Scheduling Precedence Constrained Tasks with Reduced Processor Energy on Multiprocessor Computers

Keqin Li, *Senior Member*, *IEEE*

Abstract—Energy-efficient scheduling of sequential tasks with precedence constraints on multiprocessor computers with dynamically variable voltage and speed is investigated as combinatorial optimization problems. In particular, the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint are considered. Our scheduling problems contain three nontrivial subproblems, namely, precedence constraining, task scheduling, and power supplying. Each subproblem should be solved efficiently so that heuristic algorithms with overall good performance can be developed. Such decomposition of our optimization problems into three subproblems makes design and analysis of heuristic algorithms tractable. Three types of heuristic power allocation and scheduling algorithms are proposed for precedence constrained sequential tasks with energy and time constraints, namely, prepower-determination algorithms, postpower-determination algorithms, and hybrid algorithms. The performance of our algorithms are analyzed and compared with optimal schedules analytically. Such analysis has not been conducted in the literature for any algorithm. Therefore, our investigation in this paper makes initial contribution to analytical performance study of heuristic power allocation and scheduling algorithms for precedence constrained sequential tasks. Our extensive simulation data demonstrate that for wide task graphs, the performance ratios of all our heuristic algorithms approach one as the number of tasks increases.

Index Terms—Energy consumption, list scheduling, performance analysis, power-aware scheduling, precedence constraint, simulation, task scheduling

1 Introduction

Performance-driven computer development has lasted for over six decades. Computers have been developed to achieve higher performance. While performance/hardware-cost has increased dramatically, power consumption in computer systems has also increased according to Moore's law [3]. To achieve higher computing performance per processor, microprocessor manufacturers have doubled the power density at an exponential speed over decades, which will soon reach that of a nuclear reactor [37]. Such increased energy consumption causes severe economic, ecological, and technical problems. Power conservation is critical in many computation and communication environments and has attracted extensive research activities. Reducing processor energy 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. There exists a large body of literature on power-aware computing and communication. The reader is referred to [5], [9], [36], [37] for comprehensive surveys.

There are two approaches to reducing power consumption in computing systems. The first approach is the method of thermal-aware hardware design, which can be carried out at various levels, including device-level power reduction, circuit and logic-level techniques, architecture-level power reduction (low-power processor architecture adaptations, low-power memories and memory hierarchies, and low-power interconnects). Low-power consumption and high-system reliability, availability, and usability are main concerns of modern high-performance computing system development. In addition to the traditional performance measure using FLOPS, the Green500 list uses FLOPS per Watt to rank the performance of computing systems, so that the awareness of other performance metrics such as energy efficiency and system reliability can be raised [4]. The second approach to reducing energy consumption in computing systems is the method of power-aware software design at various levels, including operating system-level power management, compiler-level power management, application-level power management, cross-layer (from transistors to applications) adaptations. The power reduction technique discussed in this paper belongs to the operating system level.

Software techniques for power reduction are supported by a mechanism called *dynamic voltage scaling* [2] (equivalently, dynamic frequency scaling, dynamic speed scaling, dynamic power scaling). 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 [35]. Power-aware task scheduling on processors with variable voltages and speeds has been extensively studied since mid 1990s. In a pioneering paper [38], the Weiser et al. first

• *The author is with the Department of Computer Science, State University of New York, New Paltz, New York 12561. E-mail: lik@newpaltz.edu.*

Manuscript received 31 Oct. 2011; revised 17 Feb. 2012; accepted 22 Apr. 2012; published online 30 May 2012.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TCSI-2011-10-0795.
Digital Object Identifier no. 10.1109/TC.2012.120.

proposed the approach to energy saving by using fine grain control of CPU speed by an operating system scheduler. The main idea is to monitor CPU idle time and to reduce energy consumption by reducing clock speed and idle time to a minimum. In a subsequent work [40], Yao et al. analyzed offline and online algorithms for scheduling tasks with arrival times and deadlines on a uniprocessor computer with minimum energy consumption. These research have been extended in [7], [11], [19], [26], [27], [28], and [41] and inspired substantial further investigation, much of which focus on real-time applications, namely, adjusting the supply voltage and clock frequency to minimize CPU energy consumption while still meeting the deadlines for task execution. In [6], [15], [16], [18], [21], [29], [30], [31], [33], [34], [39], [43], [44], [45], and [46] and many other related work, the authors addressed the problem of scheduling independent or precedence constrained tasks on uniprocessor or multiprocessor computers where the actual execution time of a task may be less than the estimated worst case execution time. The main issue is energy reduction by slack time reclamation.

There are two considerations in dealing with the energy-delay tradeoff. On the one hand, in high performance computing systems, power-aware design techniques and algorithms attempt to maximize performance under certain energy consumption constraints. On the other hand, low-power and energy-efficient design techniques and algorithms aim to minimize energy consumption while still meeting certain performance goals. In [8], Barnett studied the problems of minimizing the expected execution time given a hard energy budget and minimizing the expected energy expenditure given a hard execution deadline for a single task with randomized execution requirement. In [10], Bunde considered scheduling jobs with equal requirements on multiprocessors. In [13], the Cho and Melhem studied the relationship among parallelization, performance, and energy consumption, and the problem of minimizing energy-delay product. In [17] Khan and Ahmad and [20] Lee and Zomaya, attempted joint minimization of energy consumption and task execution time. In [32], Rusu et al. investigated the problem of system value maximization subject to both time and energy constraints.

In [22], [23], and [25], we addressed energy and time constrained power allocation and task scheduling on multiprocessor computers with dynamically variable voltage and frequency and speed and power as combinatorial optimization problems. In particular, we defined the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint on multiprocessor computers. The first problem has applications in general multiprocessor and multicore processor computing systems where energy consumption is an important concern and in mobile computers where energy conservation is a main concern. The second problem has applications in real-time multiprocessing systems and environments such as parallel signal processing, automated target recognition, and real-time MPEG encoding, where timing constraint is a major requirement. Our scheduling problems are defined such that the energy-delay product is optimized by fixing one factor and minimizing the other. In [22], and [25], we studied the problems of scheduling independent sequential tasks. In [23], and [24], we studied the problems of scheduling independent parallel tasks.

In this paper, we investigate scheduling sequential tasks with precedence constraints on multiprocessor computers with dynamically variable voltage and speed as combinatorial optimization problems. Our scheduling problems contain three nontrivial subproblems, namely, precedence constraining, task scheduling, and power supplying.

- *Precedence Constraining.* Precedence constraints make design and analysis of heuristic scheduling algorithms more difficult.
- *Task Scheduling.* It is NP-hard even scheduling independent sequential tasks without precedence constraint.
- *Power Supplying.* Tasks should be supplied with appropriate powers and execution speeds, such that the schedule length is minimized by consuming given amount of energy or the energy consumed is minimized without missing a given deadline.

Each subproblem should be solved efficiently so that heuristic algorithms with overall good performance can be developed.

There are naturally three types of power-aware task scheduling algorithms, depending on the order of power supplying and task scheduling.

- *Prepower-Determination Algorithms.* In this type of algorithms, we first determine power supplies and then schedule the tasks.
- *Postpower-Determination Algorithms.* In this type of algorithms, we first schedule the tasks and then determine power supplies.
- *Hybrid Algorithms.* In this type of algorithms, scheduling tasks and determining power supplies are interleaved among different stages of an algorithm.

We will propose heuristic power allocation and scheduling algorithms of the above three types for precedence constrained sequential tasks with energy and time constraints. We will also analyze their performance and demonstrate simulation results. Notice that we compare the performance of our algorithms with optimal schedules analytically. Such analysis has not been conducted in the literature for any algorithm. Therefore, our investigation in this paper makes initial contribution to analytical performance study of heuristic power allocation and scheduling algorithms for precedence constrained sequential tasks.

The rest of the paper is organized as follows: In Section 2, we provide background information, including the power consumption model, definitions of our problems, lower bounds for optimal solutions, and performance measures. In Section 3, we propose and analyze prepower-determination algorithms. In Section 4, we propose and analyze postpower-determination algorithms. In Section 5, we propose and analyze hybrid algorithms. In Section 6, we present simulation data and show that for wide task graphs, the performance ratios of all our heuristic algorithms approach one as the number of tasks increases. We conclude the paper in Section 7.

2 PRELIMINARIES

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 [12]. Since $s \propto f$, where s is the processor speed, and $f \propto V^\phi$ with $0 < \phi \leq 1$ [42], which implies that $V \propto f^{1/\phi}$, we know that power consumption is $p \propto f^\alpha$ and $p \propto s^\alpha$, where $\alpha = 1 + 2/\phi \geq 3$.

It is clear that due to precedence constraints, a processor may be idle between execution of tasks. It is well known that an idle processor also consumes certain power, which includes static power dissipation, short circuit power dissipation, and other leakage and wasted power [1]. In this paper, we do not include this part of the power consumption into consideration. We are only interested in managing the switching power consumption by dynamic voltage/frequency/speed/power scaling. Therefore, by power/energy consumption we mean dynamic power/energy consumption. We assume that a processor consumes no dynamic energy when it is idle.

Assume that we are given n precedence constrained sequential 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, where $1 \leq i \leq n$. We use p_i to represent the power allocated to execute task i. For ease of discussion, we will assume that p_i is simply s_i^α, where $s_i = p_i^{1/\alpha}$ is the execution speed of task i. The execution time of task i is $t_i = r_i/s_i = r_i/p_i^{1/\alpha}$. The energy consumed to execute task i is $e_i = p_i t_i = r_i p_i^{1-1/\alpha} = r_i s_i^{\alpha-1}$.

Given 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 find the power supplies $p_1, p_2, \ldots, p_n$ 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.

Given 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 find the power supplies $p_1, p_2, \ldots, p_n$ 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.

