., E is enough for $T_1 = T_2 = \cdots = T_k$, but not for $T_1 = T_2 = \cdots = T_k = T_{k+1}$. Then, we allocate $E - E^{(k)}$ to groups 1, 2, …, k, such that $T_1 = T_2 = \cdots = T_k = T$. Since $s_k = R_k/T$, we get

$$E_k = R_k\left(s_k^{\alpha-1} + \frac{p}{s_k}\right) = \frac{R_k^\alpha}{T^{\alpha-1}} + pT,$$

and

$$\sum_{k'=1}^{k} E_k = \frac{R_1^\alpha + R_2^\alpha + \cdots + R_k^\alpha}{T^{\alpha-1}} + kpT.$$

Since the available energy for groups 1, 2, …, k is

$$E' = E - \left(\sum_{k'=k+1}^{m} R_{k'}\right) p^{1-1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

we have the following equation of T,

$$\frac{R_1^\alpha + R_2^\alpha + \cdots + R_k^\alpha}{T^{\alpha-1}} + kpT = E',$$

which yields

$$kpT^\alpha - E'T^{\alpha-1} + \sum_{k'=1}^{k} R_{k'}^\alpha = 0.$$

The above discussion essentially proves the following theorem.

Theorem 3. *For a given schedule $(R_1, R_2, \ldots, R_m)$ of n tasks on a multiprocessor system with m processors, if k is the largest integer such that $E \ge E^{(k)}$, where $1 \le k \le m$, we can achieve the minimum schedule length T which satisfies*

$$kpT^\alpha - \left(E - \left(\sum_{k'=k+1}^{m} R_{k'}\right) p^{1-1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}\right) T^{\alpha-1} + \sum_{k'=1}^{k} R_{k'}^\alpha = 0,$$

where

$$T \in \left(0, \left(1 - \frac{1}{\alpha}\right)\frac{E'}{kp}\right]$$

The execution speed of tasks in group k' is $s_{k'} = R_{k'}/T$ for all $1 \le k' \le k$, and $s_{k'} = s^*$ for all $k+1 \le k' \le m$.

When $E \ge E^{(m)}$, Theorem 3 reduces to Theorem 2. As pointed out before, when $E < E^{(m)}$, Theorem 2 does not give the minimum schedule length, since the condition $T_1 = T_2 = \cdots = T_m$ is not necessary.

Closed form solutions to T can be found for special cases of α. When $\alpha = 2$, we have

$$kpT^2 - E'T + \sum_{k'=1}^{k} R_{k'}^2 = 0,$$

and

$$T = \frac{1}{2kp}\left(E' - \sqrt{(E')^2 - 4kp\sum_{k'=1}^{k} R_{k'}^2}\right).$$

When $\alpha = 3$, we have

$$kpT^3 - E'T^2 + M = 0,$$

where

$$M = \sum_{k'=1}^{k} R_{k'}^3.$$

If we let

$$y = T - \frac{E'}{3kp},$$

we get $y^3 + 3uy + v = 0$, where

$$u = -\left(\frac{E'}{3kp}\right)^2,$$

and

$$v = \frac{27(kp)^2M - 2(E')^3}{(3kp)^3}.$$

The discriminant of the equation of y is

$$\Delta = 4u^3 + v^2 = \frac{27(kp)^2M(27(kp)^2M - 4(E')^3)}{(3kp)^6}.$$

Since $E \ge E^{(k)}$, we have

$$E' \ge \frac{1}{R_k^{\alpha-1}}\left(\sum_{k'=1}^{k-1} R_{k'}^\alpha\right)(s^*)^{\alpha-1} + (k-1)p\frac{R_k}{s^*} + R_k p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}}.$$

Since

$$(s^*)^{\alpha-1} + \frac{p}{s^*} = p^{1-1/\alpha}\frac{\alpha}{(\alpha-1)^{1-1/\alpha}},$$

the last inequality is actually

$$E' \ge \frac{1}{R_k^{\alpha-1}}\left(\sum_{k'=1}^{k} R_{k'}^\alpha\right)(s^*)^{\alpha-1} + kp\frac{R_k}{s^*}.$$

When $\alpha = 3$, the last inequality is

$$E' \ge M\left(\frac{s^*}{R_k}\right)^2 + \frac{1}{2}kp\frac{R_k}{s^*} + \frac{1}{2}kp\frac{R_k}{s^*},$$

which implies that

$$E' \ge 3\sqrt[3]{\left(M\left(\frac{s^*}{R_k}\right)^2\right)\left(\frac{1}{2}kp\frac{R_k}{s^*}\right)\left(\frac{1}{2}kp\frac{R_k}{s^*}\right)},$$

and $4(E')^3 \ge 27(kp)^2M$, and $\Delta \le 0$. Therefore, y has three real roots. The one we are seeking is

$$y = 2\sqrt[3]{r}\cos\left(\frac{\theta + 4\pi}{3}\right) = \frac{2E'}{3kp}\cos\left(\frac{\theta + 4\pi}{3}\right),$$

where

$$r = \sqrt{-u^3} = \left(\frac{E'}{3kp}\right)^3,$$

and

$$\theta = \cos^{-1}\left(-\frac{v}{2r}\right) = \cos^{-1}\left(\frac{2(E')^3 - 27(kp)^2M}{2(E')^3}\right).$$

It is easy to see that $-2(E')^3 \le 2(E')^3 - 27(kp)^2M < 2(E')^3$, which implies that $0 < \theta \le \pi$, and $4\pi/3 < (\theta + 4\pi)/3 \le 5\pi/3$, and $-\frac{1}{2} < \cos((\theta + 4\pi)/3) \le \frac{1}{2}$, which gives $-E'/3kp < y \le E'/3kp$.

