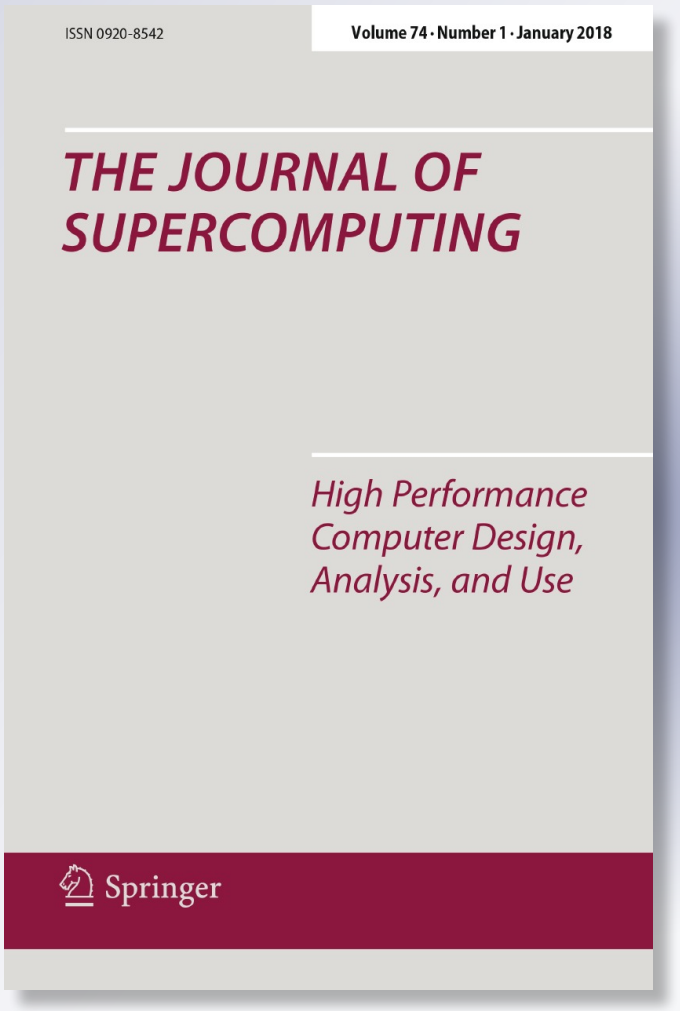
Energy constrained scheduling of stochastic tasks

Keqin Li

The Journal of Supercomputing
An International Journal of High-Performance Computer Design, Analysis, and Use

ISSN 0920-8542
Volume 74
Number 1

J Supercomput (2018) 74:485-508
DOI 10.1007/s11227-017-2137-0



Springer

J Supercomput (2018) 74:485–508
https://doi.org/10.1007/s11227-017-2137-0

Energy constrained scheduling of stochastic tasks

Keqin Li[1]

Published online: 6 September 2017

Abstract Energy-efficient scheduling of stochastic tasks is considered in this paper. The main characteristic of a stochastic task is that its execution time is a random variable whose actual value is not known in advance, but only its probability distribution. Our performance measures are the probability that the total execution time does not exceed a given bound and the probability that the total energy consumption does not exceed a given bound. Both probabilities need to be maximized. However, maximizations of the two performance measures are conflicting objectives. Our strategy is to fix one and maximize the other. Our investigation includes the following two aspects, with the purpose of maximizing the probability for the total execution time not to exceed a given bound, under the constraint that the probability for the total energy consumption not to exceed a given bound is at least certain value. First, we explore the technique of optimal processor speed setting for a given set of stochastic tasks on a processor with variable speed. It is found that the simple equal speed method (in which all tasks are executed with the same speed) yields high quality solutions. Second, we explore the technique of optimal stochastic task scheduling for a given set of stochastic tasks on a multiprocessor system, assuming that the equal speed method is used. We propose and evaluate the performance of several heuristic stochastic task scheduling algorithms. Our simulation studies identify the best methods among the proposed heuristic methods.

Keywords Energy consumption · Energy-efficient scheduling · Execution time · Heuristic algorithm · Optimization problem · Processor speed setting · Stochastic tasks

✉ Keqin Li
lik@newpaltz.edu

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

1 Introduction

1.1 Motivation

It has long been recognized that task execution times in parallel and distributed computing are unpredictable, because of variable execution paths of a parallel program due to different input data, uncertain communication delays due to network contention, and other nondeterministic costs due to various resource competition [27, p. 86]. Task execution times in cloud computing are also random due to Internet-based access of virtualized and shared resources [9, p. 291], [16], [24, Chapter 8, p. 155]. Furthermore, the random variables which represent unpredictable task execution times are usually assumed to obey normal distributions, because of their accurate approximation of random task execution times, their analytical tractability, and their closeness to other distributions such as Gamma and Beta distributions [11, p. 185], [20, p. 78], [31]. The reader is referred to [7, Chapter 3] for a gentle introduction to stochastic scheduling and [18] for a comprehensive treatment of stochastic task scheduling by using expectation-variance analysis of a schedule based on the addability of the expectations and variances of normal distributions. With the emergence of energy-aware task scheduling [29,30], energy constrained scheduling of stochastic tasks becomes theoretically interesting and practically important.

1.2 Related research

Stochastic task scheduling on multiple machines was originated in the mid 1960's. Rothkopf was the first to consider the problem of scheduling immediately available tasks with random variable service times [17]. Since then, numerous researchers have contributed to this field, and there is a large body of literature. In [15], Möhring et al. considered the problem to minimize the total weighted completion time of a set of jobs with individual release dates which have to be scheduled on identical parallel machines. In [19], Scharbrodt et al. presented a new average case analysis for the problem of scheduling jobs on multiple machines so that the sum of job completion times is minimized. In [25], Weiss considered scheduling a batch of jobs with stochastic processing times on parallel machines, with minimization of expected weighted flowtime as objective.

Stochastic task scheduling has been extended to various scheduling models. In [1], Ahmadizar et al. dealt with a stochastic group shop scheduling problem with the objective to find a job schedule which minimizes the expected makespan. In [10], Gu et al. proposed a novel parallel quantum genetic algorithm for the stochastic job shop scheduling problem with the objective of minimizing the expected value of makespan, where the processing times are subjected to independent normal distributions. In [14], Megow et al. incorporated both stochastic scheduling and online scheduling, and considered non-preemptive parallel machine scheduling, with the objective to minimize the total weighted completion times of jobs. In [26], Weiss and Pinedo considered preemptive scheduling of tasks on multiple processors with different speeds, where tasks require amounts of work which are exponentially distributed with different parameters.

Stochastic task scheduling has been investigated for precedence constrained tasks. In [2], Ando et al. presented a linear time algorithm for approximating the longest path length of a given directed acyclic graph (DAG), where each edge length is given as a normally distributed random variable. In [4], Canon and Jeannot studied the evaluation of the efficiency and the robustness of schedules of parallel applications consisting of stochastic tasks having precedence constraints. In [21], Skutella and Uetz considered parallel and identical machine scheduling problems, where the jobs are subject to precedence constraints and release dates, and where the processing times of jobs are governed by independent probability distributions. In [23], Tongsima et al. studied scheduling algorithms taking into account the probabilistic execution times, where tasks are represented as a data-flow graph.

Stochastic task scheduling has been explored for various kinds of parallel and distributed computing environments. In [6], Chen et al. considered both stochastic and robust scheduling on unrelated machines with objectives of minimizing the sum of weighted completion times and the makespan by capturing resource provisioning and job scheduling in the cloud. In [8], Dong et al. proposed a mechanism used to estimate the probability distribution of task execution time based on resource load and introduced a resource load-based stochastic scheduling algorithm for grid environments. In [13], Li et al. presented a model of scheduling precedence constrained stochastic tasks with communication times on a heterogeneous cluster system with processors of different computing capabilities to minimize a parallel application's expected completion time. In [22], Tang et al. addressed the problem of scheduling precedence constrained tasks of a parallel application with random processing times and communication times on a grid computing system so as to minimize the makespan.

However, there has been only a little research on energy-efficient stochastic task scheduling. In [12], Li et al. studied the problem of scheduling a bag-of-tasks application, made of a collection of independent stochastic tasks with normal distributions of task execution times, on a heterogeneous platform with deadline and energy consumption budget constraints. The main approach in [12] is to minimize a weighted sum of the probability that the makespan does not exceed certain deadline and the probability that the energy consumption does not exceed a given budget. The main concern of this method is that the two probabilities are different in nature and it makes little sense to consider a weighted sum.

1.3 New contributions

In this paper, we consider energy-efficient scheduling of stochastic tasks. The main characteristic of a stochastic task is that its execution time is a random variable whose actual value is not known in advance, but only its probability distribution. Our performance measures are the probability that the total execution time does not exceed a given bound and the probability that the total energy consumption does not exceed a given bound. Both probabilities need to be maximized. However, maximizations of the two performance measures are conflicting objectives, and simultaneous bi-objective optimization is impossible. Our strategy is to fix one and maximize the other, and

thus, we take a different approach from [12]. To the best of the author's knowledge, the optimization problem proposed in this paper has never been studied before.