Notice that we assume that the m processors are tightly coupled with certain highly efficient communication mechanisms such as shared memory. Therefore, we will not include the communication times among the tasks into our problems. This is because the data produced by a predecessor can be readily accessible by a successor without any delay. In other words, we focus on computation times and processor energy consumption, not communication times and network energy consumption. This is different from [47] that addresses task scheduling on clusters which contain significant communication costs. Our computing model is the class of multiprocessor computers with negligible communication costs.

To compare the solutions of our algorithms with optimal solutions, we need lower bounds for the optimal solutions. Let $R = r_1 + r_2 + \cdots + r_n$ be the total execution requirement of the n tasks.

The following theorem gives a lower bound for the optimal schedule length T^* for the problem of minimizing schedule length with energy consumption constraint when tasks are independent [22], and is certainly applicable to tasks with precedence constraints.

Theorem 1. *For the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer, we have the following lower bound,*

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

for the optimal schedule length T^*.

Notice that the above lower bound can be achieved only when the workload R can be evenly distributed among the m processors.

The following theorem gives a lower bound for the minimum energy consumption E^* for the problem of minimizing energy consumption with schedule length constraint when tasks are independent [22], and is certainly applicable to tasks with precedence constraints.

Theorem 2. *For the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer, we have the following lower bound,*

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

for the minimum energy consumption E^*.

Again, the above lower bound can be achieved only when the workload R can be evenly distributed among the m processors.

We define the *performance ratio* as $\beta = T/T^*$ and the *asymptotic performance ratio* as $\beta_\infty = \lim_{R/r^* \to \infty} \beta$ (by fixing m) for heuristic algorithms that solve the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer, where T is the length of the schedule produced by an algorithm, and $r^* = \max(r_1, r_2, \ldots, r_n)$ is the maximum task execution requirement.

We define the *performance ratio* as $\beta = E/E^*$ and the *asymptotic performance ratio* as $\beta_\infty = \lim_{R/r^* \to \infty} \beta$ (by fixing m) for heuristic algorithms that solve the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer, where E is the amount of energy consumed by an algorithm, and $r^* = \max(r_1, r_2, \ldots, r_n)$ is the maximum task execution requirement.

An algorithm is asymptotically optimal if the asymptotic performance ratio is $\beta_\infty = 1$.

Notice that the condition $R/r^* \to \infty$ can be satisfied typically when the r_i's are bounded from above and the number of tasks increases. In this case, each individual r_i becomes smaller and smaller when compared with R.

3 Prepower-Determination Algorithms

The prepower-determination algorithms in this section are called ES-A, where ES stands for *equal-speed*, and A is a list scheduling algorithm. The strategies to solve the three subproblems are described as follows:

- *Power Supplying.* All tasks are supplied with the same power and executed with the same speed (hence the name equal speed).
- *Task Scheduling.* The list scheduling algorithms are used to schedule the tasks.
- *Precedence Constraining.* Precedence constraints are dealt with by a list scheduling algorithm during task scheduling.

The class of list scheduling algorithms were originally designed for scheduling tasks with precedence constraints [14]. Given a list of precedence constrained tasks, a list scheduling algorithm works as follows: as soon as a processor is available, the first ready task whose predecessors have all been completed is scheduled on that processor and removed from the list. This process repeats until all tasks in the list are finished. There are different strategies in the initial ordering of the tasks. We mention three of them here.

- *List Scheduling* (LS). The tasks are arranged in a random order.
- *Largest Requirement First* (LRF). The tasks are arranged such that $r_1 \geq r_2 \geq \cdots \geq r_n$.
- *Smallest Requirement First* (SRF). The tasks are arranged such that $r_1 \leq r_2 \leq \cdots \leq r_n$.

In this paper, we consider $A \in \{\text{LS}, \text{LRF}, \text{SRF}\}$. Hence, we have three prepower-determination algorithms, namely, ES-LS, ES-LRF, and ES-SRF.

3.1 Minimizing Schedule Length

Let $A(t_1, t_2, \ldots, t_n)$ represent the length of the schedule produced by algorithm A for n precedence constrained tasks with execution times $t_1, t_2, \ldots, t_n$, where A is a list scheduling algorithm. The following theorem gives the performance ratio of algorithm ES-A to solve the problem of minimizing schedule length with energy consumption constraint.

Theorem 3. *By using algorithm ES-A to solve the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer, the schedule length is*

$$T = A(r_1, r_2, ..., r_n)\left(\frac{R}{E}\right)^{1/(\alpha-1)},$$

where A is a list scheduling algorithm. The performance ratio is

$$\beta \leq \frac{A(r_1, r_2, \ldots, r_n)}{R/m}.$$

For independent tasks, as $R/r^* \to \infty$, the asymptotic performance ratio is $\beta_\infty = 1$.

Proof. To solve the problem of minimizing schedule length with energy consumption constraint E by using algorithm ES-A, we have $p_1 = p_2 = \cdots = p_n = p$ and $s_1 = s_2 = \cdots = s_n = s$. Based on the fact that

$$E = r_1 p^{1-1/\alpha} + r_2 p^{1-1/\alpha} + \cdots + r_n p^{1-1/\alpha} = R p^{1-1/\alpha},$$

we get $p = (E/R)^{\alpha/(\alpha-1)}$, and $s = p^{1/\alpha} = (E/R)^{1/(\alpha-1)}$, and $t_i = r_i/s = r_i(R/E)^{1/(\alpha-1)}$. We notice that for all $x \geq 0$, we have $A(t_1, t_2, \ldots, t_n) = xA(t'_1, t'_2, \ldots, t'_n)$, if $t_i = xt'_i$ for all $1 \leq i \leq n$. That is, the schedule length is scaled by a factor of x if all the task execution times are scaled by a factor of x. When the n tasks with execution times $t_1, t_2, \ldots, t_n$ are scheduled by using a list scheduling algorithm A, we get the length of the schedule produced by algorithm ES-A as

$$T = A(t_1, t_2, \ldots, t_n) = A(r_1, r_2, \ldots, r_n)\left(\frac{R}{E}\right)^{1/(\alpha-1)}.$$

Thus, by Theorem 1, we get

$$\beta = \frac{T}{T^*} \leq \frac{A(r_1, r_2, \ldots, r_n)}{R/m}.$$

For any list scheduling algorithm A and independent tasks, we have

$$A(r_1, r_2, \ldots, r_n) \leq \frac{R}{m} + r^*.$$

The asymptotic optimality follows the last two inequalities. □

The power supply to each task is simply $(E/R)^{\alpha/(\alpha-1)}$.

3.2 Minimizing Energy Consumption

The following theorem gives the performance ratio of algorithm ES-A to solve the problem of minimizing energy consumption with schedule length constraint.

Theorem 4. *By using algorithm ES-A to solve the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer, the energy consumed is*

$$E = \left(\frac{A(r_1, r_2, \ldots, r_n)}{T}\right)^{\alpha-1} R,$$

where A is a list scheduling algorithm. The performance ratio is

$$\beta \leq \left(\frac{A(r_1, r_2, \ldots, r_n)}{R/m}\right)^{\alpha-1}.$$

For independent tasks, as $R/r^* \to \infty$, the asymptotic performance ratio is $\beta_\infty = 1$.

Proof. To solve the problem of minimizing energy consumption with schedule length constraint T by using algorithm ES-A, we need E such that

$$A(t_1, t_2, \ldots, t_n) = A(r_1, r_2, \ldots, r_n)\left(\frac{R}{E}\right)^{1/(\alpha-1)} = T,$$

which gives rise to

$$E = \left(\frac{A(r_1, r_2, \ldots, r_n)}{T}\right)^{\alpha-1} R.$$

By Theorem 2, we get

$$\beta = \frac{E}{E^*} \leq \left(\frac{A(r_1, r_2, \ldots, r_n)}{R/m}\right)^{\alpha-1}.$$

The asymptotic optimality for independent tasks follows the last inequality. □

The power supply to each task is simply $(E/R)^{\alpha/(\alpha-1)} = (A(r_1, r_2, \ldots, r_n)/T)^{\alpha}$.

4 Postpower-Determination Algorithms

The postpower-determination algorithms in this section are called LL-A, where LL stands for *level by level*, and A is a list scheduling algorithm. In this paper, we assume $A = \text{LS}, \text{LRF}, \text{SRF}$. Hence, we have three postpower-determination algorithms, namely, LL-LS, LL-LRF, and LL-SRF. The strategies to solve the three subproblems are described as follows:

- *Precedence Constraining.* We propose to use level-by-level scheduling algorithms to deal with precedence constraints.
- *Task Scheduling.* Since tasks in the same level are independent of each other, they can be scheduled by any of the efficient algorithms such as list scheduling algorithms previously developed for scheduling independent tasks.
- *Power Supplying.* We then find the optimal power supplies for the given schedule. We adopt a two-level energy/time/power allocation scheme for a given schedule, namely, optimal energy/time allocation among levels of tasks (Theorems 6 and 8) and optimal energy allocation among groups of tasks in the same level and optimal power supplies to tasks in the same group (Theorems 5 and 7).

The decomposition of scheduling precedence constrained tasks into scheduling levels of independent tasks makes analysis of level-by-level scheduling algorithms tractable.

A set of n tasks with precedence constraints can be represented by a partial order $\prec$ on the tasks, i.e., for two tasks i and j, if $i \prec j$, then task j cannot start its execution until task i finishes. It is clear that the n tasks and the partial order $\prec$ can be represented by a directed task graph, in which, there are n vertices for the n tasks and (i, j) is an arc if and only if $i \prec j$. Furthermore, such a task graph must be a directed acyclic graph (dag). An arc (i, j) is redundant if there exists k such that (i, k) and (k, j) are also arcs in the task graph. We assume that there is no redundant arc in the task graph.

