Sustainable Computing: Informatics and Systems 3 (2013) 109–118

Contents lists available at SciVerse ScienceDirect

Sustainable Computing: Informatics and Systems

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

Energy-efficient task scheduling algorithms on heterogeneous computers with continuous and discrete speeds

Luna Mingyi Zhang[a,*], Keqin Li[c], Dan Chia-Tien Lo[b], Yanqing Zhang[d]

[a] Department of Computer Science, College of Engineering, Cornell University, Ithaca, NY 14853, USA
[b] Department of Computer Science and Software Engineering, Southern Polytechnic State University, Marietta, GA 30060-2896, USA
[c] Department of Computer Science, State University of New York at New Paltz, New Paltz, NY 12561, USA
[d] Department of Computer Science, Georgia State University, Atlanta, GA 30302-3994, USA

ARTICLE INFO

Article history:
Received 10 July 2012
Accepted 28 January 2013

Keywords:
Green computing
Task scheduling
Energy reduction
Power-aware methods
Pollution reduction

ABSTRACT

A large number of computing servers and personal electronic devices waste a tremendous amount of energy and emit a considerable amount of carbon dioxide, which is the major contribution to the greenhouse effect. Thus, it is necessary to significantly reduce pollution and substantially lower energy usage. Green computing techniques are utilized in a myriad of applications in energy conservation and environment improvement. New green task scheduling algorithms for heterogeneous computers with changeable continuous speeds and changeable discrete speeds are developed to reduce energy consumption as much as possible and finish all tasks before a deadline. A newly proven theorem can determine the optimal speed for tasks assigned to a computer with continuous speeds. This project seeks to develop innovative green task scheduling algorithms that have two main steps: heuristically assigning tasks to computers, and setting optimal or near-optimal speeds for all tasks assigned to each computer. Sufficient simulation results indicate that the algorithm with the best task schedule varied. Thus, two hybrid algorithms for continuous and discrete speeds are created separately to obtain the best task schedule among candidate task schedules. Potential research applications include incorporating energy-efficient software into mobile devices, sensor networks, data centers, and cloud computing systems.

1. Introduction

Green computing is an emergent technology that applies intelligent optimization algorithms and advanced computing techniques to minimize energy consumption and reduce pollution from computing resources [1–6]. It is important for various applications in power management, energy reduction, pollution control, and environment enhancement [1–10]. Specifically, minimizing energy consumption on cloud servers and significantly reducing pollution produced by computers is an imperative research problem as energy costs are rising and the use of computers is increasing.

A typical personal computer with 17-in. LCD monitor requiring 145 W, if left on every day for one year, would use around 1270 kilowatt hours (kWh) of electricity. In 2007, the Environmental Protection Agency (EPA) predicted that the total energy consumed by U.S. data centers will double by 2012 [7]. In the U.S., about 61 billion kWh were used to power data centers in 2006 ($4.5 billion). For 2010, Google's electricity consumption was about 2.26 million MWh [25]. The EPA estimates that data centers annually consume the output of 15 average-sized power plants. Also, the EPA predicts that power consumption of data centers will soon increase to 12 GW, leading to the equivalent output of 25 power plants [9].

Computer usage accounts for 2% of anthropogenic CO_2 emission. Data center activities are estimated to release 62 million metric tons of CO_2 into the atmosphere [9]. For example, Google states that one Google search may generate about 0.2 g of carbon dioxide [4]. In 2011, Google estimated that its total carbon emissions for 2010 were 1.46 million metric tons [25]. The use of 1270 kWh of electricity is enough to emit about 1720 pounds of CO_2 into the environment.

However, Google and General Electric (GE) have applied green computing techniques to save power and reduce costs [4–6,25]. The energy used for each Google search was very small, which was 0.0003 kWh [4]. Also, Google's data centers use 50% less energy than the typical data center [25]. GE saves $2.5 million a year by implementing power control methods of Windows operating system onto its computers [6]. Nonetheless, computers can be further improved to become more energy-efficient, which is essential. The fundamental reason for our research in task scheduling on computers is to minimize computing energy consumption and consequently substantially reduce pollution produced by computers. Therefore, it is important to develop effective green computing

* Corresponding author. Tel.: +1 678 654 7090.
E-mail addresses: mingyiluna@yahoo.com, lmz22@cornell.edu (L.M. Zhang), lik@newpaltz.edu (K. Li), clo@spsu.edu (D.C.-T. Lo), yzhang@cs.gsu.edu (Y. Zhang).

2210-5379/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.suscom.2013.01.002

techniques to considerably lower energy usage on heterogeneous cloud servers. Thus, this project is critical and beneficial for the global community.

The problem of scheduling many independent tasks in a heterogeneous distributed computing system has been studied. A new green scheduling algorithm for saving energy in cloud computing was proposed [11]. However, in some cases, energy consumption was not considered as a factor that should be minimized [12]. In [13], the advantages for having task assignments were discussed, and the voltage level was considered as a factor instead of the speed level for each processor. A linear combination of the minimum and maximum processor frequencies was used to decrease energy consumption [14] and optimal frequencies were analyzed in [15]. In [16], variable processor speeds were considered, and idle intervals were exploited for power minimization. The algorithms incorporated dynamic voltage and frequency scaling to save energy [17]. Task scheduling algorithms were proposed to lower energy consumption by using shared slack reclamation on variable voltage/speed processors for task sets with precedence constraints and those without precedence [18]. In [22], two new heuristic energy-efficient task scheduling algorithms were proposed; tasks are assigned from the front in a task queue to a computer with minimum energy in the computer queue and then each computer's speed, with constraints, is adjusted to reach an overall minimal energy usage. Conventional approaches utilize maximum speeds, which is not energy-efficient. In [23], six innovative green task scheduling algorithms were developed; as many tasks as possible are assigned to a cloud server with lowest energy and then the speeds, with constraints, are lowered for all assigned tasks, assuming that the speeds are continuous, not discrete. The minimum total energy consumption for a computer with multiple identical processors occurred when all independent tasks were executed with the same power (or at the same speed) [1].