Our investigation and contributions include the following two aspects, with the purpose of maximizing the probability that the total execution time does not exceed a given bound, under the constraint that the probability that the total energy consumption does not exceed a given bound is at least certain value.

- First, we explore the technique of optimal processor speed setting for a given set of stochastic tasks on a processor with variable speed. It is found that the simple equal speed method (in which all tasks are executed with the same speed) yields high quality (i.e., near-optimality) solutions.
- Second, we explore the technique of optimal stochastic task scheduling for a given set of stochastic tasks on a multiprocessor system, assuming that the equal speed method is used. We propose and evaluate the performance of several heuristic stochastic task scheduling algorithms. Our simulation studies identify the best methods among the proposed heuristic methods.

The rest of the paper is organized as follows. In Sect. 2, we describe the speed and power model and the stochastic task model and define our energy constrained stochastic task scheduling problem. In Sect. 3, we consider optimal processor speed setting on a uniprocessor system. In Sect. 4, we consider optimal stochastic task scheduling on a multiprocessor system. In Sect. 5, we conclude the paper.

2 Preliminaries

2.1 The speed and power model

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 [5]. 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 $\phi > 0$ [28]. 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 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 $\alpha = 2\phi + 1$.

Assume that we are given n independent sequential tasks to be executed on m identical processors. There is no communication nor precedence constraint among the tasks. Let r_i represent the execution requirement (measured in the number of processor cycles or the number of instructions) of task i, where $1 \le i \le n$. 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). Since ξ is a constant which only creates scaling effect, for

ease of discussion, we will assume that $\xi = 1$. Hence, the power required to execute task i is $p_i = s_i^\alpha$. 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 $e_i = p_i t_i = r_i s_i^{\alpha-1}$.

2.2 The stochastic task model

Throughout the paper, we use $f_X(x)$ to represent the probability density function (pdf) of a random variable X, and $F_X(x)$ to represent the cumulative distribution function (cdf) of a random variable X. The mean and variance of a random variable X are represented by μ_X and σ_X^2, respectively.

A normal random variable X has pdf

$$f_X(x) = \frac{1}{\sqrt{2\pi}\sigma_X} \mathrm{e}^{-(x-\mu_X)^2/2\sigma_X^2}.$$

It is well known that $Y = (X - \mu_X)/\sigma_X$ is a standard normal random variable with mean 0 and variance 1. The pdf of Y is

$$f_Y(y) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-y^2/2}.$$

The cdf of Y is

$$F_Y(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-y^2/2} \mathrm{d}y.$$

For any normal random variable X and any z, we have

$$F_X(z) = F_Y\left(\frac{z - \mu_X}{\sigma_X}\right).$$

Although all random variables in this paper are nonnegative, for convenience, we still assume that they are regular normal random variables in the range $(-\infty, +\infty)$, not truncated normal random variables in the range $(0, +\infty)$, with the understanding that the distribution in the range $(-\infty, 0)$ is extremely small and negligible.

There are n stochastic tasks which are elaborated as follows.

For task i, where $1 \le i \le n$, the execution requirement r_i of the task is a normal random variable with mean $\mu_i = \mu_{r_i}$ and variance $\sigma_i^2 = \sigma_{r_i}^2$. The execution speed of the processor for task i is s_i. Hence, the execution time of task i is r_i/s_i, which is a normal random variable with mean μ_i/s_i and variance σ_i^2/s_i^2. The energy consumed by task i is $r_i s_i^{\alpha-1}$, which is a normal random variable with mean $\mu_i s_i^{\alpha-1}$ and variance $\sigma_i^2 s_i^{2(\alpha-1)}$.

2.3 The optimization problem

In this paper, we consider scheduling stochastic tasks on a multiprocessor system with m identical processors. A schedule of a set S of n tasks on m processors is actually a partition $(S_1, S_2, \ldots, S_m)$ of the n tasks into m disjoint subsets $S_1, S_2, \ldots, S_m$, such that $S_1 \cup S_2 \cup \cdots \cup S_m = S$.

Given a schedule $(S_1, S_2, \ldots, S_m)$, we can assign a processor speed s_i to each task i. Let T_j denote the total execution time of the jth processor, where $1 \leq j \leq m$. Since the execution time of task i is r_i/s_i, we have

$$T_j = \sum_{i \in S_j} \frac{r_i}{s_i},$$

which is a normal random variable. The total execution time of the n tasks is

$$T = \max\{T_1, T_2, \ldots, T_m\}.$$

Unfortunately, T does not have a normal distribution. Its cdf is

$$F_T(x) = \prod_{j=1}^{m} F_{T_j}(x),$$

and its pdf can be obtained as follows,

$$f_T(x) = \sum_{j=1}^{m} f_{T_j}(x) \prod_{j' \neq j} F_{T_{j'}}(x).$$

The total energy consumption, i.e.,

$$E = r_1 s_1^{\alpha-1} + r_2 s_2^{\alpha-1} + \cdots + r_n s_n^{\alpha-1},$$

is a normal random variable.

Our problem of energy constrained scheduling of stochastic tasks can be formally defined as the following combinatorial optimization problem. Given m processors and n stochastic tasks with $\mu_1, \mu_2, \ldots, \mu_n, \sigma_1, \sigma_2, \ldots, \sigma_n$, a total execution time bound T^*, a total energy consumption bound E^*, and β (where $0 < \beta < 1$), our optimization problem is to find a stochastic task schedule $(S_1, S_2, \ldots, S_m)$ of the n tasks on the m processors, and a processor speed setting $(s_1, s_2, \ldots, s_n)$, such that $F_T(T^*) = P(T \leq T^*)$ is maximized, under the condition that $F_E(E^*) = P(E \leq E^*) = \beta$. The optimization problem is defined in such a way that the probability for the total execution time T not to exceed a given bound T^* is maximized, under the constraint that the probability for the total energy consumption E not to exceed a given bound E^* is at least β.

2.4 The road map

It is clear that our optimization problem contains two subproblems, i.e., task scheduling [finding an optimal stochastic task schedule $(S_1, S_2, \ldots, S_m)$] and speed setting [finding an optimal processor speed setting $(s_1, s_2, \ldots, s_n)$]. Attempting to solve both subproblems at the same time seems difficult. Our approach to solving the optimization problem is the following.

First, in Sect. 3, we consider the subproblem of optimal processor speed setting. We explore the technique of optimal processor speed setting for a given set of stochastic tasks on a processor with variable speed, to maximize the probability for the total execution time not to exceed a given bound, under the constraint that the probability for the total energy consumption not to exceed a given bound is at least certain value. In fact, we consider the special case of a uniprocessor system. Surprisingly, even for this special case, the optimal processor speed setting problem is still too complicated to accommodate an effective method to find an optimal solution. However, it is found that the equal speed method yields high quality solutions.

Second, in Sect. 4, we consider the subproblem of optimal stochastic task scheduling. We explore the technique of optimal stochastic task scheduling for a given set of stochastic tasks on a multiprocessor system to maximize the probability for the total execution time not to exceed a given bound, under the constraint that the probability for the total energy consumption not to exceed a given bound is at least certain value, assuming that a simple power allocation method is used, i.e., the equal speed method, in which, all tasks are executed with the same speed s. Thus, we can focus on heuristic stochastic task scheduling algorithms on a multiprocessor system. We evaluate the performance of several heuristic stochastic task scheduling algorithms. Our simulation studies identify the best methods among the proposed heuristic methods.

3 Optimal processor speed setting

In this section, we consider optimal processor speed setting on a uniprocessor system. First, we define our optimization problem on a uniprocessor. Next, we present our method and a numerical algorithm to solve the optimization problem. Finally, we demonstrate numerical examples.

3.1 Problem definition

Consider n tasks executed on a single processor.