A dag can be decomposed into levels, with v being the number of levels. Tasks with no predecessors (called initial tasks) constitute level 1. Generally, a task i is in level l if the number of nodes on the longest path from some initial task to task i is l, where $1 \le l \le v$. Note that all tasks in the same level are independent of each other, and hence, they can be scheduled by any of the algorithms (e.g., list scheduling algorithms) for scheduling independent tasks. Algorithm LL-A, standing for level-by-level scheduling with algorithm A, where A is a list scheduling algorithm, schedules the n tasks level by level in the order level 1, level 2, ..., level v. Tasks in level $l+1$ cannot start their execution until all tasks in level l are completed. For each level l, where $1 \le l \le v$, we use algorithm A to generate its schedule.

In the following, we analyze the performance of algorithm LL-A, and also specify energy/time allocation among levels of tasks, energy allocation among groups of tasks in the same level, and power supplies to tasks in the same group, thus completing the description of our postpower-determination algorithms.

4.1 Minimizing Schedule Length

Assume that a set of n independent tasks is partitioned 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 the total execution requirement of the tasks in group k. For a given partition of the n tasks into m groups, we are seeking power supplies that minimize the schedule length. Let E_k be the energy consumed by all the tasks in group k. The following result characterizes the optimal power supplies [22].

Theorem 5. *For a given partition $R_1, R_2, \ldots, R_m$ of the n tasks into m groups on a multiprocessor computer, the schedule length is minimized when all tasks in group k are executed with the same power $(E_k/R_k)^{\alpha/(\alpha-1)}$, where*

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

for all $1 \le k \le m$. The optimal schedule length is

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

for the above power supplies.

To use a postpower-determination algorithm to solve the problem of minimizing schedule length with energy consumption constraint E, we need to allocate the available energy E to the v levels. We use $E_1, E_2, \ldots, E_v$ to represent an energy allocation to the v levels, where level l consumes energy E_l, and $E_1 + E_2 + \cdots + E_v = E$. Let n_l be the number of tasks in level l, and $r_{l,1}, r_{l,2}, \ldots, r_{l,n_l}$ be the execution requirements of the n_l tasks in level l, and $R_l = r_{l,1} + r_{l,2} + \cdots + r_{l,n_l}$ be the total execution requirement of tasks in level l, where $1 \le l \le v$. Theorem 6 provides optimal energy allocation to the v levels for minimizing schedule length with energy consumption constraint in scheduling precedence constrained tasks by using scheduling algorithms LL-A, where A is a list scheduling algorithm.

Theorem 6. *For a given partition $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ of the n_l tasks in level l into m groups produced by a list scheduling algorithm A, where $1 \le l \le v$, and an energy allocation $E_1, E_2, \ldots, E_v$ to the v levels, algorithm LL-A produces schedule length*

$$T = \sum_{l=1}^{v} \left(\frac{R_{l,1}^{\alpha} + R_{l,2}^{\alpha} + \cdots + R_{l,m}^{\alpha}}{E_l}\right)^{1/(\alpha-1)}.$$

The energy allocation $E_1, E_2, \ldots, E_v$ which minimizes T is

$$E_l = \left(\frac{S_l^{1/\alpha}}{S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}}\right) E,$$

where $S_l = R_{l,1}^{\alpha} + R_{l,2}^{\alpha} + \cdots + R_{l,m}^{\alpha}$, for all $1 \le l \le v$, and the minimized schedule length is

$$T = \frac{(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha})^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}},$$

by using the above energy allocation. The performance ratio is

$$\beta \le \left(1 + \frac{mvr^*}{R}\right)^{\alpha/(\alpha-1)},$$

where $r^* = \max(r_1, r_2, \ldots, r_n)$ is the maximum task execution requirement. For a fixed m, the performance ratio β can be arbitrarily close to 1 as $R/(vr^*) \to \infty$.

Proof. By Theorem 5, for a given partition $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ of the n_l tasks in level l into m groups, the schedule length T_l for level l is minimized when all tasks in group k of level l are executed with the same power $(E_{l,k}/R_{l,k})^{\alpha/(\alpha-1)}$, where

$$E_{l,k} = \left(\frac{R_{l,k}^\alpha}{R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha}\right) E_l,$$

for all $1 \le l \le v$ and $1 \le k \le m$. The optimal schedule length is

$$T_l = \left(\frac{R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha}{E_l}\right)^{1/(\alpha-1)},$$

for the above power supplies. Since the level-by-level scheduling algorithm produces schedule length $T = T_1 + T_2 + \cdots + T_v$, we have

$$T = \sum_{l=1}^{v} \left(\frac{R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha}{E_l}\right)^{1/(\alpha-1)}.$$

By the definition of S_l, we obtain

$$T = \left(\frac{S_1}{E_1}\right)^{1/(\alpha-1)} + \left(\frac{S_2}{E_2}\right)^{1/(\alpha-1)} + \cdots + \left(\frac{S_v}{E_v}\right)^{1/(\alpha-1)}.$$

To minimize T, we use the Lagrange multiplier system

$$\nabla T(E_1, E_2, \ldots, E_v) = \lambda \nabla F(E_1, E_2, \ldots, E_v),$$

where λ is the Lagrange multiplier, and F is the constraint $E_1 + E_2 + \cdots + E_v - E = 0$. Since

$$\frac{\partial T}{\partial E_l} = \lambda \frac{\partial F}{\partial E_l},$$

that is,

$$S_l^{1/(\alpha-1)} \left(-\frac{1}{\alpha - 1}\right) \frac{1}{E_l^{1/(\alpha-1)+1}} = \lambda,$$

for all $1 \le l \le v$, we get

$$E_l = S_l^{1/\alpha} \left(\frac{1}{\lambda(1-\alpha)}\right)^{(\alpha-1)/\alpha},$$

which implies that

$$E = \left(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}\right) \left(\frac{1}{\lambda(1-\alpha)}\right)^{(\alpha-1)/\alpha},$$

and

$$E_l = \left(\frac{S_l^{1/\alpha}}{S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}}\right) E,$$

for all $1 \le l \le v$. By using the above energy allocation, we have

$$\begin{aligned} T &= \sum_{l=1}^{v} \left(\frac{S_l}{E_l}\right)^{1/(\alpha-1)} \\ &= \sum_{l=1}^{v} \frac{S_l^{1/(\alpha-1)}}{\left(\left(\dfrac{S_l^{1/\alpha}}{S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}}\right) E\right)^{1/(\alpha-1)}} \\ &= \sum_{l=1}^{v} \frac{S_l^{1/\alpha} \left(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}\right)^{1/(\alpha-1)}}{E^{1/(\alpha-1)}} \\ &= \frac{\left(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}\right)^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}}. \end{aligned}$$

For any list scheduling algorithm A, we have

$$R_{l,k} \le \frac{R_l}{m} + r^*,$$

for all $1 \le l \le v$ and $1 \le k \le m$. Therefore,

$$S_l \le m \left(\frac{R_l}{m} + r^*\right)^\alpha,$$

and

$$S_l^{1/\alpha} \le m^{1/\alpha} \left(\frac{R_l}{m} + r^*\right),$$

and

$$S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha} \le m^{1/\alpha} \left(\frac{R}{m} + vr^*\right),$$

which implies that

$$T \le m^{1/(\alpha-1)} \left(\frac{R}{m} + vr^*\right)^{\alpha/(\alpha-1)} \frac{1}{E^{1/(\alpha-1)}}.$$

By Theorem 1, we get

$$\beta = \frac{T}{T^*} \le \left(1 + \frac{mvr^*}{R}\right)^{\alpha/(\alpha-1)}.$$

It is clear that for a fixed m, β can be arbitrarily close to 1 as $R/(vr^*)$ becomes large. □

Theorems 5 and 6 give the power supply to the tasks in group k of level l as

$$\left(\frac{E_{l,k}}{R_{l,k}}\right)^{\alpha/(\alpha-1)} = \left(\left(\frac{R_{l,k}^\alpha}{R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha}\right) \left(\frac{S_l^{1/\alpha}}{S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}}\right) \frac{E}{R_{l,k}}\right)^{\alpha/(\alpha-1)},$$

for all $1 \le l \le v$ and $1 \le k \le m$.

4.2 Minimizing Energy Consumption

The following result gives the optimal power supplies that minimize energy consumption for a given partition of a set of n independent tasks into m groups on a multiprocessor computer [22].

Theorem 7. *For a given partition $R_1, R_2, \ldots, R_m$ of the n tasks into m groups on a multiprocessor computer, the total energy consumption is minimized when all tasks in group k are executed with the same power $(R_k/T)^\alpha$, where $1 \le k \le m$. The minimum energy consumption is*

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

for the above power supplies.

To use a postpower-determination algorithm to solve the problem of minimizing energy consumption with schedule length constraint T, we need to allocate the time T to the v levels. We use $T_1, T_2, \ldots, T_v$ to represent a time allocation to the v levels, where tasks in level l are executed within deadline T_l, and $T_1 + T_2 + \cdots + T_v = T$. Theorem 8 provides optimal time allocation to the v levels for minimizing energy consumption with schedule length constraint in scheduling precedence constrained tasks by using scheduling algorithms LL-A, where A is a list scheduling algorithm.

Theorem 8. *For a given partition $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ of the n_l tasks in level l into m groups produced by a list scheduling algorithm A, where $1 \le l \le v$, and a time allocation $T_1, T_2, \ldots, T_v$ to the v levels, algorithm LL-A consumes energy*

$$E = \sum_{l=1}^{v} \left(\frac{R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha}{T_l^{\alpha-1}} \right).$$

The time allocation $T_1, T_2, \ldots, T_v$ which minimizes E is

$$T_l = \left(\frac{S_l^{1/\alpha}}{S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}} \right) T,$$

where $S_l = R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha$, for all $1 \le l \le v$, and the minimized energy consumption is

$$E = \frac{\left(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}\right)^\alpha}{T^{\alpha-1}},$$

by using the above time allocation. The performance ratio is

$$\beta \le \left(1 + \frac{mvr^*}{R}\right)^\alpha,$$

where $r^* = \max(r_1, r_2, \ldots, r_n)$ is the maximum task execution requirement. For a fixed m, the performance ratio β can be arbitrarily close to 1 as $R/(vr^*) \to \infty$.