Finally, we have $T = y + E'/3kp$, and $0 < T \le 2E'/3kp$.

(Note: The other two solutions are obtained by replacing $\theta + 4\pi$ by θ, which gives $T > (1 - 1/\alpha)E'/kp$, and $\theta + 4\pi$ by $\theta + 2\pi$, which gives $T < 0$.)

In Fig. 2, we show T obtained in Theorem 3 as a function of E, where $\alpha = 3$, $m = 7$, $(R_1, R_2, R_3, R_4, R_5, R_6, R_7) = (16, 14, 12, 10, 8, 6, 4)$, and $p = 2, 4, 6, 8, 10$. Notice that for a given p, there is a

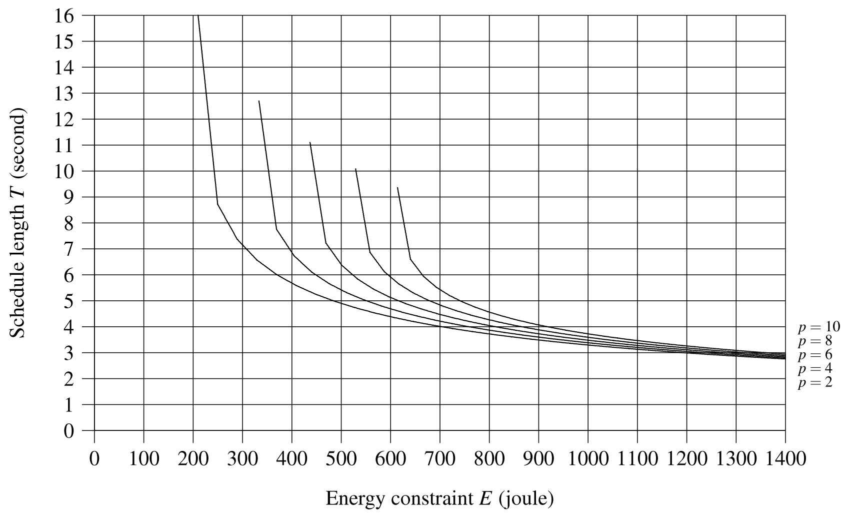

Fig. 2. Schedule length T vs. energy constraint E (multiprocessor).

minimum energy requirement in Theorem 3, for which, we get the maximum schedule length R_1/s^*. It is clear that as E increases, T decreases accordingly.

4.3. Lower bound

As mentioned earlier, a reduced $R_1^\alpha+R_2^\alpha+\cdots+R_m^\alpha$ decreases the solution to T. For a set of tasks with total execution requirement R, the best possible schedule $(R_1, R_2, \ldots, R_m)$ which minimizes $R_1^\alpha+R_2^\alpha+\cdots+R_m^\alpha$ and T is the one which has $R_1 = R_2 = \cdots = R_m = R/m$. By applying Theorem 3 on such a perfect schedule, we get the following theorem which gives a lower bound for the optimal schedule length. Since a perfect schedule is achievable, the lower bound is tight.

Theorem 4. *For a set of tasks with total execution requirement R on a multiprocessor system with m processors, let T be the solution to the following equation,*

$$mpT^\alpha - ET^{\alpha-1} + \frac{R^\alpha}{m^{\alpha-1}} = 0,$$

where

$$T \in \left(0, \left(1-\frac{1}{\alpha}\right)\frac{E}{mp}\right]$$

The optimal schedule length is $T^* \ge T$. The lower bound is tight.

The most significant application of a lower bound is to evaluate the performance of a heuristic algorithm.

When $p = 0$, we have

$$T = \frac{R^{\alpha/(\alpha-1)}}{mE^{1/(\alpha-1)}},$$

which was obtained in [15].

Closed form solutions to T can be found for special cases of α. When $\alpha = 2$, we have

$$mpT^2 - ET + \frac{R^2}{m} = 0,$$

and

$$T = \frac{1}{2mp}\left(E - \sqrt{E^2 - 4pR^2}\right).$$

When $\alpha = 3$, we have

$$mpT^3 - ET^2 + \frac{R^3}{m^2} = 0,$$

and

$$T = \frac{E}{3mp} + \frac{2E}{3mp} \times \cos\left(\frac{1}{3}\left(\cos^{-1}\left(\frac{2E^3 - 27(mp)^2R^3/m^2}{2E^3}\right) + 4\pi\right)\right),$$

which can be derived from our earlier discussion.

In Fig. 3, we show T obtained in Theorem 4 as a function of E, where $\alpha = 3$, $m = 7$, $R = 70$, and $p = 2, 4, 6, 8, 10$. Notice that for a given p, there is a minimum energy requirement in Theorem 4, for which, we get the maximum schedule length $R/m/s^*$. It is clear that as E increases, T decreases accordingly.