The total execution time of the n tasks is

$$T = \frac{r_1}{s_1} + \frac{r_2}{s_2} + \cdots + \frac{r_n}{s_n}.$$

T is a normal random variable with mean

$$\mu_T = \frac{\mu_1}{s_1} + \frac{\mu_2}{s_2} + \cdots + \frac{\mu_n}{s_n},$$

and variance

$$\sigma_T^2 = \frac{\sigma_1^2}{s_1^2} + \frac{\sigma_2^2}{s_2^2} + \cdots + \frac{\sigma_n^2}{s_n^2}.$$

The total energy consumption of the n tasks is

$$E = r_1 s_1^{\alpha-1} + r_2 s_2^{\alpha-1} + \cdots + r_n s_n^{\alpha-1}.$$

E is a normal random variable with mean

$$\mu_E = \mu_1 s_1^{\alpha-1} + \mu_2 s_2^{\alpha-1} + \cdots + \mu_n s_n^{\alpha-1},$$

and variance

$$\sigma_E^2 = \sigma_1^2 s_1^{2(\alpha-1)} + \sigma_2^2 s_2^{2(\alpha-1)} + \cdots + \sigma_n^2 s_n^{2(\alpha-1)}.$$

The pdf of T is

$$f_T(x) = \frac{1}{\sqrt{2\pi}\sigma_T} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2}.$$

The cdf of T is

$$F_T(T^*) = P(T \le T^*) = \int_{-\infty}^{T^*} f_T(x)\mathrm{d}x.$$

The pdf of E is

$$f_E(x) = \frac{1}{\sqrt{2\pi}\sigma_E} \mathrm{e}^{-(x-\mu_E)^2/2\sigma_E^2}.$$

The cdf of E is

$$F_E(E^*) = P(E \le E^*) = \int_{-\infty}^{E^*} f_E(x)\mathrm{d}x.$$

Notice that both $F_T(T^*)$ and $F_E(E^*)$ can be viewed as functions of $s_1, s_2, \ldots, s_n$, i.e., $F_T(T^*) = F_T(s_1, s_2, \ldots, s_n, T^*)$, and $F_E(E^*) = F_E(s_1, s_2, \ldots, s_n, E^*)$.

We are now ready to define our problem of optimal processor speed setting. Given n stochastic tasks with $\mu_1, \mu_2, \ldots, \mu_n, \sigma_1, \sigma_2, \ldots, \sigma_n$, a total execution time bound T^*, a total energy consumption bound E^*, and β (where $0 < \beta < 1$), our optimization problem is to find an optimal processor speed setting $(s_1, s_2, \ldots, s_n)$, in the sense that $F_T(s_1, s_2, \ldots, s_n, T^*)$ is maximized, under the condition that $F_E(s_1, s_2, \ldots, s_n, E^*) = \beta$.

3.2 The method

We are facing a mutli-variable optimization problem. We can maximize $F_T(s_1, s_2, \ldots, s_n, T^*)$ by using the method of Lagrange multiplier, namely

$$\nabla F_T(s_1, s_2, \ldots, s_n, T^*) = \phi \nabla F_E(s_1, s_2, \ldots, s_n, E^*),$$

that is,

$$\frac{\partial F_T(s_1, s_2, \ldots, s_n, T^*)}{\partial s_i} = \phi \frac{\partial F_E(s_1, s_2, \ldots, s_n, E^*)}{\partial s_i},$$

for all $1 \leq i \leq n$, where ϕ is a Lagrange multiplier.

It can be shown that

$$\frac{\partial F_T(s_1, s_2, \ldots, s_n, T^*)}{\partial s_i} = f_T(T^*) \left(\frac{\mu_i}{s_i^2} + \frac{1}{\sigma_T} \left(\frac{T^* - \mu_T}{\sigma_T} \right) \frac{\sigma_i^2}{s_i^3} \right),$$

and

$$\frac{\partial F_E(s_1, s_2, \ldots, s_n, E^*)}{\partial s_i} = -(\alpha - 1) f_E(E^*) \left(\mu_i s_i^{\alpha-2} + 2 \left(\frac{E^* - \mu_E}{\sigma_E} \right) \sigma_i^2 s_i^{2\alpha-3} \right).$$

The derivation of the above two equations is given in "Appendix 1".

Therefore, we need to solve the following equation, i.e.,

$$\begin{aligned} & f_T(T^*) \left(\frac{\mu_i}{s_i^2} + \frac{1}{\sigma_T} \left(\frac{T^* - \mu_T}{\sigma_T} \right) \frac{\sigma_i^2}{s_i^3} \right) \\ & = -\phi(\alpha - 1) f_E(E^*) \left(\mu_i s_i^{\alpha-2} + 2 \left(\frac{E^* - \mu_E}{\sigma_E} \right) \sigma_i^2 s_i^{2\alpha-3} \right), \end{aligned}$$

or equivalently,

$$\begin{aligned} F_i = & f_T(T^*) \left(\frac{\mu_i}{s_i^2} + \sigma_i^2 \left(\frac{T^* - \mu_T}{\sigma_T^2 s_i^3} \right) \right) \\ & + \phi(\alpha - 1) f_E(E^*) \left(\mu_i s_i^{\alpha-2} + 2\sigma_i^2 s_i^{2\alpha-3} \left(\frac{E^* - \mu_E}{\sigma_E} \right) \right) = 0, \end{aligned}$$

for all $1 \leq i \leq n$. The above equation, together with

$$F_0 = F_E(s_1, s_2, \ldots, s_n, E^*) - \beta = 0,$$

constitute a nonlinear system of $n + 1$ equations with $n + 1$ unknowns, i.e., $s_1, s_2, \ldots, s_n$, and ϕ.

An analytical solution to the above equations is impossible. We will seek numerical solutions.

3.3 A numerical algorithm

We have a nonlinear system of equations, i.e.,

$$\begin{aligned} F_0(\phi, s_1, \ldots, s_n) &= 0, \\ F_1(\phi, s_1, \ldots, s_n) &= 0, \\ &\vdots \\ F_n(\phi, s_1, \ldots, s_n) &= 0. \end{aligned}$$

By using vector notation to represent the variables $\phi, s_1, \ldots, s_n$, we write

$$\mathbf{y} = (y_0, y_1, \ldots, y_n) = (\phi, s_1, \ldots, s_n),$$

and $F_i(\phi, s_1, \ldots, s_n) = F_i(y_0, y_1, \ldots, y_n) = F_i(\mathbf{y})$, where $F_i : \mathbb{R}^{n+1} \to \mathbb{R}$ maps $(n+1)$-dimensional space $\mathbb{R}^{n+1}$ into the real line $\mathbb{R}$. By defining a function $\mathbf{F} : \mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$ which maps $\mathbb{R}^{n+1}$ into $\mathbb{R}^{n+1}$,

$$\mathbf{F}(\mathbf{y}) = (F_0(y_0, y_1, \ldots, y_n), F_1(y_0, y_1, \ldots, y_n), \ldots, F_n(y_0, y_1, \ldots, y_n)),$$

namely,

$$\mathbf{F}(\mathbf{y}) = (F_0(\mathbf{y}), F_1(\mathbf{y}), \ldots, F_n(\mathbf{y})),$$

then our nonlinear system of equations is

$$\mathbf{F}(\mathbf{y}) = \mathbf{0},$$

where $\mathbf{0} = (0, 0, \ldots, 0)$.

The above nonlinear system of equations can be solved by using Newton's method. To this end, we need the Jacobian matrix $J(\mathbf{y})$ defined as

$$J(\mathbf{y}) = \begin{bmatrix} \frac{\partial F_0(\mathbf{y})}{\partial y_0} & \frac{\partial F_0(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_0(\mathbf{y})}{\partial y_n} \\ \frac{\partial F_1(\mathbf{y})}{\partial y_0} & \frac{\partial F_1(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_1(\mathbf{y})}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_n(\mathbf{y})}{\partial y_0} & \frac{\partial F_n(\mathbf{y})}{\partial y_1} & \cdots & \frac{\partial F_n(\mathbf{y})}{\partial y_n} \end{bmatrix}.$$

The various components of the above matrix are calculated in "Appendix 2".

Our numerical algorithm for finding an optimal processor speed setting $(s_1, \ldots, s_n)$ and the Lagrange multiplier ϕ, i.e., the vector $\mathbf{y} = (\phi, s_1, \ldots, s_n)$ which satisfies the

nonlinear system of equations $\mathbf{F}(\mathbf{y}) = \mathbf{0}$, is given in Algorithm 1. This is essentially the standard Newton's iterative method [3, p. 451]. Our initial approximation of $\mathbf{y}$ is $\phi = -1$ and $s_i = s$ for all $1 \leq i \leq n$ [line (1)], where s is the constant speed of the processor, which satisfies

$$F_E(s, s, \ldots, s, E^*) = \beta.$$

Let

$$\mu = \sum_{i=1}^{n} \mu_i,$$

and

$$\sigma^2 = \sum_{i=1}^{n} \sigma_i^2.$$

Then, we have $\mu_E = \mu s^{\alpha-1}$ and $\sigma_E^2 = \sigma^2 s^{2(\alpha-1)} = (\sigma s^{\alpha-1})^2$ and

$$f_E(x) = \frac{1}{\sqrt{2\pi}\sigma s^{\alpha-1}} \mathrm{e}^{-(x-\mu s^{\alpha-1})^2/2(\sigma s^{\alpha-1})^2}.$$

Hence s is the unique value which satisfies

$$F_E(s, s, \ldots, s, E^*) = F_Y\left(\frac{E^* - \mu s^{\alpha-1}}{\sigma s^{\alpha-1}}\right) = \beta,$$

where we notice that $F_E(s, s, \ldots, s, E^*)$ is a decreasing function of s. Since $E^* - \mu s^{\alpha-1} \geq 0$, we have $s \leq (E^*/\mu)^{1/(\alpha-1)}$. Thus, s can be found by using the classic bisection method [3, p. 21] in the interval $[0, (E^*/\mu)^{1/(\alpha-1)}]$. The value of $\mathbf{y}$ is then repeatedly modified as $\mathbf{y} + \mathbf{z}$ [line (6)], where $\mathbf{z}$ is the solution to the linear system of equations $J(\mathbf{y})\mathbf{z} = -\mathbf{F}(\mathbf{y})$ [line (5)]. Such modification is repeated until $\|\mathbf{z}\| \leq \varepsilon$ [line (7)], where

$$\|\mathbf{z}\| = \sqrt{z_0^2 + z_1^2 + \cdots + z_n^2},$$

and ε is a sufficiently small constant, say, 10^{-10}. The linear system of equations in line (5) can be solved by using the classic Gaussian elimination with backward substitution algorithm [3, pp. 268–269].