Proof. By Theorem 7, for a given partition $R_{l,1}, R_{l,2}, \ldots, R_{l,m}$ of the n_l tasks in level l into m groups, the total energy E_l consumed by level l is minimized when all tasks in group k are executed with the same power $(R_{l,k}/T_l)^\alpha$, where $1 \le l \le v$ and $1 \le k \le m$. The minimum energy consumption is

$$E_l = \frac{R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha}{T_l^{\alpha-1}},$$

for the above power supplies. Since the level-by-level scheduling algorithm consumes energy $E = E_1 + E_2 + \cdots + E_v$, we have

$$E = \sum_{l=1}^{v} \left(\frac{R_{l,1}^\alpha + R_{l,2}^\alpha + \cdots + R_{l,m}^\alpha}{T_l^{\alpha-1}} \right).$$

By the definition of S_l, we obtain

$$E = \frac{S_1}{T_1^{\alpha-1}} + \frac{S_2}{T_2^{\alpha-1}} + \cdots + \frac{S_v}{T_v^{\alpha-1}}.$$

To minimize E, we use the Lagrange multiplier system

$$\nabla E(T_1, T_2, \ldots, T_v) = \lambda \nabla F(T_1, T_2, \ldots, T_v),$$

where λ is the Lagrange multiplier, and F is the constraint $T_1 + T_2 + \cdots + T_v - T = 0$. Since

$$\frac{\partial E}{\partial T_l} = \lambda \frac{\partial F}{\partial T_l},$$

that is,

$$S_l \left(\frac{1-\alpha}{T_l^\alpha} \right) = \lambda,$$

for all $1 \le l \le v$, we get

$$T_l = S_l^{1/\alpha} \left(\frac{1-\alpha}{\lambda} \right)^{1/\alpha},$$

which implies that

$$T = \left(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}\right) \left(\frac{1-\alpha}{\lambda} \right)^{1/\alpha},$$

and

$$T_l = \left(\frac{S_l^{1/\alpha}}{S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}} \right) T,$$

for all $1 \le l \le v$. By using the above time allocation, we have

$$\begin{aligned} E &= \sum_{l=1}^{v} \frac{S_l}{T_l^{\alpha-1}} \\ &= \sum_{l=1}^{v} \frac{S_l}{\left(\left(\frac{S_l^{1/\alpha}}{S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}} \right) T \right)^{\alpha-1}} \\ &= \sum_{l=1}^{v} \frac{S_l^{1/\alpha} \left(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}\right)^{\alpha-1}}{T^{\alpha-1}} \\ &= \frac{\left(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha}\right)^\alpha}{T^{\alpha-1}}. \end{aligned}$$

Similar to the proof of Theorem 6, we have

$$S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha} \le m^{1/\alpha}\left(\frac{R}{m} + vr^*\right),$$

which implies that

$$E \le m\left(\frac{R}{m} + vr^*\right)^{\alpha} \frac{1}{T^{\alpha-1}}.$$

By Theorem 2, we get

$$\beta = \frac{E}{E^*} \le \left(1 + \frac{mvr^*}{R}\right)^{\alpha}.$$

It is clear that for a fixed m, β can be arbitrarily close to 1 as $R/(vr^*)$ becomes large. □

Theorems 7 and 8 give the power supply to the tasks in group k of level l as

$$\left(\frac{R_{l,k}}{T_l}\right)^{\alpha} = \left(\frac{R_{l,k}(S_1^{1/\alpha} + S_2^{1/\alpha} + \cdots + S_v^{1/\alpha})}{S_l^{1/\alpha} T}\right)^{\alpha},$$

for all $1 \le l \le v$ and $1 \le k \le m$.

5 Hybrid Algorithms

The hybrid algorithms in this section are called LL-ES-A, where LL stands for *level by level*, ES stands for *equal speed*, and A is a list scheduling algorithm. In this paper, we consider $A \in \{\text{LS}, \text{LRF}, \text{SRF}\}$. Hence, we have three hybrid algorithms, namely, LL-ES-LS, LL-ES-LRF, and LL-ES-SRF. The strategies to solve the three subproblems are described as follows:

- *Precedence Constraining.* The level-by-level scheduling algorithms are employed to deal with precedence constraints.
- *Task Scheduling.* Independent tasks in the same level are supplied with the same power and executed with the same speed and scheduled by using list scheduling algorithms.
- *Power Supplying.* We then determine optimal energy/time allocation among levels of tasks for the given schedule (Theorems 9 and 10).

Notice that power allocation and task scheduling are mixed. The equal-speed strategy implies prepower-determination for tasks in the same level. The level-by-level scheduling method implies postpower-determination for tasks of different levels.

5.1 Minimizing Schedule Length

Theorem 9 provides optimal energy allocation to the v levels for minimizing schedule length with energy consumption constraint in scheduling precedence constrained tasks by using hybrid scheduling algorithms LL-ES-A, where A is a list scheduling algorithm.

Theorem 9. *For a given energy allocation $E_1, E_2, \ldots, E_v$ to the v levels, the scheduling algorithm LL-ES-A, where A is a list scheduling algorithm, produces schedule length*

$$T = A_1\left(\frac{R_1}{E_1}\right)^{1/(\alpha-1)} + A_2\left(\frac{R_2}{E_2}\right)^{1/(\alpha-1)} + \cdots + A_v\left(\frac{R_v}{E_v}\right)^{1/(\alpha-1)},$$

where $A_l = A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})$ is the length of the schedule produced by algorithm A for n_l tasks with execution times $r_{l,1}, r_{l,2}, \ldots, r_{l,n_l}$. The en\textbackslash ergy allocation $E_1, E_2, \ldots, E_v$ which minimizes T is

$$E_l = \left(\frac{A_l^{1-1/\alpha} R_l^{1/\alpha}}{A_1^{1-1/\alpha} R_1^{1/\alpha} + A_2^{1-1/\alpha} R_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}}\right) E,$$

for all $1 \le l \le v$, and the minimized schedule length is

$$T = \frac{(A_1^{1-1/\alpha} R_1^{1/\alpha} + A_2^{1-1/\alpha} R_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha})^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}},$$

by using the above energy allocation. The performance ratio is

$$\beta \le \left(1 + \frac{mvr^*}{R}\right)^{\alpha/(\alpha-1)},$$

where $r^* = \max(r_1, r_2, \ldots, r_n)$ is the maximum task execution requirement. For a fixed m, the performance ratio β can be arbitrarily close to 1 as $R/(vr^*) \to \infty$.

Proof. From Theorem 3, we know that the schedule length for the n_l tasks in level l is

$$T_l = A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})\left(\frac{R_l}{E_l}\right)^{1/(\alpha-1)} = A_l\left(\frac{R_l}{E_l}\right)^{1/(\alpha-1)},$$

by using algorithm ES-A, where A is a list scheduling algorithm. Since the level-by-level scheduling algorithm produces schedule length $T = T_1 + T_2 + \cdots + T_v$, we have

$$T = A_1\left(\frac{R_1}{E_1}\right)^{1/(\alpha-1)} + A_2\left(\frac{R_2}{E_2}\right)^{1/(\alpha-1)} + \cdots + A_v\left(\frac{R_v}{E_v}\right)^{1/(\alpha-1)}.$$

To minimize T, we use the Lagrange multiplier system

$$\nabla T(E_1, E_2, \ldots, E_v) = \lambda \nabla F(E_1, E_2, \ldots, E_v),$$

where λ is the Lagrange multiplier, and F is the constraint $E_1 + E_2 + \cdots + E_v - E = 0$. Since

$$\frac{\partial T}{\partial E_l} = \lambda \frac{\partial F}{\partial E_l},$$

that is,

$$A_l R_l^{1/(\alpha-1)}\left(-\frac{1}{\alpha-1}\right)\frac{1}{E_l^{1/(\alpha-1)+1}} = \lambda,$$

for all $1 \le l \le v$, we get

$$E_l = A_l^{1-1/\alpha} R_l^{1/\alpha}\left(\frac{1}{\lambda(1-\alpha)}\right)^{(\alpha-1)/\alpha},$$

which implies that

$$E = \left(A_1^{1-1/\alpha} R_1^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}\right)\left(\frac{1}{\lambda(1-\alpha)}\right)^{(\alpha-1)/\alpha},$$

and

$$E_l = \left(\frac{A_l^{1-1/\alpha} R_l^{1/\alpha}}{A_1^{1-1/\alpha} R_1^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}}\right) E,$$

for all $1 \le l \le v$. By using the above energy allocation, we have

$$\begin{aligned} T &= \sum_{l=1}^{v} A_l \left(\frac{R_l}{E_l}\right)^{1/(\alpha-1)} \\ &= \sum_{l=1}^{v} \frac{A_l R_l^{1/(\alpha-1)}}{\left(\left(\frac{A_l^{1-1/\alpha} R_l^{1/\alpha}}{A_1^{1-1/\alpha} R_1^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}}\right) E\right)^{1/(\alpha-1)}} \\ &= \left(A_1^{1-1/\alpha} R_1^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}\right)^{1/(\alpha-1)} \\ &\quad \sum_{l=1}^{v} \frac{A_l^{1-1/\alpha} R_l^{1/\alpha}}{E^{1/(\alpha-1)}} \\ &= \frac{\left(A_1^{1-1/\alpha} R_1^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}\right)^{\alpha/(\alpha-1)}}{E^{1/(\alpha-1)}}. \end{aligned}$$