5. Energy minimization

5.1. Uniprocessor systems

It is clear that on a uniprocessor system with time constraint T, where $T \le R/s^*$, the problem of minimizing energy consumption with schedule length constraint is simply to find the execution speeds $s_1, s_2, \ldots, s_n$, such that the total energy consumption

$$E(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} e_i = \sum_{i=1}^{n} r_i\left(s_i^{\alpha-1} + \frac{p}{s_i}\right)$$

is minimized and the schedule length $t_1 + t_2 + \cdots + t_n$ does not exceed T, i.e.,

$$F(s_1, s_2, \ldots, s_n) = \sum_{i=1}^{n} t_i = \sum_{i=1}^{n} \frac{r_i}{s_i} \le T.$$

The energy consumption $E(s_1, s_2, \ldots, s_n)$ and the schedule length $F(s_1, s_2, \ldots, s_n)$ are viewed as functions of $s_1, s_2, \ldots, s_n$.

We can minimize $E(s_1, s_2, \ldots, s_n)$ subject to the constraint $F(s_1, s_2, \ldots, s_n) = T$ by using the Lagrange multiplier system:

$$\nabla E(s_1, s_2, \ldots, s_n) = \lambda \nabla F(s_1, s_2, \ldots, s_n),$$

where λ is a Lagrange multiplier. Since

$$\frac{\partial E(s_1, s_2, \ldots, s_n)}{\partial s_i} = \lambda \frac{\partial F(s_1, s_2, \ldots, s_n)}{\partial s_i},$$

that is,

$$r_i\left((\alpha-1)s_i^{\alpha-2} - \frac{p}{s_i^2}\right) = -\lambda\frac{r_i}{s_i^2},$$

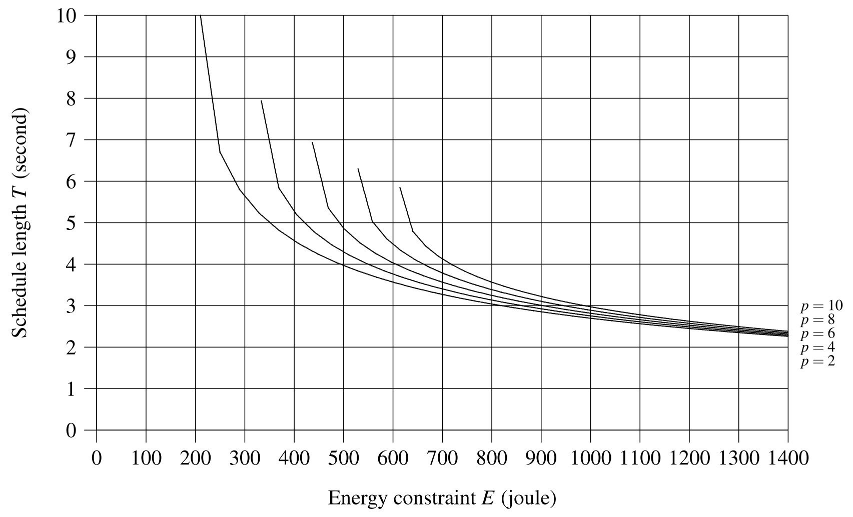

Fig. 3. Schedule length T vs. energy constraint E (lower bound).

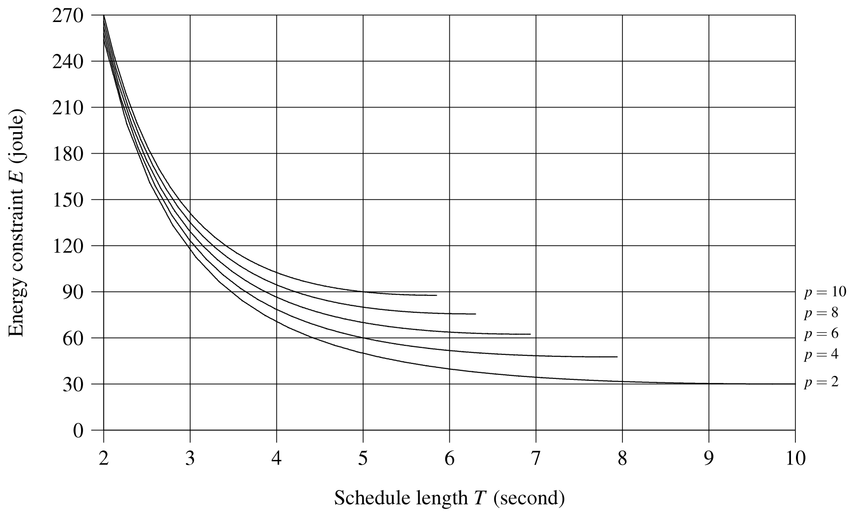

Fig. 4. Energy constraint E vs. schedule length T (uniprocessor).

where $\lambda \le 0$, we have

$$s_i = s = \left(\frac{p-\lambda}{\alpha-1}\right)^{1/\alpha} \ge s^*,$$

for all $1 \le i \le n$. Substituting the above s_i into the constraint $F(s_1, s_2, \ldots, s_n) = T$, we get $s = R/T$.

The above discussion gives rise to the following theorem which gives the optimal speed setting and the minimum energy consumption.

Theorem 5. *On a uniprocessor system, for a given schedule length constraint T satisfying $T \le R/s^*$, the total energy consumption is minimized when all tasks are executed with the same speed $s = R/T$. The minimum energy consumption is $E = R(s^{\alpha-1}+p/s) = R^\alpha/T^{\alpha-1}+pT$.*

Notice that E in Theorem 5 can be obtained from the equation of T in Theorem 1.