Although there is no analytical result on the time complexity of the algorithm, our experiments reveal that the algorithm is reasonably fast.

Algorithm 1: A numerical algorithm for finding an optimal processor speed setting.

Algorithm: Optimal Processor Speed Setting
Input: Parameters $\mu_1, \mu_2, \ldots, \mu_n, \sigma_1, \sigma_2, \ldots, \sigma_n, T^*, E^*$, and β.
Output: An optimal processor speed setting and ϕ, i.e., $\mathbf{y} = (\phi, s_1, \ldots, s_n)$, which satisfies $\mathbf{F}(\mathbf{y}) = \mathbf{0}$.

$\mathbf{y} \leftarrow (-1, s, \ldots, s)$; (1)
repeat (2)
  Calculate $J(\mathbf{y})$, where $J(\mathbf{y})_{i,j} = \partial F_i(\mathbf{y})/\partial y_j$ for $0 \le i, j \le n$; (3)
  Calculate $\mathbf{F}(\mathbf{y}) = (F_0(\mathbf{y}), F_1(\mathbf{y}), \ldots, F_n(\mathbf{y}))$; (4)
  Solve the linear system of equations $J(\mathbf{y})\mathbf{z} = -\mathbf{F}(\mathbf{y})$; (5)
  $\mathbf{y} \leftarrow \mathbf{y} + \mathbf{z}$; (6)
until $\|\mathbf{z}\| \le \varepsilon$. (7)

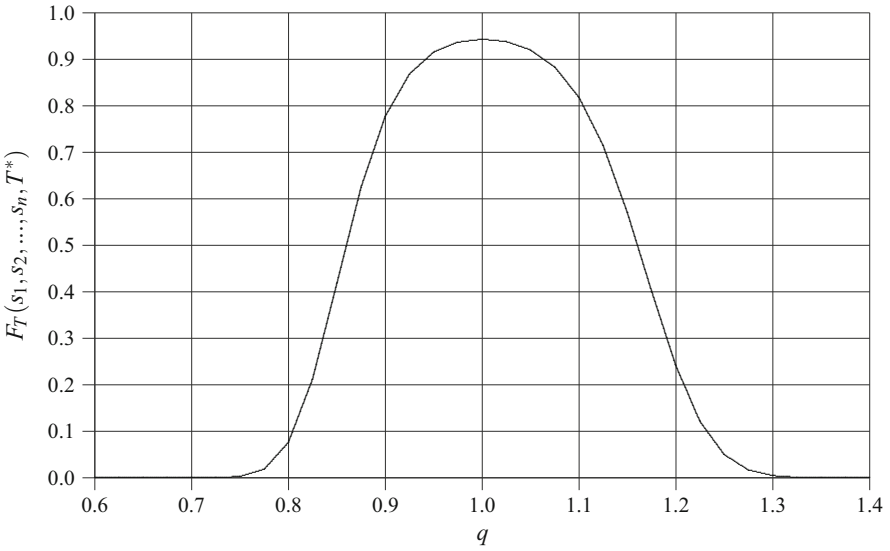


Fig. 1 $F_T(s_1, s_2, \ldots, s_n, T^*)$ versus q

3.4 Numerical examples

Let us consider $n = 7$ tasks with parameters $\mu_i = 5.0 + 2.0(i - 1)$ and $\sigma_i = 1.0 + 0.4(i - 1)$, for all $1 \le i \le n$. We set $T^* = 80$, $E^* = 100$, and $\beta = 0.9$. We set $\alpha = 3$ in all the examples in this paper. (All values are chosen for illustrative purpose only.)

We consider the following processor speed setting, $(s_1, s_1 q, s_1 q^2, \ldots, s_1 q^{n-1})$, i.e., $s_i = s_1 q^{i-1}$, for all $1 \le i \le n$, where s_1 is determined in such a way that $F_E(s_1, s_2, \ldots, s_n, E^*) = \beta$. In Fig. 1, we show $F_T(s_1, s_2, \ldots, s_n, T^*)$ as a function of q, where $0.6 \le q \le 1.4$. It is seen that different processor speed settings do result in significantly different quality of stochastic task scheduling. Hence, finding the optimal processor speed setting which maximizes $F_T(s_1, s_2, \ldots, s_n, T^*)$ is indeed an important problem.

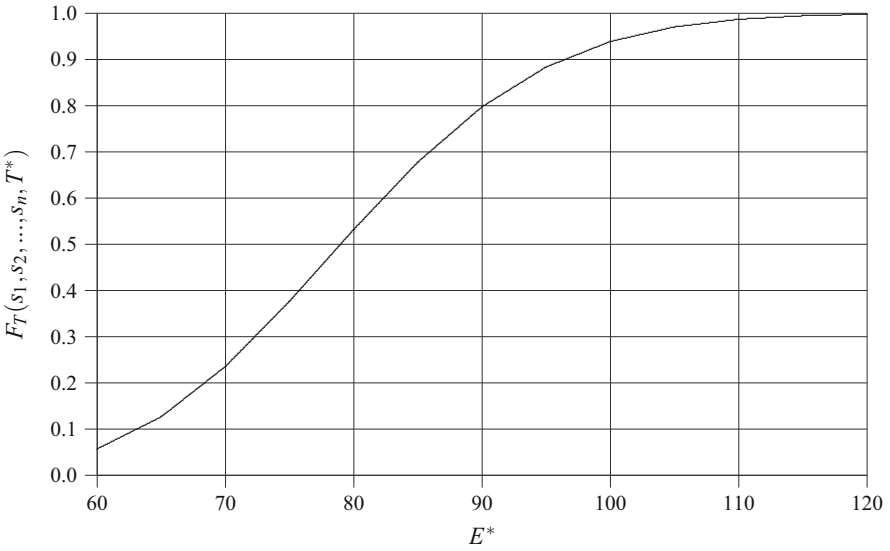


Fig. 2 $F_T(s_1, s_2, \ldots, s_n, T^*)$ versus E^*

Using our algorithm in the last section, the optimal processor speed setting found by Algorithm 1 is

$$\begin{aligned}&(s_1, s_2, s_3, s_4, s_5, s_6, s_7)\\ &\quad = (1.1889282, 1.1490253, 1.1183048, 1.0936121, 1.0731642,\\ &\qquad 1.0558559, 1.0409555),\end{aligned}$$

which gives rise to $F_T(s_1, s_2, \ldots, s_n, T^*) = 0.9390434$.

In Fig. 2, we show $F_T(s_1, s_2, \ldots, s_n, T^*)$ as a function of E^*, where $60 \leq E^* \leq 120$. The $(s_1, s_2, s_3, s_4, s_5, s_6, s_7)$ is obtained by using Algorithm 1. It is observed that as more and more energy resource is provided, the quality of stochastic task scheduling, i.e., the probability that a given time deadline is satisfied, increases.

Unfortunately, due to the sophistication of $F_T(s_1, s_2, \ldots, s_n, T^*)$ and $F_E(s_1, s_2, \ldots, s_n, E^*)$, Newton's iterative method exhibits two weaknesses. First, it is very sensitive to the initial approximation and does not always converge to a solution. Second, even though it converges, it does not yield an optimal solution. Surprisingly, the simple equal speed method, in which, all tasks are executed with the same speed, may perform better than Newton's iterative method. For the above example, when $s_i = 1.0850523$, for all $1 \leq i \leq n$, we get $F_T(s_1, s_2, \ldots, s_n, T^*) = 0.9432856$, which is greater than that of Newton's iterative method.

Of course, the equal speed method is not optimal either. Let us consider only the first two tasks in the above example. Newton's iterative method gives

$$(s_1, s_2) = (1.2025708, 1.1755580),$$

and $F_T(s_1, s_2, T^*) = 0.9498905$. The equal speed method gives

$$(s_1, s_2) = (1.1865784, 1.1865784),$$

and $F_T(s_1, s_2, T^*) = 0.9501387$. It is clear that the constraint $F_E(s_1, s_2, E^*) = \beta$ makes s_2 as a function of s_1, i.e., $s_2 = g(s_1)$ for some g. Therefore, $F_E(s_1, s_2, E^*) = F_E(s_1, g(s_1), E^*)$ becomes a function of s_1. It is observed that in a small interval $[1.180, 1.188]$ of s_1, $F_E(s_1, g(s_1), E^*)$ increases as s_1 increases and then decreases as s_1 further increases. When

$$(s_1, s_2) = (1.1840000, 1.1883252),$$

we have $F_T(s_1, s_2, T^*) = 0.9501429$, which is greater than that of the equal speed method. Due to the unavailability of g analytically, there is no way to obtain the optimal solution based on $\partial F_E(s_1, g(s_1), E^*)/\partial s_1 = 0$ and the fact that it is a decreasing function of s_1 so that the bisection method is applicable.