Since

$$A_l \le \frac{R_l}{m} + r^*,$$

we obtain

$$\begin{aligned} A_l^{1-1/\alpha} R_l^{1/\alpha} &\le \left(\frac{R_l}{m} + r^*\right)^{1-1/\alpha} R_l^{1/\alpha} \\ &= \left(\frac{R_l + mr^*}{m}\right)^{1-1/\alpha} R_l^{1/\alpha} \\ &\le \left(\frac{R_l + mr^*}{m}\right)^{1-1/\alpha} (R_l + mr^*)^{1/\alpha} \\ &= \frac{R_l + mr^*}{m^{1-1/\alpha}}, \end{aligned}$$

and

$$A_1^{1-1/\alpha} R_1^{1/\alpha} + A_2^{1-1/\alpha} R_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha} \le \frac{R + mvr^*}{m^{1-1/\alpha}},$$

and

$$T \le \frac{(R + mvr^*)^{\alpha/(\alpha-1)}}{mE^{1/(\alpha-1)}}.$$

By Theorem 1, we get

$$\beta = \frac{T}{T^*} \le \left(1 + \frac{mvr^*}{R}\right)^{\alpha/(\alpha-1)}.$$

It is clear that for a fixed m, β can be arbitrarily close to 1 as $R/(vr^*)$ becomes large. □

Theorems 3 and 9 give the power supply to the tasks in level l as

$$\left(\frac{E_l}{R_l}\right)^{\alpha/(\alpha-1)} = \left(\left(\frac{A_l^{1-1/\alpha} R_l^{1/\alpha}}{A_1^{1-1/\alpha} R_1^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}}\right)\frac{E}{R_l}\right)^{\alpha/(\alpha-1)},$$

for all $1 \le l \le v$.

5.2 Minimizing Energy Consumption

Theorem 10 provides optimal time allocation to the v levels for minimizing energy consumption with schedule length constraint in scheduling precedence constrained tasks by using hybrid scheduling algorithms LL-ES-A, where A is a list scheduling algorithm.

Theorem 10. *For a given time allocation $T_1, T_2, \ldots, T_v$ to the v levels, the scheduling algorithm LL-ES-A, where A is a list scheduling algorithm, consumes energy*

$$E = \left(\frac{A_1}{T_1}\right)^{\alpha-1} R_1 + \left(\frac{A_2}{T_2}\right)^{\alpha-1} R_2 + \cdots + \left(\frac{A_v}{T_v}\right)^{\alpha-1} R_v,$$

where $A_l = A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})$ is the length of the schedule produced by algorithm A for n_l tasks with execution times $r_{l,1}, r_{l,2}, \ldots, r_{l,n_l}$. The time allocation $T_1, T_2, \ldots, T_v$ which minimizes E is

$$T_l = \left(\frac{A_l^{1-1/\alpha} R_l^{1/\alpha}}{A_1^{1-1/\alpha} R_1^{1/\alpha} + A_2^{1-1/\alpha} R_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha}}\right) T,$$

for all $1 \le l \le v$, and the minimized energy consumption is

$$E = \frac{(A_1^{1-1/\alpha} R_1^{1/\alpha} + A_2^{1-1/\alpha} R_2^{1/\alpha} + \cdots + A_v^{1-1/\alpha} R_v^{1/\alpha})^{\alpha}}{T^{\alpha-1}},$$

by using the above time allocation. The performance ratio is

$$\beta \le \left(1 + \frac{mvr^*}{R}\right)^{\alpha},$$

where $r^* = \max(r_1, r_2, \ldots, r_n)$ is the maximum task execution requirement. For a fixed m, the performance ratio β can be arbitrarily close to 1 as $R/(vr^*) \to \infty$.

Proof. From Theorem 4, we know that the energy consumed by the n_l tasks in level l is

$$E_l = \left(\frac{A(r_{l,1}, r_{l,2}, \ldots, r_{l,n_l})}{T_l}\right)^{\alpha-1} R_l = \left(\frac{A_l}{T_l}\right)^{\alpha-1} R_l,$$

by using algorithm ES-A, where A is a list scheduling algorithm. Since the level-by-level scheduling algorithm consumes energy $E = E_1 + E_2 + \cdots + E_v$, we have

$$E = \left(\frac{A_1}{T_1}\right)^{\alpha-1} R_1 + \left(\frac{A_2}{T_2}\right)^{\alpha-1} R_2 + \cdots + \left(\frac{A_v}{T_v}\right)^{\alpha-1} R_v.$$

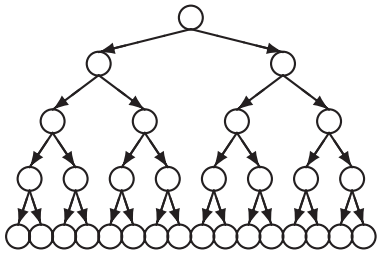

Fig. 1. CT(b,h): a complete binary tree with $b=2$ and height $h=4$.

To minimize E, we use the Lagrange multiplier system

$$\nabla E(T_1,T_2,\ldots,T_v)=\lambda\nabla F(T_1,T_2,\ldots,T_v),$$

where λ is the Lagrange multiplier, and F is the constraint $T_1+T_2+\cdots+T_v-T=0$. Since

$$\frac{\partial E}{\partial T_l}=\lambda\frac{\partial F}{\partial T_l},$$

that is,

$$A_l^{\alpha-1}R_l\left(\frac{1-\alpha}{T_l^{\alpha}}\right)=\lambda,$$

for all $1\le l\le v$, we get

$$T_l=A_l^{1-1/\alpha}R_l^{1/\alpha}\left(\frac{1-\alpha}{\lambda}\right)^{1/\alpha},$$

which implies that

$$T=(A_1^{1-1/\alpha}R_1^{1/\alpha}+A_2^{1-1/\alpha}R_2^{1/\alpha}+\cdots+A_v^{1-1/\alpha}R_v^{1/\alpha})\left(\frac{1-\alpha}{\lambda}\right)^{1/\alpha},$$

and

$$T_l=\left(\frac{A_l^{1-1/\alpha}R_l^{1/\alpha}}{A_1^{1-1/\alpha}R_1^{1/\alpha}+A_2^{1-1/\alpha}R_2^{1/\alpha}+\cdots+A_v^{1-1/\alpha}R_v^{1/\alpha}}\right)T,$$

for all $1\le l\le v$. By using the above time allocation, we have

$$\begin{aligned}E&=\sum_{l=1}^{v}\left(\frac{A_l}{T_l}\right)^{\alpha-1}R_l\\&=\sum_{l=1}^{v}\frac{A_l^{\alpha-1}R_l}{\left(\left(\frac{A_l^{1-1/\alpha}R_l^{1/\alpha}}{A_1^{1-1/\alpha}R_1^{1/\alpha}+\cdots+A_v^{1-1/\alpha}R_v^{1/\alpha}}\right)T\right)^{\alpha-1}}\\&=\frac{(A_1^{1-1/\alpha}R_1^{1/\alpha}+\cdots+A_v^{1-1/\alpha}R_v^{1/\alpha})^{\alpha-1}}{T^{\alpha-1}}\sum_{l=1}^{v}A_l^{1-1/\alpha}R_l^{1/\alpha}\\&=\frac{(A_1^{1-1/\alpha}R_1^{1/\alpha}+\cdots+A_v^{1-1/\alpha}R_v^{1/\alpha})^{\alpha}}{T^{\alpha-1}}.\end{aligned}$$

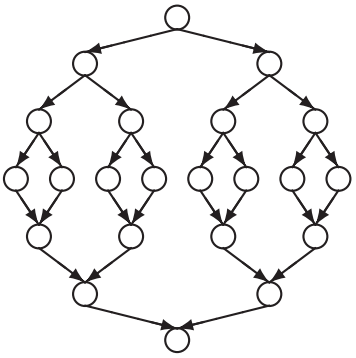

Fig. 2. PA(b,h): a partitioning algorithm with $b=2$ and $h=3$.

Similar to the proof of Theorem 9, we have

$$A_1^{1-1/\alpha}R_1^{1/\alpha}+A_2^{1-1/\alpha}R_2^{1/\alpha}+\cdots+A_v^{1-1/\alpha}R_v^{1/\alpha}\le\frac{R+mvr^*}{m^{1-1/\alpha}},$$

and

$$E\le\frac{(R+mvr^*)^{\alpha}}{m^{\alpha-1}T^{\alpha-1}}.$$

By Theorem 2, we get

$$\beta=\frac{E}{E^*}\le\left(1+\frac{mvr^*}{R}\right)^{\alpha}.$$

It is clear that for a fixed m,β can be arbitrarily close to 1 as $\mathrm{R}/(vr^*)$ becomes large. □

Theorems 4 and 10 give the power supply to the tasks in level l as

$$\left(\frac{A_l}{T_l}\right)^{\alpha}=\left(\frac{A_l(A_1^{1-1/\alpha}R_1^{1/\alpha}+A_2^{1-1/\alpha}R_2^{1/\alpha}+\cdots+A_v^{1-1/\alpha}R_v^{1/\alpha})}{A_l^{1-1/\alpha}R_l^{1/\alpha}T}\right)^{\alpha},$$

for all $1\le l\le v$.

6 Simulation Results

In this section, we present simulation data for several typical task graphs in parallel and distributed computing. The following task graphs are considered in our experiments.