When $p = 0$, we have $E = R^\alpha/T^{\alpha-1}$, which was obtained in [15].

In Fig. 4, we show E obtained in Theorem 5 as a function of T, where $\alpha = 3$, $R = 10$, and $p = 2, 4, 6, 8, 10$. Notice that for a given p, there is a maximum schedule length R/s^* in Theorem 5, for which, we get the minimum energy consumption. It is clear that as T increases, E decreases accordingly.

5.2. Multiprocessor systems

By Theorem 5, for a given time constraint T, where $T \le R_k/s^*$, for all $1 \le k \le m$, the energy consumed by tasks in group k is minimized as $E_k = R_k^\alpha/T^{\alpha-1}+pT$ by executing all the tasks in group k with the same speed R_k/T without missing the time deadline T. The minimum energy consumption is simply

$$E_1 + E_2 + \cdots + E_m = \frac{R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha}{T^{\alpha-1}} + mpT.$$

The following result gives the optimal speed setting that minimizes energy consumption for a given schedule of n tasks on m processors.

Theorem 6. *For a given schedule $(R_1, R_2, \ldots, R_m)$ of n tasks on a multiprocessor system with m processors, and a schedule length constraint T satisfying $T \le R_k/s^*$, for all $1 \le k \le m$, the total energy consumption is minimized when all the tasks in group k are executed with the same speed R_k/T, where $1 \le k \le m$. The energy consumption is*

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

for the above speed setting.

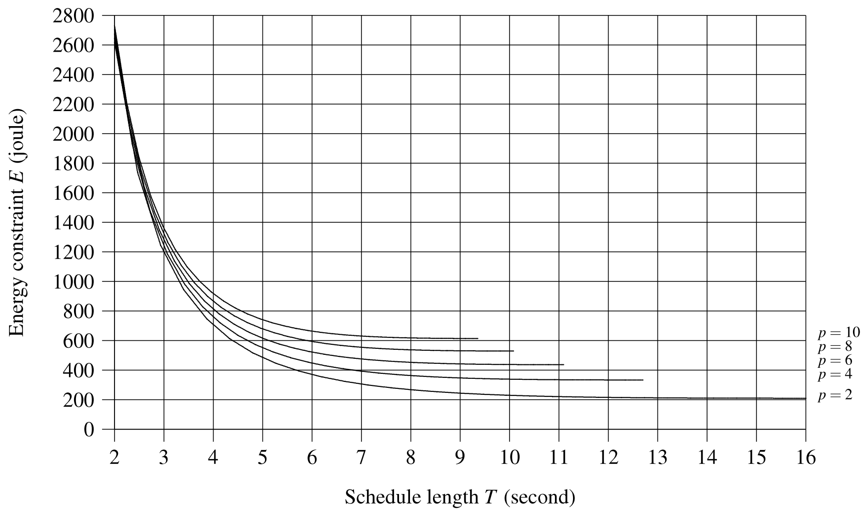

Fig. 5. Energy constraint E vs. schedule length T (multiprocessor).

Notice that E in Theorem 6 can be obtained from the equation of T in Theorem 2. When $m = 1$, Theorem 6 is equivalent to Theorem 5.

When $p = 0$, we have $E = (R_1^\alpha + R_2^\alpha + \cdots + R_m^\alpha)/T^{\alpha-1}$, which was obtained in [15].

The speed setting obtained by Theorem 6 is based on the condition that T is sufficiently short, i.e., $T \le \min\{R_1/s^*, R_2/s^*, \ldots, R_m/s^*\}$. For longer T, more careful treatment is required. Again, without loss of generality, we assume that $R_1 \ge R_2 \ge \cdots \ge R_m$. Assume that k is the largest integer such that $T \le R_k/s^*$, where $1 \le k \le m$. Then, we can apply Theorem 5 for groups 1, 2, ..., k, and set the minimum speed s^* for groups $k+1, k+2, \ldots, m$. The minimum energy consumption is

$$\frac{R_1^\alpha + R_2^\alpha + \cdots + R_k^\alpha}{T^{\alpha-1}} + kpT + \sum_{k'=k+1}^{m} R_{k'}\left((s^*)^{\alpha-1} + \frac{p}{s^*}\right).$$

The above discussion is summarized in the following theorem.

Theorem 7. *For a given schedule $(R_1, R_2, \ldots, R_m)$ of n tasks on a multiprocessor system with m processors, if k is the largest integer such that $T \le R_k/s^*$, where $1 \le k \le m$, the total energy consumption is minimized when all the tasks in group k' are executed with the same speed $R_{k'}/T$ for all $1 \le k' \le k$, and s^* for all $k+1 \le k' \le m$. The energy consumption is*

$$E = \frac{R_1^\alpha + R_2^\alpha + \cdots + R_k^\alpha}{T^{\alpha-1}} + kpT + \left(\sum_{k'=k+1}^{m} R_{k'}\right) p^{1-1/\alpha} \frac{\alpha}{(\alpha-1)^{1-1/\alpha}}$$

for the above speed setting.

Notice that E in Theorem 7 can be obtained from the equation of T in Theorem 3. When $k = m$, Theorem 7 reduces to Theorem 6.