Notice that on a multiprocessor system, both of the two probabilities $F_T(s_1, s_2, \ldots, s_n, T^*)$ and $F_E(s_1, s_2, \ldots, s_n, E^*)$ depend on a task schedule $(S_1, S_2, \ldots, S_m)$ and a speed setting $(s_1, s_2, \ldots, s_n)$. It is clear that given $\mu_1, \mu_2, \ldots, \mu_n, \sigma_1, \sigma_2, \ldots, \sigma_n$, a total execution time bound T^*, a total energy consumption bound E^*, β $(0 < \beta < 1)$, and a schedule $(S_1, S_2, \ldots, S_m)$, there is an optimal processor speed setting $(s_1, s_2, \ldots, s_n)$, such that $F_T(s_1, s_2, \ldots, s_n, T^*)$ is maximized, and that $F_E(s_1, s_2, \ldots, s_n, E^*) = \beta$. Although theoretically, a processor speed setting can be calculated by using the method developed in this section, the computational procedure is excessively complicated.

4 Optimal stochastic task scheduling

In this section, we consider optimal stochastic task scheduling on a multiprocessor system, First, we define our optimization problem. Next, we present heuristic stochastic task scheduling algorithms on a multiprocessor. Finally, we evaluate the performance of the heuristic methods by simulations.

4.1 Problem definition

It is conceivable that the problem of finding an optimal stochastic task schedule $(S_1, S_2, \ldots, S_m)$ and an optimal processor speed setting $(s_1, s_2, \ldots, s_n)$ is extremely challenging. Here, we have two subproblems simultaneously, namely task scheduling (i.e., to find a stochastic task schedule $(S_1, S_2, \ldots, S_m)$), and power allocation (i.e., to find a processor speed setting $(s_1, s_2, \ldots, s_n)$). However, since the equal speed method yields high quality (i.e., near-optimality) processor speed setting, we can focus on task scheduling by assuming that all tasks are executed with the same speed.

Hence, our problem of energy constrained scheduling of stochastic tasks becomes a pure optimal stochastic task scheduling problem. Given n stochastic tasks with $\mu_1, \mu_2, \ldots, \mu_n, \sigma_1, \sigma_2, \ldots, \sigma_n$, a total execution time bound T^*, a total energy consumption bound E^*, and β (where $0 < \beta < 1$), our optimization problem is to find an optimal stochastic task schedule $(S_1, S_2, \ldots, S_m)$, in the sense that $F_T(s_1, s_2, \ldots, s_n, T^*)$ is maximized, under the condition that $F_E(s_1, s_2, \ldots, s_n, E^*) = \beta$.

4.2 Heuristic scheduling algorithms

In this section, we consider a simple power allocation method, i.e., the equal speed method, in which, all tasks are executed with the same speed s, where s is the unique value which satisfies

$$F_E(s, s, \ldots, s, E^*) = F_Y\left(\frac{E^* - \mu s^{\alpha-1}}{\sigma s^{\alpha-1}}\right) = \beta.$$

For such a simple power allocation method, the problem of finding an optimal schedule is still NP-hard, even though all random execution requirements are deterministic values (i.e., $\sigma_i \to 0$). In this case, maximizing $F_T(s, s, \ldots, s, T^*)$ is equivalent to determine whether n tasks with execution times $\mu_1/s, \mu_2/s, \ldots, \mu_n/s$, where $s = (E^*/\mu)^{1/(\alpha-1)}$, can be completed by the deadline T^*.

We evaluate the performance of several heuristic task scheduling algorithms. Let

$$\mu^{(j)} = \sum_{i \in S_j} \mu_i,$$

and

$$(\sigma^{(j)})^2 = \sum_{i \in S_j} \sigma_i^2.$$

Then, we have $\mu_{T_j} = \mu^{(j)}/s$ and $\sigma_{T_j}^2 = (\sigma^{(j)})^2/s^2 = (\sigma^{(j)}/s)^2$. Furthermore, we get

$$f_{T_j}(x) = \frac{1}{\sqrt{2\pi}\sigma^{(j)}/s} \mathrm{e}^{-(x-\mu^{(j)}/s)^2/2(\sigma^{(j)}/s)^2},$$

and

$$F_T(s, s, \ldots, s, T^*) = \prod_{j=1}^{m} F_Y\left(\frac{T^* - \mu^{(j)}/s}{\sigma^{(j)}/s}\right).$$

By using the equal speed method, $F_T(s, s, \ldots, s, T^*)$ only depends on a schedule. Our goal is to compare $F_T(s, s, \ldots, s, T^*)$ produced by different task scheduling algorithms.

Algorithm 2: A heuristic algorithm for optimal stochastic task scheduling.

Algorithm: Optimal Stochastic Task Scheduling
Input: A set of n stochastic tasks with parameters $\mu_1, \mu_2, \ldots, \mu_n$, and $\sigma_1, \sigma_2, \ldots, \sigma_n$, m identical processors, T^*, E^*, β, and a list L of the n tasks.
Output: A schedule $(S_1, S_2, \ldots, S_m)$ such that $F_T(s, s, \ldots, s, T^*)$ is as high as possible.

$s \leftarrow$ the unique value which satisfies $F_E(s, s, \ldots, s, E^*) = \beta$; (1)
for $(j \leftarrow 1; j \leq m; j{+}{+})$ **do** (2)
  $S_j \leftarrow \emptyset$; (3)
  $\mu^{(j)} \leftarrow 0$; (4)
  $(\sigma^{(j)})^2 \leftarrow 0$; (5)
  $z_j \leftarrow \infty$; (6)
end for; (7)
for $(k \leftarrow 1; k \leq n; k{+}{+})$ **do** (8)
  $i \leftarrow$ the next unscheduled task in L; (9)
  remove i from L; (10)
  find j such that $z_j = \max\{z_1, z_2, \ldots, z_m\}$; //Ties are broken arbitrarily. (11)
  $S_j \leftarrow S_j \cup \{i\}$; (12)
  $\mu^{(j)} \leftarrow \mu^{(j)} + \mu_i$; (13)
  $(\sigma^{(j)})^2 \leftarrow (\sigma^{(j)})^2 + \sigma_i^2$; (14)
  $z_j \leftarrow (T^* - \mu^{(j)}/s)/(\sigma^{(j)}/s)$; (15)
end for; (16)
calculate $F_T(s, s, \ldots, s, T^*)$; (17)
return $(S_1, S_2, \ldots, S_m)$ and $F_T(s, s, \ldots, s, T^*)$. (18)

Our heuristic algorithm for optimal stochastic task scheduling is given in Algorithm 2. It is clear that to keep $F_T(s, s, \ldots, s, T^*)$ as high as possible, we need to keep z_j as large as possible for all $1 \leq j \leq m$, where $z_j = (T^* - \mu^{(j)}/s)/(\sigma^{(j)}/s)$. In the beginning, we have $z_j = \infty$, for all $1 \leq j \leq m$ [line (6)]. As more and more tasks are scheduled on processor j, $\mu^{(j)}$ and $(\sigma^{(j)})^2$ get larger and larger, while z_j becomes smaller and smaller. Our strategy is to let the z_j's be reduced at about the same pace. For each task i, a processor j with the maximum z_j is chosen, and task i is scheduled on processor j [line (11)].

The n tasks are initially arranged in certain order L and then scheduled in this order. There are several arrangements of the n tasks into a list L. Let $c_i = \sigma_i/\mu_i$ be the coefficient of variation of task i, where $1 \leq i \leq n$. We consider the following heuristics.

- *Smallest μ First* (SμF): The n tasks are sorted such that $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$.
- *Largest μ First* (LμF): The n tasks are sorted such that $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_n$.
- *Smallest σ First* (SσF): The n tasks are sorted such that $\sigma_1 \leq \sigma_2 \leq \cdots \leq \sigma_n$.
- *Largest σ First* (LσF): The n tasks are sorted such that $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$.
- *Smallest c First* (ScF): The n tasks are sorted such that $c_1 \leq c_2 \leq \cdots \leq c_n$.
- *Largest c First* (LcF): The n tasks are sorted such that $c_1 \geq c_2 \geq \cdots \geq c_n$.