1. *Tree-Structured Computations.* Many computations are tree-structured, including backtracking search, branch-and-bound computations, game-tree evaluation, functional and logical programming, and various numeric computations. For simplicity, we consider CT(b,h), i.e., complete b-ary trees of height h (see Fig. 1 with $b=2$ and $h=4$). It is easy to see that there are $v=h+1$ levels numbered as $0,1,2,\ldots,h$, and $n_l=b^l$ for $0\le l\le h$, and $n=(b^{h+1}-1)/(b-1)$.
2. *Partitioning Algorithms*. A partitioning algorithm $PA(b,h)$ represents a divide-and-conquer computation with branching factor b and height (i.e., depth of recursion) h (see Fig. 2 with $b=2$ and $h=3$). The dag of $PA(b,h)$ has $v=2h+1$ levels numbered as $0,1,2,\ldots,2h$. A partitioning algorithm proceeds in three stages. In levels $0,1,\ldots,h-1$, each task is divided into b subtasks. Then, in level h, subproblems

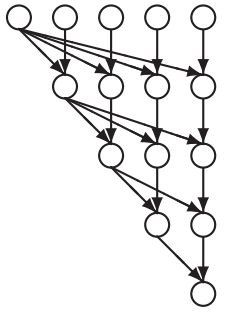

Fig. 3. LA(v): a linear algebra task graph with $v = 5$.

of small sizes are solved directly. Finally, in levels $h+1,\ h+2, \ldots, 2h$, solutions to subproblems are combined to form the solution to the original problem. Clearly, $n_l = n_{2h-l} = b^l$, for all $0 \le l \le h-1, n_h = b^h$, and $n = (b^{h+1} + b^h - 2)/(b-1)$.

3. *Linear Algebra Task Graphs.* A linear algebra task graph LA(v) with v levels (see Fig. 3 where $v = 5$) has $n_l = v - l + 1$ for $l = 1, 2, \ldots, v$, and $n = v(v+1)/2$.
4. *Diamond Dags.* A diamond dag DD(d) (see Fig. 4 with $d = 4$) contains $v = 2d - 1$ levels numbered as $1, 2, \ldots, 2d-1$. It is clear that $n_l = n_{2d-l} = l$, for all $1 \le l \le d-1, n_d = d$, and $n = d^2$.

Since each task graph has at least one parameter, we are actually dealing with classes of task graphs.

We define the *normalized schedule length* (NSL) as

$$\mathrm{NSL} = \frac{T}{\left(\frac{m}{E}\left(\frac{R}{m}\right)^{\alpha}\right)^{1/(\alpha-1)}},$$

where T is the schedule length produced by a heuristic algorithm. NSL is an upper bound for the performance ratio $\beta = T/T^*$ for the problem of minimizing schedule length with energy consumption constraint on a multiprocessor computer. When the r_i's are random variables, T, T^*, β, and NSL all become random variables. It is clear that for the problem of minimizing schedule length with energy consumption constraint, we have $\bar{\beta} \le \overline{\mathrm{NSL}}$, i.e., the expected performance ratio is no larger 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

$$\mathrm{NEC} = \frac{E}{m\left(\frac{R}{m}\right)^{\alpha}\frac{1}{T^{\alpha-1}}},$$

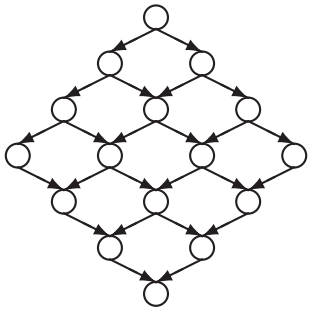

Fig. 4. DD(d): a diamond dag with $d = 4$.

TABLE 1
Simulation Data for Expected (a) NSL on CT$(2, h)$ and (b) NEC on CT$(2, h)$

h	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
5	1.8601787	1.8054790	1.6156009	1.5005729	1.4718523	1.4340418	1.6015145	1.4872121	1.5203700
6	1.4857452	1.4536065	1.2979306	1.2477428	1.2305744	1.2079401	1.3145215	1.2401244	1.2697667
7	1.2710970	1.2544910	1.1476742	1.1222740	1.1133057	1.1007698	1.1569609	1.1213663	1.1394819
8	1.1510369	1.1405473	1.0729121	1.0610503	1.0562149	1.0505149	1.0802998	1.0594901	1.0696557
9	1.0837549	1.0781178	1.0360843	1.0302434	1.0281548	1.0250168	1.0404063	1.0300509	1.0345639
10	1.0459644	1.0427650	1.0180459	1.0151433	1.0140383	1.0123639	1.0203920	1.0150787	1.0173829
11	1.0250755	1.0233005	1.0089964	1.0076459	1.0070623	1.0061795	1.0102764	1.0073152	1.0087619

(99% confidence interval = $\pm 0.704\%$)

(a)

h	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
5	3.4637906	3.2902070	2.6390688	2.2861762	2.1681872	2.0503588	2.5860302	2.2306265	2.3454536
6	2.2049866	2.1128305	1.6888755	1.5544095	1.5199808	1.4609233	1.7224255	1.5394900	1.6188107
7	1.6145514	1.5696380	1.3174445	1.2587920	1.2389101	1.2142411	1.3469180	1.2535133	1.2940118
8	1.3240845	1.3012270	1.1505403	1.1250565	1.1151975	1.1031702	1.1677682	1.1260000	1.1442418
9	1.1739183	1.1622002	1.0739406	1.0615065	1.0568706	1.0506630	1.0825660	1.0598513	1.0709854
10	1.0935561	1.0874299	1.0362926	1.0305829	1.0283473	1.0249369	1.0409395	1.0301825	1.0351943
11	1.0508143	1.0475158	1.0181097	1.0151070	1.0141015	1.0123899	1.0204272	1.0148779	1.0176548

(99% confidence interval = $\pm 1.395\%$)

(b)

where E is the energy consumed by a heuristic algorithm. NEC is an upper bound for the performance ratio $\beta = E/E^*$ for the problem of minimizing energy consumption with schedule length constraint on a multiprocessor computer. For the problem of minimizing energy consumption with schedule length constraint, we have $\bar{\beta} \le \overline{\mathrm{NEC}}$.

Notice that for a given algorithm and a given class of task graphs, the expected normalized schedule length $\overline{\mathrm{NSL}}$ and the expected normalized energy consumption $\overline{\mathrm{NEC}}$ are determined by m, n, α, and the probability distribution of the r_i's. In our simulations, the number of processors is set as $m = 10$ and the parameter α is set as 3. For convenience, the r_i's are treated as independent and identically distributed (i.i.d.) continuous random variables uniformly distributed in $[0, 1)$.

Tables 1, 2, 3, and 4, show our simulation data of the expected NSL and the expected NEC for the four classes of task graphs. For each combination of n and algorithm $A \in \{$ LL-ES-SRF, LL-ES-LS, LL-ES-LRF, LL-SRF, LL-LS, LL-LRF, ES-SRF, ES-LS, ES-LRF $\}$, we generate 1,000 sets of n 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 $\overline{\mathrm{NSL}}$ (or $\overline{\mathrm{NEC}}$). The 99 percent confidence interval of all the data in the same table is also given.

It is observed that in all cases, $\overline{\mathrm{NSL}}$ and $\overline{\mathrm{NEC}}$ (and $\bar{\beta}$ as well) quickly approach one as n increases. This is explained as follows:

A class of task graphs are called *wide task graphs* if $v/n \to 0$ as $n \to \infty$. It is easily verified that all the four classes of task graphs are wide task graphs. Since wide task graphs exhibit large parallelism due to increasing number of independent tasks as v increases, the performance of algorithm ES-A in scheduling precedence constrained tasks is dominated by the performance of algorithm ES-A in scheduling levels of independent tasks, where A is a list

TABLE 2
Simulation Data for Expected (a) NSL on $\mathrm{PA}(2,h)$ and (b) NEC on $\mathrm{PA}(2,h)$

h	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
5	1.8671892	1.8157929	1.6156395	1.6669948	1.6294073	1.5873848	1.7805405	1.6660863	1.7268876
6	1.4861053	1.4550348	1.2963406	1.3309903	1.3105776	1.2800075	1.4210073	1.3235510	1.3920778
7	1.2716364	1.2527494	1.1468311	1.1632442	1.1510080	1.1372272	1.2252668	1.1612234	1.2090985
8	1.1510778	1.1407777	1.0724776	1.0811720	1.0756815	1.0669383	1.1217568	1.0806381	1.1126049
9	1.0836392	1.0779679	1.0363098	1.0404426	1.0376562	1.0331486	1.0642748	1.0402611	1.0590541
10	1.0458885	1.0428178	1.0180318	1.0202928	1.0188258	1.0166134	1.0339421	1.0201088	1.0316610
11	1.0251128	1.0232896	1.0090334	1.0100248	1.0093307	1.0083305	1.0178333	1.0100590	1.0164186

(99% confidence interval = ±0.578%)

(a)

h	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
5	3.4884860	3.2990370	2.6272856	2.8066312	2.6701725	2.5400746	3.2033972	2.7847298	2.9917052
6	2.2085865	2.1174134	1.6897045	1.7707847	1.7171255	1.6352998	2.0307414	1.7634645	1.9262787
7	1.6172714	1.5751117	1.3175670	1.3544548	1.3274688	1.2909039	1.5067801	1.3476258	1.4628710
8	1.3249080	1.3018517	1.1495539	1.1693205	1.1570031	1.1376587	1.2601976	1.1688263	1.2381733
9	1.1736028	1.1620051	1.0739850	1.0823421	1.0768965	1.0683199	1.1336153	1.0823027	1.1221985
10	1.0942678	1.0875402	1.0364787	1.0408721	1.0379481	1.0336997	1.0684926	1.0404467	1.0633563
11	1.0507786	1.0471716	1.0180223	1.0202853	1.0187783	1.0166884	1.0357303	1.0201979	1.0332918

(99% confidence interval = ±1.224%)

(b)