In Fig. 5, we show E obtained in Theorem 7 as a function of T, where $\alpha = 3$, $m = 7$, $(R_1, R_2, R_3, R_4, R_5, R_6, R_7) = (16, 14, 12, 10, 8, 6, 4)$, and $p = 2, 4, 6, 8, 10$. Notice that for a given p, there is a maximum schedule length R_1/s^* in Theorem 7, for which, we get the minimum energy consumption. It is clear that as T increases, E decreases accordingly.

5.3. Lower bound

By applying Theorem 7 on a perfect schedule with $R_1 = R_2 = \cdots = R_m = R/m$, we get the following theorem which gives a lower bound for the optimal energy consumption.

Theorem 8. *For a set of tasks with total execution requirement R on a multiprocessor system with m processors, let*

$$E = \frac{R^\alpha}{(mT)^{\alpha-1}} + mpT.$$

The optimal energy consumption is $E^* \ge E$. The lower bound is tight.

Notice that E in Theorem 8 can be obtained from the equation of T in Theorem 4.

When $p = 0$, we have $E = R^\alpha/(mT)^{\alpha-1}$, which was obtained in [15].

In Fig. 6, we show E obtained in Theorem 8 as a function of T, where $\alpha = 3$, $m = 7$, $R = 70$, and $p = 2, 4, 6, 8, 10$. Notice that for a given p, there is a maximum schedule length $R/m/s^*$ in Theorem 8, for which, we get the minimum energy consumption. It is clear that as T increases, E decreases accordingly.

6. Simulation data

In this section, we demonstrate experimental data for some heuristic algorithms.

As mentioned earlier, both the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint on m identical processors can be solved by finding a partition R_1, $R_2, \ldots, R_m$ of the n tasks into m groups. Such a partition is essentially a schedule of the n tasks on m processors. Once a partition (i.e., a schedule) is available, Theorems 3 and 7 can be used to decide the actual task execution speeds which minimize either schedule length or energy consumption.

We consider the classic list scheduling (LS) algorithm [11] and its variations to solve the scheduling problem. Assume that 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 LS algorithm which works as follows to schedule a list of tasks 1, 2, ..., n.

- *List Scheduling* (LS): Initially, task k is scheduled on processor (or group) k, where $1 \le k \le m$, and tasks 1, 2, ..., 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.

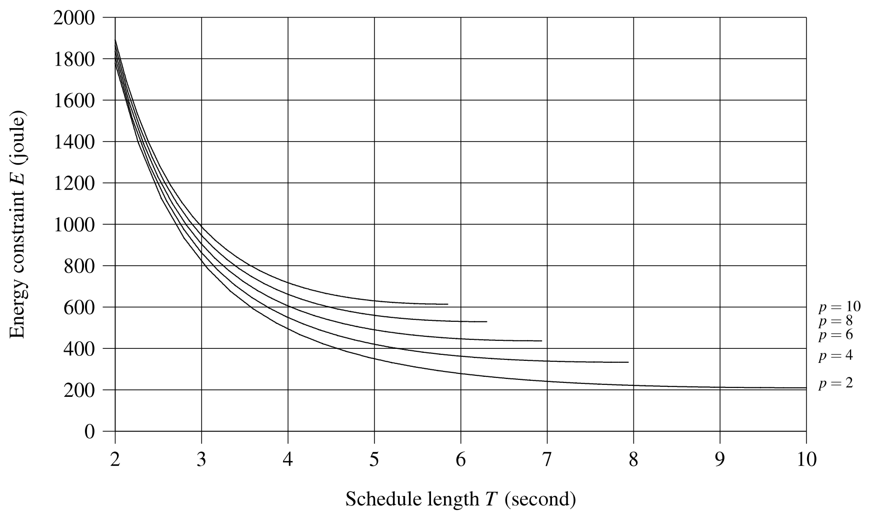

Fig. 6. Energy constraint E vs. schedule length T (lower bound).

Table 1
Simulation Data for Expected NSL (Uniform Distribution).

n	SRF	LS	LRF
30	1.0594634	1.0410098	1.0026986
40	1.0336427	1.0233217	1.0008751
50	1.0216661	1.0147830	1.0003638
60	1.0150351	1.0102320	1.0001795
70	1.0110245	1.0075399	1.0000966
80	1.0084437	1.0057102	1.0000586
90	1.0066710	1.0045194	1.0000369

(99% Confidence Interval ±0.064%).

Table 2
Simulation Data for Expected NSL (Exponential Distribution).

n	SRF	LS	LRF
30	1.2750190	1.1704856	1.0539067
40	1.1764939	1.0982698	1.0129433
50	1.1232332	1.0627054	1.0036916
60	1.0925662	1.0429723	1.0011996
70	1.0720423	1.0312238	1.0003108
80	1.0579561	1.0238927	1.0001484
90	1.0476960	1.0193924	1.0000548

(99% Confidence Interval ±0.454%).

- *Largest Requirement First* (LRF): This algorithm is the same as the LS algorithm, except that the tasks are arranged such that $r_1 \geq r_2 \geq \cdots \geq r_n$.
- *Smallest Requirement First* (SRF): This algorithm is the same as the LS algorithm, except that the tasks are arranged such that $r_1 \leq r_2 \leq \cdots \leq r_n$.