Table 1 Simulation results

n	SμF	LμF	SσF	LσF	ScF	LcF
20	0.1238265	0.2301079	0.1220124	0.2253457	0.1300818	0.1311873
30	0.1067940	0.2925648	0.1469017	0.3028942	0.2153746	0.2074393
40	0.2353938	0.3681579	0.2482962	0.3659853	0.2905738	0.2883837
50	0.2865000	0.4403613	0.3184542	0.4412833	0.3597027	0.3664775
60	0.3412605	0.4818616	0.3756390	0.5000767	0.4201616	0.4214347
70	0.5161491	0.5651946	0.4546225	0.5560961	0.4806573	0.4887583
80	0.4530198	0.5859283	0.5018688	0.6011409	0.5344322	0.5422561
90	0.5852184	0.6463159	0.5619365	0.6425824	0.5872445	0.5831224
100	0.5695922	0.6749998	0.5962498	0.6760803	0.6294984	0.6364103

4.3 Performance evaluation

The main purpose of this section is to examine the performance of the heuristic methods proposed in the last section in scheduling stochastic tasks.

In Table 1, we demonstrate our experimental results. We consider $m = 7$ identical processors. For each combination of the number of tasks n and the six heuristic methods M, where $n = 20, 30, \ldots, 100$, and $M \in \{\text{S}\mu\text{F}, \text{L}\mu\text{F}, \text{S}\sigma\text{F}, \text{L}\sigma\text{F}, \text{S}c\text{F}, \text{L}c\text{F}\}$, we generate 10,000 sets of n stochastic tasks, where μ_i is uniformly distributed in the range $[10.0, 30.0)$, and σ_i is uniformly distributed in the range $[0.15\mu_i, 0.25\mu_i)$. The time deadline is set as $T^* = 20(n/m)$. The energy constraint is set as $E^* = 25n$. The value of β is 0.9. The heuristic method M is applied to each set of stochastic tasks, and the mean of the 10,000 values of $F_T(s, s, \ldots, s, T^*)$ is reported. We have the following observations from Table 1.

- Different methods do result in different quality of scheduling.
- LμF performs better than SμF, and LσF performs better than SσF. However, LcF has about the same performs as ScF.
- Overall speaking, LμF and LσF are best methods among the six heuristic methods.

5 Conclusions

We have introduced the energy constrained stochastic task scheduling problem and pointed out that the problem is extremely challenging due to the sophistication of the two performance measures. We have made some initial efforts in solving this optimization problem. However, our investigation is far from being satisfactory, and there are much more work to be carried out. First, further research should be directed toward the optimal solutions to the problems of optimal processor speed setting and optimal stochastic task scheduling, and the integration of the two. Second, future research efforts can also be made to apply the algorithms to real-world tasks and applications. Third, the optimization problem proposed in this paper can be addressed as a multi-objective optimization problem, i.e., simultaneously maximizing the probability that

the total execution time does not exceed a given bound and the probability that the total energy consumption does not exceed a given bound. Fourth, more sophisticated task models (e.g., tasks with precedence constraints and communication costs) can be considered.

Acknowledgements The author deeply appreciates eighteen (18) anonymous reviewers for their corrections, criticism, and comments on the original manuscript.

Appendix 1: Derivation of $\partial F_T/\partial s_i$ and $\partial F_E/\partial s_i$

Notice that

$$\frac{\partial F_T(s_1, s_2, \ldots, s_n, T^*)}{\partial s_i} = \int_{-\infty}^{T^*} \frac{\partial f_T(x)}{\partial s_i} \mathrm{d}x.$$

Furthermore, we have

$$\begin{aligned}
&\frac{\partial f_T(x)}{\partial s_i} \\
&= \frac{1}{\sqrt{2\pi}} \left(-\frac{1}{\sigma_T^2} \frac{\partial \sigma_T}{\partial s_i} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \right. \\
&\quad \left. + \frac{1}{\sigma_T} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \left(-\frac{1}{2} \right) 2 \left(\frac{x-\mu_T}{\sigma_T} \right) \frac{-\partial \mu_T/\partial s_i \cdot \sigma_T - (x-\mu_T)\partial \sigma_T/\partial s_i}{\sigma_T^2} \right) \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T^2} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \left(\frac{\partial \sigma_T}{\partial s_i} - \left(\frac{x-\mu_T}{\sigma_T^2} \right) \left(\sigma_T \frac{\partial \mu_T}{\partial s_i} + (x-\mu_T) \frac{\partial \sigma_T}{\partial s_i} \right) \right) \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T^2} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \left(\frac{\partial \sigma_T}{\partial s_i} - \frac{\partial \mu_T}{\partial s_i} \left(\frac{x-\mu_T}{\sigma_T} \right) - \frac{\partial \sigma_T}{\partial s_i} \left(\frac{x-\mu_T}{\sigma_T} \right)^2 \right).
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
&\frac{\partial F_T(s_1, s_2, \ldots, s_n, T^*)}{\partial s_i} \\
&= \int_{-\infty}^{T^*} -\frac{1}{\sqrt{2\pi}\sigma_T^2} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \left(\frac{\partial \sigma_T}{\partial s_i} - \frac{\partial \mu_T}{\partial s_i} \left(\frac{x-\mu_T}{\sigma_T} \right) - \frac{\partial \sigma_T}{\partial s_i} \left(\frac{x-\mu_T}{\sigma_T} \right)^2 \right) \mathrm{d}x \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T^2} \left(\frac{\partial \sigma_T}{\partial s_i} \int_{-\infty}^{T^*} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \mathrm{d}x - \frac{\partial \mu_T}{\partial s_i} \int_{-\infty}^{T^*} \left(\frac{x-\mu_T}{\sigma_T} \right) \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \mathrm{d}x \right. \\
&\quad \left. - \frac{\partial \sigma_T}{\partial s_i} \int_{-\infty}^{T^*} \left(\frac{x-\mu_T}{\sigma_T} \right)^2 \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \mathrm{d}x \right).
\end{aligned}$$

By letting

$$y = \frac{x-\mu_T}{\sigma_T},$$

we have

$$\begin{aligned}
&\frac{\partial F_T(s_1, s_2, \ldots, s_n, T^*)}{\partial s_i} \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T}\left(\frac{\partial \sigma_T}{\partial s_i}\int_{-\infty}^{(T^*-\mu_T)/\sigma_T} \mathrm{e}^{-y^2/2}\mathrm{d}y - \frac{\partial \mu_T}{\partial s_i}\int_{-\infty}^{(T^*-\mu_T)/\sigma_T} y\mathrm{e}^{-y^2/2}\mathrm{d}y\right. \\
&\quad \left. -\frac{\partial \sigma_T}{\partial s_i}\int_{-\infty}^{(T^*-\mu_T)/\sigma_T} y^2\mathrm{e}^{-y^2/2}\mathrm{d}y\right).
\end{aligned}$$

Since

$$\int y\mathrm{e}^{-y^2/2}\mathrm{d}y = -\mathrm{e}^{-y^2/2},$$

and

$$\int y^2\mathrm{e}^{-y^2/2}\mathrm{d}y = -y\mathrm{e}^{-y^2/2} + \int \mathrm{e}^{-y^2/2}\mathrm{d}y,$$

we obtain

$$\begin{aligned}
&\frac{\partial F_T(s_1, s_2, \ldots, s_n, T^*)}{\partial s_i} \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T}\left(\frac{\partial \sigma_T}{\partial s_i}\sqrt{2\pi}F_Y\left(\frac{T^*-\mu_T}{\sigma_T}\right) - \frac{\partial \mu_T}{\partial s_i}\left(-\mathrm{e}^{-y^2/2}\right)\Big|_{-\infty}^{(T^*-\mu_T)/\sigma_T}\right. \\
&\quad \left. -\frac{\partial \sigma_T}{\partial s_i}\left(-y\mathrm{e}^{-y^2/2}\Big|_{-\infty}^{(T^*-\mu_T)/\sigma_T} + \sqrt{2\pi}F_y\left(\frac{T^*-\mu_T}{\sigma_T}\right)\right)\right) \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T}\left(\frac{\partial \sigma_T}{\partial s_i}\sqrt{2\pi}F_Y\left(\frac{T^*-\mu_T}{\sigma_T}\right) + \frac{\partial \mu_T}{\partial s_i}\mathrm{e}^{-(T^*-\mu_T)^2/2\sigma_T^2}\right. \\
&\quad \left. +\frac{\partial \sigma_T}{\partial s_i}\left(\left(\frac{T^*-\mu_T}{\sigma_T}\right)\mathrm{e}^{-(T^*-\mu_T)^2/2\sigma_T^2} - \sqrt{2\pi}F_Y\left(\frac{T^*-\mu_T}{\sigma_T}\right)\right)\right) \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T}\left(\frac{\partial \mu_T}{\partial s_i}\mathrm{e}^{-(T^*-\mu_T)^2/2\sigma_T^2} + \frac{\partial \sigma_T}{\partial s_i}\left(\frac{T^*-\mu_T}{\sigma_T}\right)\mathrm{e}^{-(T^*-\mu_T)^2/2\sigma_T^2}\right) \\
&= -\frac{1}{\sqrt{2\pi}\sigma_T}\mathrm{e}^{-(T^*-\mu_T)^2/2\sigma_T^2}\left(\frac{\partial \mu_T}{\partial s_i} + \left(\frac{T^*-\mu_T}{\sigma_T}\right)\frac{\partial \sigma_T}{\partial s_i}\right) \\
&= -f_T(T^*)\left(\frac{\partial \mu_T}{\partial s_i} + \left(\frac{T^*-\mu_T}{\sigma_T}\right)\frac{\partial \sigma_T}{\partial s_i}\right).
\end{aligned}$$