TABLE 4
Simulation Data for Expected (a) NSL on DD(d) and (b) NEC on DD(d)

d	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
20	1.9235036	1.8383103	1.6554138	1.4742537	1.4195711	1.3630728	1.4478837	1.4098412	1.4286794
30	1.6085639	1.5518975	1.3208944	1.2473833	1.2124985	1.1592638	1.2027628	1.1804612	1.1976611
40	1.4534910	1.4109231	1.1959713	1.1557567	1.1301961	1.0892249	1.1160178	1.1014682	1.1160190
50	1.3622684	1.3280160	1.1334305	1.1081058	1.0886178	1.0569869	1.0762105	1.0645863	1.0773391
60	1.3008789	1.2733463	1.0972029	1.0797477	1.0647149	1.0394792	1.0543309	1.0451079	1.0561044
70	1.2578431	1.2341416	1.0743307	1.0615081	1.0495786	1.0289978	1.0405486	1.0330590	1.0432558
80	1.2252367	1.2045662	1.0589018	1.0490160	1.0393337	1.0221819	1.0318375	1.0253013	1.0339335

(99% confidence interval = ±0.243%)

(a)

d	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
20	3.7004688	3.3790013	2.7524143	2.1685689	2.0159646	1.8597687	2.0926169	1.9907932	2.0403639
30	2.5852084	2.4064370	1.7458650	1.5581552	1.4699689	1.3433066	1.4449070	1.3919814	1.4367557
40	2.1153620	1.9929982	1.4304800	1.3359774	1.2765087	1.1860861	1.2483980	1.2131208	1.2475414
50	1.8558489	1.7629848	1.2842566	1.2274252	1.1856400	1.1174252	1.1599230	1.1348046	1.1601049
60	1.6933471	1.6211249	1.2036751	1.1657221	1.1337173	1.0803371	1.1117541	1.0917767	1.1156973
70	1.5821714	1.5223023	1.1543991	1.1266359	1.1014686	1.0588174	1.0841946	1.0671195	1.0872053
80	1.5012763	1.4512074	1.1213265	1.1005402	1.0801714	1.0449060	1.0657094	1.0513518	1.0676899

(99% confidence interval = ±0.485%)

(b)

scheduling algorithm. By Theorems 3 and 4, the asymptotic performance ratio is $\beta_\infty = 1$ as $R/r^* \to \infty$.

It is clear that for a class of wide task graphs, a uniform distribution of the r_i's in $[0,1)$, and any small $\epsilon > 0$, there exists sufficiently large n^*, such that $(vr^*)/R \leq \epsilon$ with high probability (w.h.p.) if $n \geq n^*$. By Theorems 6 and 9, the performance ratio is $\beta \leq (1+m\epsilon)^{\alpha/(\alpha-1)}$ w.h.p. for the problem of minimizing schedule length with energy consumption constraint, for all LL-A and LL-ES-A, where A is a list scheduling algorithm. By Theorems 8 and 10, the performance ratio is $\beta \leq (1+m\epsilon)^{\alpha}$ w.h.p. for the problem of minimizing energy consumption with schedule length constraint, for all LL-A and LL-ES-A, where A is a list scheduling algorithm.

It is also observed that the postpower-determination algorithm LL-LRF performs better than all other algorithms.

TABLE 3
Simulation Data for Expected (a) NSL on LA(v) and (b) NEC on LA(v)

v	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
20	1.8991823	1.8182813	1.6312428	1.4533896	1.4029103	1.3489137	1.5386796	1.4233060	1.4739078
30	1.5993086	1.5444073	1.3119372	1.2425422	1.2063395	1.1545856	1.2988333	1.1890116	1.2456850
40	1.4482262	1.4062025	1.1918450	1.1525605	1.1276112	1.0867870	1.1951581	1.1081315	1.1601876
50	1.3585367	1.3250108	1.1310792	1.1062397	1.0870539	1.0558465	1.1415176	1.0694490	1.1139570
60	1.2988059	1.2709296	1.0958734	1.0786512	1.0639131	1.0388212	1.1092971	1.0482220	1.0892504
70	1.2559991	1.2324861	1.0735645	1.0607232	1.0490292	1.0285874	1.0880072	1.0355806	1.0715725
80	1.2237370	1.2035314	1.0584246	1.0485616	1.0390107	1.0219826	1.0736566	1.0274061	1.0599042

(99% confidence interval = ±0.419%)

(a)

v	LL-ES-SRF	LL-ES-LS	LL-ES-LRF	LL-SRF	LL-LS	LL-LRF	ES-SRF	ES-LS	ES-LRF
20	3.5964504	3.3073735	2.6597043	2.1137115	1.9706139	1.8133286	2.3957086	2.0193917	2.1723456
30	2.5585566	2.3794043	1.7255332	1.5414634	1.4560932	1.3321634	1.6989907	1.4165725	1.5593887
40	2.0973933	1.9781189	1.4196296	1.3285044	1.2714000	1.1817502	1.4337702	1.2278256	1.3444287
50	1.8432089	1.7566532	1.2793701	1.2240210	1.1820942	1.1145441	1.3060945	1.1437412	1.2444285
60	1.6861833	1.6149168	1.2001283	1.1632342	1.1311192	1.0789180	1.2347610	1.0984010	1.1856898
70	1.5765439	1.5192726	1.1526643	1.1255187	1.1004167	1.0578911	1.1858206	1.0727552	1.1486937
80	1.4972127	1.4487190	1.1196313	1.0995558	1.0790740	1.0442487	1.1530414	1.0552723	1.1238569

(99% confidence interval = ±0.843%)

(b)

7 Concluding Remarks

We have addressed power-aware scheduling of sequential tasks with precedence constraints on multiprocessor computers with dynamically variable voltage and speed as combinatorial optimization problems. We have investigated two problems, namely, the problem of minimizing schedule length with energy consumption constraint and the problem of minimizing energy consumption with schedule length constraint. We identified three nontrivial subproblems in our scheduling problems, i.e., precedence constraining, task scheduling, and power supplying. Such decomposition of our optimization problems into three subproblems makes design and analysis of heuristic algorithms tractable. We have proposed three types of heuristic power allocation and scheduling algorithms for precedence constrained sequential tasks with energy and time constraints, namely, prepower-determination algorithms, postpower-determination algorithms, and hybrid algorithms. We have analyzed the performance of our algorithms and compared their solutions with optimal schedules analytically. We also presented extensive simulation data and demonstrated that for wide task graphs, the performance ratios of all our heuristic algorithms approach one as the number of tasks increases. Our algorithms are applicable to energy-efficient task scheduling in general multiprocessor and multicore processor computing systems and large scale parallel computing systems, real-time multiprocessing systems and environments, and mobile computing and communication environments.

We would like to mention the following further research directions. 1) The lower bounds in Theorems 1 and 2 should be refined for precedence constrained tasks. 2) We conjecture that for any class of wide task graphs, any probability distribution of the r_i's with finite mean and variance, and any small $\epsilon > 0$, there exists sufficiently large n^*, such that $A(r_1, r_2, \ldots, r_n)/(R/m) \leq 1 + \epsilon$ w.h.p. if $n \geq n^*$, where A is a list scheduling algorithm, and hence, the performance ratio is $\beta \leq 1 + \epsilon$ w.h.p. for the problem of minimizing schedule length with energy consumption constraint and $\beta \leq (1 + \epsilon)^{\alpha - 1}$ w.h.p. for the problem of minimizing energy consumption with schedule length constraint, for all ES-A, where A is a list scheduling algorithm. A proof of the above conjecture would be of great interest. 3) We conjecture that for any class of wide task graphs, any probability distribution of the r_i's with finite mean and variance, and any small $\epsilon > 0$, there exists sufficiently large n^*, such that $(vr^*)/R \leq \epsilon$ w.h.p. if $n \geq n^*$, and hence, the performance ratio is $\beta \leq (1 + m\epsilon)^{\alpha/(\alpha-1)}$ w.h.p. for the problem of minimizing schedule length with energy consumption constraint and $\beta \leq (1 + m\epsilon)^{\alpha}$ w.h.p. for the problem of minimizing energy consumption with schedule length constraint, for all LL-A and LL-ES-A, where A is a list scheduling algorithm. A proof of the above conjecture would be of great interest.

ACKNOWLEDGMENTS

Thanks are due to the reviewers for their comments. A preliminary version of the paper was presented on the seventh Workshop on High-Performance, Power-Aware Computing, Anchorage, Alaska, May 16-20, 2011.