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

We define the *performance ratio* as $\beta = T/T^*$ for a heuristic algorithm that solves the problem of minimizing schedule length with energy consumption constraint, where T is the minimized schedule length for the partition $(R_1, R_2, \ldots, R_m)$ produced by the heuristic algorithm according to Theorem 3, and T^* is the optimal schedule length. We define the *performance ratio* as $\beta = E/E^*$ for a heuristic algorithm that solves the problem of minimizing energy consumption with schedule length constraint, where E is the minimized energy consumption for the partition $(R_1, R_2, \ldots, R_m)$ produced by the heuristic algorithm according to Theorem 7, and E^* is the optimal energy consumption.

We define the *normalized schedule length* (NSL) as T divided by the lower bound obtained by Theorem 4. NSL is an upper bound for the performance ratio $\beta = T/T^*$ for the problem of minimizing schedule length with energy consumption constraint. When the r_i's are random variables, T, T^*, β, and NSL all become random variables. It is clear that $\bar{\beta} \leq \overline{\text{NSL}}$, i.e., the expected performance ratio is no greater than the expected normalized schedule length. (We use $\bar{x}$ to represent the expectation of a random variable x.) We define the *normalized energy consumption* (NEC) as E divided by the lower bound obtained by Theorem 8. NEC is an upper bound for the performance ratio $\beta = E/E^*$ for the problem of minimizing energy consumption with schedule length constraint. It is clear that $\bar{\beta} \leq \overline{\text{NEC}}$.

Notice that for a given heuristic algorithm, the expected normalized schedule length $\overline{\text{NSL}}$ (the expected normalized energy consumption $\overline{\text{NEC}}$, respectively) are determined by m, n, α, p, the probability distribution of the r_i's, and E (T, respectively). In our simulations, the number of processors is set as $m = 10$. The number of tasks is in the range $n = 30, 40, \ldots, 90$. The parameter α is set as 3. The static power consumption p is set as 5. The r_i's are treated as independent and identically distributed (i.i.d.) continuous random variables. Two probability distributions are considered, i.e., a uniform distribution in the range [0, 1) and an exponential distribution with mean 1. The energy constraint E is set as $2(e_1^* + e_2^* + \cdots + e_n^*)$, i.e., twice the minimum required energy. The time constraint T is set as $(R_1/s^*)/2$, i.e., half of the longest execution time of all processors with the minimum execution speed.

Tables 1–4 show our simulation data. For each combination of n, probability distribution, and algorithm $A \in \{$ SRF, LS, LRF $\}$, we generate 5000 sets of n random tasks, produce their schedules by using algorithm A, calculate their NSL (or NEC), and report the average of NSL (or NEC), which is the experimental value of $\overline{\text{NSL}}$ (or $\overline{\text{NEC}}$). The 99% confidence interval of all the data in the same table is also given.

We observe the following facts. (1) The performance of the three list scheduling algorithms are ranked as SRF, LS, LRF, from the worst to the best. This is not surprising since LRF schedules tasks with long execution times earlier and causes more balanced task distribution among the processors. On the other hand, SRF schedules tasks with short execution times earlier and causes more imbalanced task distribution among the processors. (2) In all cases, $\overline{\text{NSL}}$ and $\overline{\text{NEC}}$ (and $\bar{\beta}$ as well) quickly approach one as n increases, which means that the performance of list scheduling algorithms is close to the optimal. (3) Finally, a probability distribution of task requirements with greater coefficient of variation (e.g., exponential distribution vs. uniform distribution) results in degraded performance of the list scheduling algorithms.

Table 3
Simulation Data for Expected NEC (Uniform Distribution).

n	SRF	LS	LRF
30	1.0704568	1.0490473	1.0038345
40	1.0421567	1.0294917	1.0012634
50	1.0280285	1.0192497	1.0005388
60	1.0197812	1.0135356	1.0002687
70	1.0148981	1.0100991	1.0001469
80	1.0114983	1.0077859	1.0000863
90	1.0091699	1.0062641	1.0000537

(99% Confidence Interval ±0.069%).

Table 4
Simulation Data for Expected NEC (Exponential Distribution).

n	SRF	LS	LRF
30	1.1592101	1.1172734	1.0394368
40	1.1345771	1.0827573	1.0121896
50	1.1102559	1.0612325	1.0036915
60	1.0902935	1.0453380	1.0010875
70	1.0750003	1.0347227	1.0004506
80	1.0628774	1.0276285	1.0001421
90	1.0538246	1.0219334	1.0000577

(99% Confidence Interval ±0.156%).

We would like to mention that while a heuristic energy-efficient task scheduling algorithm such as LRF indeed performs very well, we would not be able to know its performance ratio, i.e., how close its performance is when compared with an optimal algorithm. The significance of the lower bounds obtained in this paper is that for any heuristic algorithm, we are able to know the performance of the algorithm when compared with an optimal algorithm, since the performance ratio is no greater than NSL or NEC, which are analytically and experimentally available.

We would like to emphasize that the performance data of the above evaluated algorithms actually depend on the derived lower bounds in this paper. The simulation data for the expected NSL and the expected NEC depend not only on the performance of a scheduling algorithm, but also on the quality of the lower bounds. The close-to-optimal performance of the list scheduling algorithms implies that our lower bounds are of high quality. These lower bounds can be applied to evaluate other energy-efficient task scheduling algorithms with both dynamic and static power consumptions.