It is clear that

$$\frac{\partial \mu_T}{\partial s_i} = -\frac{\mu_i}{s_i^2}.$$

Since

$$\sigma_T = \left(\frac{\sigma_1^2}{s_1^2} + \frac{\sigma_2^2}{s_2^2} + \cdots + \frac{\sigma_n^2}{s_n^2}\right)^{1/2},$$

we get

$$\frac{\partial \sigma_T}{\partial s_i} = \frac{1}{2}\left(\frac{\sigma_1^2}{s_1^2} + \frac{\sigma_2^2}{s_2^2} + \cdots + \frac{\sigma_n^2}{s_n^2}\right)^{-1/2} \sigma_i^2 \left(-\frac{2}{s_i^3}\right) = -\frac{1}{\sigma_T} \cdot \frac{\sigma_i^2}{s_i^3}.$$

Consequently, we get

$$\frac{\partial F_T(s_1, s_2, \ldots, s_n, T^*)}{\partial s_i} = f_T(T^*)\left(\frac{\mu_i}{s_i^2} + \frac{1}{\sigma_T}\left(\frac{T^* - \mu_T}{\sigma_T}\right)\frac{\sigma_i^2}{s_i^3}\right).$$

In a similar way, we can also derive

$$\frac{\partial F_E(s_1, s_2, \ldots, s_n, E^*)}{\partial s_i} = -f_E(E^*)\left(\frac{\partial \mu_E}{\partial s_i} + \left(\frac{E^* - \mu_E}{\sigma_E}\right)\frac{\partial \sigma_E}{\partial s_i}\right).$$

It is clear that

$$\frac{\partial \mu_E}{\partial s_i} = (\alpha - 1)\mu_i s_i^{\alpha - 2},$$

and

$$\frac{\partial \sigma_E}{\partial s_i} = 2(\alpha - 1)\sigma_i^2 s_i^{2\alpha - 3}.$$

Consequently, we get

$$\frac{\partial F_E(s_1, s_2, \ldots, s_n, E^*)}{\partial s_i} = -(\alpha - 1) f_E(E^*)\left(\mu_i s_i^{\alpha - 2} + 2\left(\frac{E^* - \mu_E}{\sigma_E}\right)\sigma_i^2 s_i^{2\alpha - 3}\right).$$

Appendix 2: Calculation of $\partial F_i(\mathbf{y})/\partial y_j$

First, we have

$$\frac{\partial F_0(\mathbf{y})}{\partial y_0} = \frac{\partial F_0(\mathbf{y})}{\partial \phi} = 0,$$

and

$$\frac{\partial F_0(\mathbf{y})}{\partial y_j} = \frac{\partial F_0(\mathbf{y})}{\partial s_j} = \frac{\partial F_E(s_1, s_2, \ldots, s_n, E^*)}{\partial s_j}$$
$$= -(\alpha - 1) f_E(E^*) \left(\mu_j s_j^{\alpha-2} + 2 \left(\frac{E^* - \mu_E}{\sigma_E} \right) \sigma_j^2 s_j^{2\alpha-3} \right),$$

for all $1 \leq j \leq n$. Next, we have

$$\frac{\partial F_i(\mathbf{y})}{\partial y_0} = \frac{\partial F_i(\mathbf{y})}{\partial \phi} = (\alpha - 1) f_E(E^*) \left(\mu_i s_i^{\alpha-2} + 2 \left(\frac{E^* - \mu_E}{\sigma_E} \right) \sigma_i^2 s_i^{2\alpha-3} \right),$$

for all $1 \leq i \leq n$. Recall that

$$\frac{\partial f_T(x)}{\partial s_i}$$
$$= -\frac{1}{\sqrt{2\pi}\sigma_T^2} \mathrm{e}^{-(x-\mu_T)^2/2\sigma_T^2} \left(\frac{\partial \sigma_T}{\partial s_i} - \frac{\partial \mu_T}{\partial s_i} \left(\frac{x - \mu_T}{\sigma_T} \right) - \frac{\partial \sigma_T}{\partial s_i} \left(\frac{x - \mu_T}{\sigma_T} \right)^2 \right)$$
$$= \frac{f_T(x)}{\sigma_T} \left(-\frac{\partial \mu_T}{\partial s_i} \left(\frac{x - \mu_T}{\sigma_T} \right) + \frac{\partial \sigma_T}{\partial s_i} \left(1 - \left(\frac{x - \mu_T}{\sigma_T} \right)^2 \right) \right)$$
$$= \frac{f_T(x)}{\sigma_T} \left(\frac{\mu_i}{s_i^2} \left(\frac{x - \mu_T}{\sigma_T} \right) - \frac{1}{\sigma_T} \cdot \frac{\sigma_i^2}{s_i^3} \left(1 - \left(\frac{x - \mu_T}{\sigma_T} \right)^2 \right) \right),$$

which implies that

$$\frac{\partial f_T(T^*)}{\partial s_i} = \frac{f_T(T^*)}{\sigma_T} \left(\frac{\mu_i}{s_i^2} \left(\frac{T^* - \mu_T}{\sigma_T} \right) - \frac{1}{\sigma_T} \cdot \frac{\sigma_i^2}{s_i^3} \left(1 - \left(\frac{T^* - \mu_T}{\sigma_T} \right)^2 \right) \right).$$

Similarly, we can also get

$$\frac{\partial f_E(x)}{\partial s_i}$$
$$= -\frac{1}{\sqrt{2\pi}\sigma_E^2} \mathrm{e}^{-(x-\mu_E)^2/2\sigma_E^2} \left(\frac{\partial \sigma_E}{\partial s_i} - \frac{\partial \mu_E}{\partial s_i} \left(\frac{x - \mu_E}{\sigma_E} \right) - \frac{\partial \sigma_E}{\partial s_i} \left(\frac{x - \mu_E}{\sigma_E} \right)^2 \right)$$
$$= \frac{f_E(x)}{\sigma_E} \left(-\frac{\partial \mu_E}{\partial s_i} \left(\frac{x - \mu_E}{\sigma_E} \right) + \frac{\partial \sigma_E}{\partial s_i} \left(1 - \left(\frac{x - \mu_E}{\sigma_E} \right)^2 \right) \right)$$
$$= \frac{f_E(x)}{\sigma_E} \left(-(\alpha-1)\mu_i s_i^{\alpha-2} \left(\frac{x-\mu_E}{\sigma_E} \right) + 2(\alpha-1)\sigma_i^2 s_i^{2\alpha-3} \left(1 - \left(\frac{x-\mu_E}{\sigma_E} \right)^2 \right) \right),$$

which implies that

$$\frac{\partial f_E(E^*)}{\partial s_i} = (\alpha-1)\frac{f_E(E^*)}{\sigma_E}\left(-\mu_i s_i^{\alpha-2}\left(\frac{E^*-\mu_E}{\sigma_E}\right) + 2\sigma_i^2 s_i^{2\alpha-3}\left(1-\left(\frac{E^*-\mu_E}{\sigma_E}\right)^2\right)\right).$$

Hence, we have

$$\begin{aligned}
\frac{\partial F_i(\mathbf{y})}{\partial y_i} &= \frac{\partial F_i(\mathbf{y})}{\partial s_i} \\
&= \frac{\partial f_T(T^*)}{\partial s_i}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right) + f_T(T^*)\frac{\partial}{\partial s_i}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right) \\
&\quad +\phi(\alpha-1)\left(\frac{\partial f_E(E^*)}{\partial s_i}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\right. \\
&\quad \left.+f_E(E^*)\frac{\partial}{\partial s_i}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\right) \\
&= \frac{\partial f_T(T^*)}{\partial s_i}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right) \\
&\quad +f_T(T^*)\left(-\frac{2\mu_i}{s_i^3}+\sigma_i^2\left(\frac{-\partial\mu_T/\partial s_i\cdot\sigma_T^2 s_i^3-(T^*-\mu_T)(2\sigma_T\partial\sigma_T/\partial s_i\cdot s_i^3+\sigma_T^2 3s_i^2)}{\sigma_T^4 s_i^6}\right)\right) \\
&\quad +\phi(\alpha-1)\left(\frac{\partial f_E(E^*)}{\partial s_i}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\right. \\
&\quad +f_E(E^*)\left(\mu_i(\alpha-2)s_i^{\alpha-3}+2\sigma_i^2\left((2\alpha-3)s_i^{2\alpha-4}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right.\right. \\
&\quad \left.\left.\left.+s_i^{2\alpha-3}\left(\frac{-\partial\mu_E/\partial s_i\cdot\sigma_E-(E^*-\mu_E)\partial\sigma_E/\partial s_i}{\sigma_E^2}\right)\right)\right)\right) \\
&= \frac{\partial f_T(T^*)}{\partial s_i}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right) \\
&\quad +f_T(T^*)\left(-\frac{2\mu_i}{s_i^3}+\sigma_i^2\left(\frac{\sigma_T^2\mu_i s_i-(T^*-\mu_T)(-2\sigma_i^2+3\sigma_T^2 s_i^2)}{\sigma_T^4 s_i^6}\right)\right) \\
&\quad +\phi(\alpha-1)\left(\frac{\partial f_E(E^*)}{\partial s_i}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\right. \\
&\quad +f_E(E^*)\left((\alpha-2)\mu_i s_i^{\alpha-3}+2\sigma_i^2\left((2\alpha-3)s_i^{2\alpha-4}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right.\right. \\
&\quad \left.\left.\left.-(\alpha-1)s_i^{2\alpha-3}\left(\frac{\sigma_E\mu_i s_i^{\alpha-2}+2(E^*-\mu_E)\sigma_i^2 s_i^{2\alpha-3}}{\sigma_E^2}\right)\right)\right)\right),
\end{aligned}$$