REFERENCES

[1] http://en.wikipedia.org/wiki/CMOS, 2012.
[2] http://en.wikipedia.org/wiki/Dynamic_voltage_scaling, 2012.
[3] http://en.wikipedia.org/wiki/Moore's_law, 2012.
[4] http://www.green500.org/, 2012.
[5] S. Albers, "Energy-Efficient Algorithms," *Comm. of the ACM,* vol. 53, no. 5, pp. 86-96, 2010.
[6] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez, "Power-Aware Scheduling for Periodic Real-Time Tasks," *IEEE Trans. Computers,* vol. 53, no. 5, pp. 584-600, May 2004.
[7] N. Bansal, T. Kimbrel, and K. Pruhs, "Dynamic Speed Scaling to Manage Energy and Temperature," *Proc. IEEE 45th Symp. Foundation of Computer Science,* pp. 520-529, 2004.
[8] J.A. Barnett, "Dynamic Task-Level Voltage Scheduling Optimizations," *IEEE Trans. Computers,* vol. 54, no. 5, pp. 508-520, May 2005.
[9] L. Benini, A. Bogliolo, and G. De Micheli, "A Survey of Design Techniques for System-Level Dynamic Power Management," *IEEE Trans. Very Large Scale Integration Systems,* vol. 8, no. 3, pp. 299-316, June 2000.
[10] D.P. Bunde, "Power-Aware Scheduling for Makespan and Flow," *Proc. 18th ACM Symp. Parallelism in Algorithms and Architectures,* pp. 190-196, 2006.
[11] H.-L. Chan, W.-T. Chan, T.-W. Lam, L.-K. Lee, K.-S. Mak, and P.W.H. Wong, "Energy Efficient Online Deadline Scheduling," *Proc. 18th ACM-SIAM Symp. Discrete Algorithms,* pp. 795-804, 2007.
[12] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power CMOS Digital Design," *IEEE J. Solid-State Circuits,* vol. 27, no. 4, pp. 473-484, Apr. 1992.
[13] S. Cho and R.G. Melhem, "On the Interplay of Parallelization, Program Performance, and Energy Consumption," *IEEE Trans. Parallel and Distributed Systems,* vol. 21, no. 3, pp. 342-353, Mar. 2010.
[14] R.L. Graham, "Bounds on Multiprocessing Timing Anomalies," *SIAM J. Applied Math.,* vol. 2, pp. 416-429, 1969.
[15] I. Hong, D. Kirovski, G. Qu, M. Potkonjak, and M.B. Srivastava, "Power Optimization of Variable-Voltage Core-Based Systems," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 18, no. 12, pp. 1702-1714, Dec. 1999.
[16] C. Im, S. Ha, and H. Kim, "Dynamic Voltage Scheduling with Buffers in Low-Power Multimedia Applications," *ACM Trans. Embedded Computing Systems,* vol. 3, no. 4, pp. 686-705, 2004.
[17] S.U. Khan and I. Ahmad, "A Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids," *IEEE Trans. Parallel and Distributed Systems,* vol. 20, no. 3, pp. 346-360, Mar. 2009.
[18] C.M. Krishna and Y.-H. Lee, "Voltage-Clock-Scaling Adaptive Scheduling Techniques for Low Power in Hard Real-Time Systems," *IEEE Trans. Computers,* vol. 52, no. 12, pp. 1586-1593, Dec. 2003.
[19] W.-C. Kwon and T. Kim, "Optimal Voltage Allocation Techniques for Dynamically Variable Voltage Processors," *ACM Trans. Embedded Computing Systems,* vol. 4, no. 1, pp. 211-230, 2005.
[20] Y.C. Lee and A.Y. Zomaya, "Energy Conscious Scheduling for Distributed Computing Systems Under Different Operating Conditions," *IEEE Trans. Parallel and Distributed Systems,* vol. 22, no. 8, pp. 1374-1381, Aug. 2011.
[21] Y.-H. Lee and C.M. Krishna, "Voltage-Clock Scaling for Low Energy Consumption in Fixed-Priority Real-Time Systems," *Real-Time Systems,* vol. 24, no. 3, pp. 303-317, 2003.
[22] K. Li, "Performance Analysis of Power-Aware Task Scheduling Algorithms on Multiprocessor Computers with Dynamic Voltage and Speed," *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 11, pp. 1484-1497, Nov. 2008.
[23] K. Li, "Energy Efficient Scheduling of Parallel Tasks on Multiprocessor Computers," *J. Supercomputing,* vol. 60, no. 2, pp. 223-247, 2012.
[24] K. Li, "Algorithms and Analysis of Energy-Efficient Scheduling of Parallel Tasks," *Handbook of Energy-Aware and Green Computing,* I. Ahmad and S. Ranka, eds., vol. 1, ch. 15, pp. 331-360, CRC Press/Taylor & Francis Group, 2012.
[25] K. Li, "Power Allocation and Task Scheduling on Multiprocessor Computers with Energy and Time Constraints," *Energy Aware Distributed Computing Systems,* A. Zomaya and Y.-C. Lee, eds., ch. 1, John Wiley & Sons, July 2012.
[26] M. Li, B.J. Liu, and F.F. Yao, "Min-Energy Voltage Allocation for Tree-Structured Tasks," *J. Combinatorial Optimization,* vol. 11, pp. 305-319, 2006.
[27] M. Li, A.C. Yao, and F.F. Yao, "Discrete and Continuous Min-Energy Schedules for Variable Voltage Processors," *Proc. Nat'l Academy of Sciences of USA,* vol. 103, no. 11, pp. 3983-3987, 2006.
[28] M. Li and F.F. Yao, "An Efficient Algorithm for Computing Optimal Discrete Voltage Schedules," *SIAM J. Computing,* vol. 35, no. 3, pp. 658-671, 2006.
[29] J.R. Lorch and A.J. Smith, "PACE: A New Approach to Dynamic Voltage Scaling," *IEEE Trans. Computers,* vol. 53, no. 7, pp. 856-869, July 2004.
[30] R.N. Mahapatra and W. Zhao, "An Energy-Efficient Slack Distribution Technique for Multimode Distributed Real-Time Embedded Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 16, no. 7, pp. 650-662, July 2005.
[31] G. Quan and X.S. Hu, "Energy Efficient DVS Schedule for Fixed-Priority Real-Time Systems," *ACM Trans. Embedded Computing Systems,* vol. 6, no. 4, article 29, 2007.
[32] C. Rusu, R. Melhem, and D. Mossé, "Maximizing the System Value While Satisfying Time and Energy Constraints," *Proc. IEEE 23rd Real-Time Systems Symp.,* pp. 256-265, 2002.
[33] D. Shin and J. Kim, "Power-Aware Scheduling of Conditional Task Graphs in Real-Time Multiprocessor Systems," *Proc. Int'l Symp. Low Power Electronics and Design,* pp. 408-413, 2003.
[34] D. Shin, J. Kim, and S. Lee, "Intra-Task Voltage Scheduling for Low-Energy Hard Real-Time Applications," *IEEE Design & Test of Computers,* vol. 18, no. 2, pp. 20-30, Mar./Apr. 2001.
[35] M.R. Stan and K. Skadron, "Guest Editors' Introduction: Power-Aware Computing," *Computer,* vol. 36, no. 12, pp. 35-38, Dec. 2003.
[36] O.S. Unsal and I. Koren, "System-Level Power-Aware Design Techniques in Real-Time Systems," *Proc. IEEE,* vol. 91, no. 7, pp. 1055-1069, July 2003.
[37] V. Venkatachalam and M. Franz, "Power Reduction Techniques for Microprocessor Systems," *ACM Computing Surveys,* vol. 37, no. 3, pp. 195-237, 2005.

[38] M. Weiser, B. Welch, A. Demers, and S. Shenker, "Scheduling for Reduced CPU Energy," *Proc. First USENIX Symp. Operating Systems Design and Implementation,* pp. 13-23, 1994.
[39] P. Yang, C. Wong, P. Marchal, F. Catthoor, D. Desmet, D. Verkest, and R. Lauwereins, "Energy-Aware Runtime Scheduling for Embedded-Multiprocessor SOCs," *IEEE Design & Test of Computers,* vol. 18, no. 5, pp. 46-58, Sept./Oct. 2001.
[40] F. Yao, A. Demers, and S. Shenker, "A Scheduling Model for Reduced CPU Energy," *Proc. IEEE 36th Symp. Foundations of Computer Science,* pp. 374-382, 1995.
[41] H.-S. Yun and J. Kim, "On Energy-Optimal Voltage Scheduling for Fixed-Priority Hard Real-Time Systems," *ACM Trans. Embedded Computing Systems,* vol. 2, no. 3, pp. 393-430, 2003.
[42] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," *Proc. 41st Design Automation Conf.,* pp. 868-873, 2004.
[43] X. Zhong and C.-Z. Xu, "Energy-Aware Modeling and Scheduling for Dynamic Voltage Scaling with Statistical Real-Time Guarantee," *IEEE Trans. Computers,* vol. 56, no. 3, pp. 358-372, Mar. 2007.
[44] D. Zhu, R. Melhem, and B.R. Childers, "Scheduling with Dynamic Voltage/Speed Adjustment Using Slack Reclamation in Multiprocessor Real-Time Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 14, no. 7, pp. 686-700, July 2003.
[45] D. Zhu, D. Mossé, and R. Melhem, "Power-Aware Scheduling for AND/OR Graphs in Real-Time Systems," *IEEE Trans. Parallel and Distributed Systems,* vol. 15, no. 9, pp. 849-864, Sept. 2004.
[46] J. Zhuo and C. Chakrabarti, "Energy-Efficient Dynamic Task Scheduling Algorithms for DVS Systems," *ACM Trans. Embedded Computing Systems,* vol. 7, no. 2, article 17, 2008.
[47] Z. Zong, A. Manzanares, X. Ruan, and X. Qin, "EAD and PEBD: Two Energy-Aware Duplication Scheduling Algorithms for Parallel Tasks on Homogeneous Clusters," *IEEE Trans. Computers,* vol. 60, no. 3, pp. 360-374, Mar. 2011.

Keqin Li is a SUNY distinguished professor of computer science in the State University of New York at New Paltz. He is also an intellectual ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China. His research interests include design and analysis of algorithms, parallel and distributed computing, and computer networking. He has contributed extensively to processor allocation and resource management; design and analysis of sequential/parallel, deterministic/probabilistic, and approximation algorithms; parallel and distributed computing systems performance analysis, prediction, and evaluation; job scheduling, task dispatching, and load balancing in heterogeneous distributed systems; dynamic tree embedding and randomized load distribution in static networks; parallel computing using optical interconnections; dynamic location management in wireless communication networks; routing and wavelength assignment in optical networks; energy-efficient computing and communication. His current research interests include lifetime maximization in sensor networks, file sharing in peer-to-peer systems, power management and performance optimization, and cloud computing. He has published more than 240 journal articles, book chapters, and research papers in refereed international conference proceedings. He has received several Best Paper Awards for his highest quality work. He has served in various capacities for numerous international conferences as general chair, program chair, workshop chair, track chair, and steering/advisory/award/program committee member. Currently, he is on the editorial board of *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *Journal of Parallel and Distributed Computing*, *International Journal of Parallel, Emergent and Distributed Systems*, *International Journal of High Performance Computing and Networking*, and *Optimization Letters*. He is a senior member of the IEEE and the IEEE Computer Society.