7. Extensions

Assume that we are given n independent parallel tasks to be executed on m identical processors. Task i requires π_i processors to execute, where $1 \le i \le n$, and any π_i of the m processors can be allocated to task i. It is possible that in executing task i, the π_i processors may have different execution requirements. Let r_i represent the maximum execution requirement of the π_i processors executing task i. The execution time of task i is $t_i = r_i/s_i$, where s_i is the execution speed. Note that all the π_i processors allocated to task i have the same execution speed s_i for duration t_i, although some of the π_i processors may be idle for some time. The amount of work performed for task i is $w_i = \pi_i t_i$. We use p_i to represent the power required to execute task i, which is $p_i = s_i^\alpha + p$. The energy consumed to execute task i is $e_i = \pi_i p_i t_i = \pi_i(s_i^\alpha + p)(r_i/s_i)$. Let

$$W = w_1 + w_2 + \cdots + w_n = \pi_1 r_1 + \pi_2 r_2 + \cdots + \pi_n r_n$$

denote the total amount of work to be performed for the n tasks.

Imagine that each parallel task i is broken into π_i sequential tasks, each having execution requirement r_i. It is clear that any speed setting and schedule of the n parallel tasks is also a legitimate speed setting and schedule of the $n' = \pi_1+\pi_2+\cdots+\pi_n$ sequential tasks. However, it is more flexible to schedule the n' sequential tasks, since the π_i sequential tasks obtained from parallel task i can have different execution speeds and do not need to be scheduled simultaneously. Hence, the optimal schedule length of the n' sequential tasks is no longer than the optimal schedule length of the n parallel tasks. The optimal schedule length of the n' sequential tasks has a lower bound given in Theorem 4, where R is the total execution requirement of the n' sequential tasks. It is clear that $R = \pi_1 r_1 + \pi_2 r_2 + \cdots + \pi_n r_n = W$. Therefore, the optimal schedule length of the n parallel tasks also has a lower bound given in Theorem 4, with R replaced by W.

For n precedence constrained tasks (sequential or parallel), we have a similar argument. Imagine that the n tasks are treated as independent tasks. It is clear that any speed setting and schedule of the n precedence constrained tasks is also a legitimate speed setting and schedule of the n independent tasks. However, it is more flexible to schedule the n independent tasks, which can have arbitrary execution order. Hence, the optimal schedule length of the n independent tasks is no longer than the optimal schedule length of the n precedence constrained tasks.

In our study, it has been assumed that task execution speed is a continuous and unbounded variable. For a real processor, its speed has several discrete and bounded levels. It is clear that the optimal schedule length on processors with continuous and unbounded speed levels is no longer than the optimal schedule length on processors with discrete or bounded speed levels.

The above discussion can be summarized in the following theorem.

Theorem 9. *For a set of tasks (sequential or parallel, and independent or precedence constrained) with total execution requirement W on m processors (with discrete or continuous speed levels, and bounded or unbounded speed ranges), let T be the solution to the following equation,*

$$mpT^\alpha - ET^{\alpha-1} + \frac{W^\alpha}{m^{\alpha-1}} = 0,$$

where

$$T \in \left(0, \left(1 - \frac{1}{\alpha}\right)\frac{E}{mp}\right]$$

The optimal schedule length is $T^* \ge T$. The lower bound is tight.

Similarly, we can prove the following theorem.

Theorem 10. *For a set of tasks (sequential or parallel, independent or precedence constrained) with total execution requirement W on m processors (with discrete or continuous speed levels, and bounded or unbounded speed ranges), let*

$$E = \frac{W^\alpha}{(mT)^{\alpha-1}} + mpT.$$

The optimal energy consumption is $E^* \ge E$. The lower bound is tight.

8. Conclusions

We have addressed energy-efficient task scheduling when static power consumption is a significant part of energy consumption which cannot be ignored. We have made the following progress. We investigated the problems of scheduling a set of independent sequential tasks on identical processors so that the schedule length is minimized for a given energy consumption constraint or the energy consumption is minimized for a given schedule length constraint. In particular, for a given schedule, we are able to find the optimal task execution speed setting analytically for delay and energy minimization. We have established lower bounds for the minimum schedule length of a set of tasks with a given energy consumption constraint and the minimum energy consumption of a set of tasks with a given schedule length constraint. Our lower bounds are applicable to sequential or parallel, and independent or precedence constrained tasks, on processors with discrete or continuous speed levels, and bounded or unbounded speed ranges. The significance of these lower bounds is that they can be used to evaluate the performance of any heuristic algorithms when compared with optimal algorithms. As an example, we have performed experimental study on the performance of list scheduling algorithms and showed that their performance is very close to the optimal. The paper has made significant contributions to analytical study of energy-efficient task scheduling with both dynamic and static power consumptions.

Acknowledgments

The author wishes to express his gratitude to the editor and two anonymous reviewers for their constructive comments which help to improve the presentation of the paper.