for all $1 \le i \le n$, and

$$\begin{aligned}
\frac{\partial F_i(\mathbf{y})}{\partial y_j} &= \frac{\partial F_i(\mathbf{y})}{\partial s_j} \\
&= \frac{\partial f_T(T^*)}{\partial s_j}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right) + f_T(T^*)\frac{\partial}{\partial s_j}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right) \\
&\quad +\phi(\alpha-1)\left(\frac{\partial f_E(E^*)}{\partial s_j}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\right.
\end{aligned}$$

$$
\begin{aligned}
&+f_E(E^*)\frac{\partial}{\partial s_j}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\Bigg)\\
&=\frac{\partial f_T(T^*)}{\partial s_j}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right)\\
&\quad+f_T(T^*)\frac{\sigma_i^2}{s_i^3}\left(\frac{-\partial\mu_T/\partial s_j\cdot\sigma_T^2-(T^*-\mu_T)2\sigma_T\partial\sigma_T/\partial s_j}{\sigma_T^4}\right)\\
&\quad+\phi(\alpha-1)\left(\frac{\partial f_E(E^*)}{\partial s_j}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\right.\\
&\quad\left.+2f_E(E^*)\sigma_i^2 s_i^{2\alpha-3}\left(\frac{-\partial\mu_E/\partial s_j\cdot\sigma_E-(E^*-\mu_E)\partial\sigma_E/\partial s_j}{\sigma_E^2}\right)\right)\\
&=\frac{\partial f_T(T^*)}{\partial s_j}\left(\frac{\mu_i}{s_i^2}+\sigma_i^2\left(\frac{T^*-\mu_T}{\sigma_T^2 s_i^3}\right)\right)\\
&\quad+f_T(T^*)\frac{\sigma_i^2}{s_i^3}\left(\frac{\sigma_T^2\mu_j/s_j^2+2(T^*-\mu_T)\sigma_j^2/s_j^3}{\sigma_T^4}\right)\\
&\quad+\phi(\alpha-1)\left(\frac{\partial f_E(E^*)}{\partial s_j}\left(\mu_i s_i^{\alpha-2}+2\sigma_i^2 s_i^{2\alpha-3}\left(\frac{E^*-\mu_E}{\sigma_E}\right)\right)\right.\\
&\quad\left.-2(\alpha-1)f_E(E^*)\sigma_i^2 s_i^{2\alpha-3}\left(\frac{\sigma_E\mu_j s_j^{\alpha-2}+2(E^*-\mu_E)\sigma_j^2 s_j^{2\alpha-3}}{\sigma_E^2}\right)\right),
\end{aligned}
$$

for all $1\le i\le n$ and all $1\le j\ne i\le n$.

References

1. Ahmadizar F, Ghazanfari M, Ghomi SMTF (2010) Group shops scheduling with makespan criterion subject to random release dates and processing times. Comput Oper Res 37(1):152–162
2. Ando E, Nakata T, Yamashita M (2009) Approximating the longest path length of a stochastic DAG by a normal distribution in linear time. J Discrete Algorithms 7(4):420–438
3. Burden RL, Faires JD, Reynolds AC (1981) Numerical analysis, 2nd edn. Prindle, Weber & Schmidt, Boston
4. Canon L-C, Jeannot E (2009) Precise evaluation of the efficiency and the robustness of stochastic DAG schedules. Research report RR-6895 INRIA
5. Chandrakasan AP, Sheng S, Brodersen RW (1992) Low-power CMOS digital design. IEEE J Solid State Circuits 27(4):473–484
6. Chen L, Megow N, Rischke R, Stougie L (2015) Stochastic and robust scheduling in the cloud. In: 18th Int'l Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'15) and 19th Int'l Workshop on Randomization and Computation (RANDOM'15), pp 175–186
7. Chrétienne P, Coffman EG, Lenstra JK, Liu Z (eds) (1995) Scheduling theory and its applications. Wiley, Chichester
8. Dong F, Luo J, Song A, Jin J (2010) Resource load based stochastic DAGs scheduling mechanism for grid environment. In: 12th IEEE International Conference on High Performance Computing and Communications, pp 197–204
9. Furht B, Escalante A (eds) (2010) Handbook of cloud computing. Springer, New York

10. Gu J, Gu X, Gu M (2009) A novel parallel quantum genetic algorithm for stochastic job shop scheduling. J Math Anal Appl 355(1):63–81
11. Jeannot E, Namyst R, Roman J (eds) (2011) Euro-Par 2011 parallel processing. LNCS 6852. Springer, Berlin
12. Li K, Tang X, Li K (2014) Energy-efficient stochastic task scheduling on heterogeneous computing systems. IEEE Trans Parallel Distrib Syst 25(11):2867–2876
13. Li K, Tang X, Veeravalli B, Li K (2015) Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. IEEE Trans Comput 64(1):191–204
14. Megow N, Uetz M, Vredeveld T (2006) Models and algorithms for stochastic online scheduling. Math Oper Res 31(3):513–525
15. Möhring RH, Schulz AS, Uetz M (1999) Approximation in stochastic scheduling: the power of LP-based priority policies. J ACM 46(6):924–942
16. Peng Z, Cui D, Zuo J, Li Q, Xu B, Lin W (2015) Random task scheduling scheme based on reinforcement learning in cloud computing. Cluster Comput 18(4):1595–1607
17. Rothkopf MH (1966) Scheduling with random service times. Manage Sci 12(9):703–713
18. Sarin SC, Nagarajan B, Liao L (2010) Stochastic scheduling: expectation-variance analysis of a schedule. Cambridge University Press, Cambridge
19. Scharbrodt M, Schickinger T, Steger A (2006) A new average case analysis for completion time scheduling. J ACM 53(1):121–146
20. Shin SY, Gantenbein R, Kuo T-W, Hong J (2011) Reliable and autonomous computational science. Birkhäuser, Basel
21. Skutella M, Uetz M (2005) Stochastic machine scheduling with precedence constraints. SIAM J Comput 34(4):788–802
22. Tang X, Li K, Liao G, Fang K, Wu F (2011) A stochastic scheduling algorithm for precedence constrained tasks on grid. Future Gener Comput Syst 27(8):1083–1091
23. Tongsima S, Sha EHM, Chantrapornchai C, Surma DR, Passos NL (2000) Probabilistic loop scheduling for applications with uncertain execution time. IEEE Trans Comput 49(1):65–80
24. Wang L, Ranjan R, Chen J, Benatallah B (eds) (2012) Cloud computing: methodology, systems, and applications. CRC Press, Boca Raton
25. Weiss G (1992) Turnpike optimality of Smith's rule in parallel machines stochastic scheduling. Math Oper Res 17(2):255–270
26. Weiss G, Pinedo M (1980) Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. J Appl Probab 17(1):187–202
27. Xhafa F, Abraham A (eds) (2008) Metaheuristics for scheduling in distributed computing environments. Springer, Berlin
28. Zhai B, Blaauw D, Sylvester D, Flautner K (2004) Theoretical and practical limits of dynamic voltage scaling. In: Proceedings of the 41st Design Automation Conference, pp 868–873
29. Zhang W, Bai E, He H, Cheng AMK (2015) Solving energy-aware real-time tasks scheduling problem with shuffled frog leaping algorithm on heterogeneous platforms. Sensors 15(6):13778–13804
30. Zhang W, Xie H, Cao B, Cheng AMK (2014) Energy-aware real-time task scheduling for heterogeneous multiprocessors with particle swarm optimization algorithm. Math Prob Eng, Article ID 287475
31. Zhang Z, Su S, Zhang J, Shuang K, Xu P (2015) Energy aware virtual network embedding with dynamic demands: online and offline, part 3. Comput Netw 93:448–459