References

[1] S. Albers, Energy-efficient algorithms, Commun. ACM 53 (5) (2010) 86–96.
[2] A. Antoniadis, C.-C. Huang, S. Ott, A fully polynomial-time approximation scheme for speed scaling with sleep state, in: Proceedings of the 26th Annual ACM-SIAM Symposium on Discrete Algorithms, 2015, pp. 1102–1113.
[3] P. Baptiste, M. Chrobak, C. Dürr, Polynomial-time algorithms for minimum energy scheduling, ACM Trans. Algorithms 8 (3) (2012) article (26).
[4] S. Borkar, Design challenges of technology scaling, IEEE Micro 19 (4) (1999) 23–29.
[5] J.A. Butts, G.S. Sohi, A static power model for architects, in: Proceedings of the 33rd ACM/IEEE International Symposium on Microarchitecture, 2000, pp. 191–201.
[6] J. Cao, K. Li, I. Stojmenovic, Optimal power allocation and load distribution for multiple heterogeneous multicore server processors across clouds and data centers, IEEE Trans. Comput. 63 (1) (2014) 45–58.
[7] S.-H. Chan, T.-W. Lam, L.-K. Lee, C.-M. Liu, H.-F. Ting, Sleep management on multiple machines for energy and flow time, Lecture Notes in Comput. Sci. 6755 (2011) 219–231.
[8] P. Chrétienne, E.G. Coffman, J.K. Lenstra, Z. Liu (Eds.), Scheduling Theory and Its Applications, John Wiley & Sons, Chichester, England, 1995.
[9] 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.
[10] M.E.T. Gerards, Algorithmic Power Management – Energy Minimization under Real-Time Constraints (Ph.D. thesis), University of Twente, Netherlands, 2014.
[11] R.L. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (2) (1969) 416–429.
[12] D.S. Hochbaum (Ed.), Approximation Algorithms for NP-Hard Problems, PWS Publishing Company, Boston, MA, 1997.
[13] S. Irani, S. Shukla, R. Gupta, Algorithms for power savings, ACM Trans. Algorithms 3 (4) (2007) Article (41).
[14] 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.
[15] 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.
[16] K. Li, Energy efficient scheduling of parallel tasks on multiprocessor computers, J. Supercomput. 60 (2) (2012a) 223–247.
[17] K. Li, Scheduling precedence constrained tasks with reduced processor energy on multiprocessor computers, IEEE Trans. Comput. 61 (12) (2012b) 1668–1681.
[18] K. Li, Power and performance management for parallel computations in clouds and data centers, J. Comput. System Sci. 82 (2016a) 174–190.
[19] K. Li, Energy and time constrained task scheduling on multiprocessor computers with discrete speed levels, J. Parallel Distrib. Comput. 95 (2016b) 15–28.
[20] K. Li, Energy-efficient task scheduling on multiple heterogeneous computers: algorithms, analysis, and performance evaluation, IEEE Trans. Sustain. Comput. 1 (1) (2016c) 7–19.
[21] K. Li, Scheduling parallel tasks with energy and time constraints on multiple manycore processors in a cloud computing environment, Future Gener. Comput. Syst. 82 (2018) 591–605.
[22] 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.
[23] A.F. Lorenzon, M.C. Cera, A.C.S. Beck, On the influence of static power consumption in multicore embedded systems, in: IEEE International Symposium on Circuits and Systems, Lisbon, Portugal, 2015, pp. 24–27.
[24] A.F. Lorenzon, M.C. Cera, A.C.S. Beck, Investigating different general-purpose and embedded multicores to achieve optimal trade-offs between performance and energy, J. Parallel Distrib. Comput. 95 (2016) 107–123.
[25] J. Mei, K. Li, K. Li, Energy-aware task scheduling in heterogeneous computing environments, Cluster Comput. 17 (2) (2014) 537–550.
[26] K. Pruhs, R. van Stee, P. Uthaisombut, Speed scaling of tasks with precedence constraints, Theory Comput. Syst. 43 (1) (2008) 67–80.
[27] 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.
[28] J. Srinivasan, An Overview of Static Power Dissipation, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.4173.
[29] 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.
[30] G. Xie, Y. Chen, X. Xiao, C. Xu, R. Li, K. Li, Energy-efficient fault-tolerant scheduling of reliable parallel applications on heterogeneous embedded systems, IEEE Trans. Sustain. Comput. 3 (3) (2018) 167–181.
[31] G. Xie, J. Jiang, R. Li, K. Li, Minimizing energy consumption of real-time parallel applications using downward and upward approaches on heterogeneous systems, IEEE Trans. Ind. Inform. 13 (3) (2017a) 1068–1078.
[32] G. Xie, G. Zeng, R. Li, K. Li, Energy-aware processor merging algorithms for deadline-constrained parallel applications in heterogeneous cloud computing, IEEE Trans. Sustain. Comput. 2 (2) (2017b) 62–75.
[33] G. Xie, G. Zeng, R. Li, K. Li, Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems, IEEE Trans. Parallel Distrib. Syst. 28 (12) (2017c) 3426–3442.
[34] Y. Xu, K. Li, L. He, L. Zhang, K. Li, A hybrid chemical reaction optimization scheme for task scheduling on heterogeneous computing systems, IEEE Trans. Parallel Distrib. Syst. 26 (12) (2015) 3208–3222.
[35] Y. Xu, K. Li, J. Hu, K. Li, A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, Inform. Sci. 270 (2014) 255–287.
[36] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, Inform. Sci. 319 (2015) 113–131.
[37] B. Zhao, H. Aydin, D. Zhu, On maximizing reliability of real-time embedded applications under hard energy constraint, IEEE Trans. Ind. Inform. 6 (3) (2010) 316–328.
[38] 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.
[39] 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.
[40] 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.
[41] 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.
[42] D. Zwillinger (Ed.), Standard Mathematical Tables and Formulae, thirtyth ed., CRC Press, Boca Raton, FL, 1996.

Dr. 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 590 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He is currently serving or has served on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Services Computing*, and *IEEE Transactions on Sustainable Computing*. He is an IEEE Fellow.