This paper focuses on developing novel green task scheduling algorithms for completing sequential tasks on heterogeneous servers with variable continuous and discrete speeds to minimize energy consumption via energy consumption parameters in a cloud computing environment with a certain deadline. For continuous speeds, it focuses on real applications with heterogeneous processors with different parameters. Also, speed constraints for processors were not taken into account [1]. A new method in this project uses various speed constraints for heterogeneous computers to better model real systems. For discrete speeds, it focuses on real applications with heterogeneous processors with different parameters using discrete speeds. For example, a microcontroller such as MSP430 has discrete speeds (i.e. clock frequencies). To better model real systems, new green task scheduling algorithms for heterogeneous computers with variable discrete speeds and energy consumption parameters are developed to reduce energy consumption as much as possible and finish all tasks before a certain deadline.

The rest of the paper is organized as follows. In Section 2, all of the definitions and parameters are given. In Section 3, the optimization problem is given, and new energy-efficient task scheduling algorithms with continuous speeds are proposed. In Section 4, the optimization problem is given, and new energy-efficient task scheduling algorithms with discrete speeds are proposed. In Section 5, the simulations and performance analysis are discussed. In Section 6, conclusions and future works are given.

2. Definitions and parameters

Let n computers be used to finish m tasks by the deadline time T (s) where $m \geq n$. Assume that m_i tasks P_k^i for $k = 1, 2, \ldots, m_i$ are executed on computer i for $m = \sum_{i=1}^{n} m_i$. A changeable speed (discrete or continuous) for task P_k^i is denoted as S_k^i for $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$. The speed is defined as the number of instructions per second. The number of instructions of task P_k^i is denoted as R_k^i. The execution time for task P_k^i on computer i is R_k^i/S_k^i, so the total execution time for m_i tasks P_k^i on computer i is $T_i = \sum_{k=1}^{m_i} R_k^i/S_k^i$. From [1], the energy (in J) for P_k^i on computer i is $E_k^i = C_i R_k^i [S_k^i]^{\alpha_i - 1}$ where C_i is a constant, $\alpha_i = 1 + (2/\phi_i) \geq 3$ for $0 < \phi_i \leq 1$, $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$. The total energy is $E = \sum_{i=1}^{n} \sum_{k=1}^{m_i} C_i R_k^i [S_k^i]^{\alpha_i - 1}$. Let $\gamma_k^i = C_i [S_k^i]^{\alpha_i - 1}$. For a speed S_k^i, γ_k^i is called the energy slope (a constant for each computer and task). Then, the total energy is $E = \sum_{i=1}^{n} \sum_{k=1}^{m_i} \gamma_k^i R_k^i$.

3. Energy-efficient task scheduling algorithms with continuous speeds

The problem is to minimize $E = \sum_{i=1}^{n} \sum_{k=1}^{m_i} \gamma_k^i R_k^i$ with constraints: $1 \leq m_i \leq m - n + 1$, $m = \sum_{i=1}^{n} m_i$, $\sum_{k=1}^{m_i} R_k^i/S_k^i \leq T$ and $0 < a_i \leq S_k^i \leq b_i$, where a_i is the minimum speed and b_i is the maximum speed of computer i, for $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$.

For a single computer case, the superscript i is omitted, so the energy for one computer is then $E = \sum_{k=1}^{m} C R_k [S_k]^{\alpha - 1}$, where R_k denotes the number of instructions and S_k denotes the execution speed (the number of instructions executed per second) for the kth task. Thus, the problem is to minimize $E = \sum_{k=1}^{m} C R_k [S_k]^{\alpha - 1}$ subject to $\sum_{k=1}^{m} R_k/S_k \leq T$ and $0 < a \leq S_k \leq b$ for $k = 1, 2, \ldots, m$, where a is the minimum speed and b is the maximum speed of the computer [22,23]. The minimal E occurs when each of the computers consumes energy at its minimal level to finish its tasks on time.

Theorem 1. *After m tasks with the same speed $\bar{S}$ for $0 < a \leq \bar{S} \leq b$ are successfully assigned to a computer for $\sum_{k=1}^{m} R_k/\bar{S} < T$, the minimal total energy consumption is $E = C[S^*]^{\alpha - 1} \sum_{k=1}^{m} R_k$ for $\alpha = 1 + 2/\phi \geq 3$ and $0 < \phi \leq 1$ when all tasks are executed with the same optimal speed S^* where $S^* = \text{maximum}\,(a, \sum_{k=1}^{m} R_k/T)$ [22].*

Proof. Because $\sum_{k=1}^{m} R_k/\bar{S} < T$, speeds S_k for $0 < a \leq S_k \leq b$ and $k = 1, 2, \ldots, m$ can be optimized to minimize the total energy consumption $E = \sum_{k=1}^{m} C R_k [S_k]^{\alpha - 1}$ [22].

The Lagrangian function [24] is defined as $L = \sum_{k=1}^{m} C R_k [S_k]^{\alpha - 1} - \lambda(T - \sum_{k=1}^{m} R_k/S_k)$.

$\partial L/\partial S_k = C R_k (\alpha - 1)[S_k]^{\alpha - 2} - \lambda(R_k/S_k^2) = 0$ for $k = 1, 2, \ldots, m$. The first order equalities are $\lambda(T - \sum_{k=1}^{m} R_k/S_k) = 0$ for $k = 1, 2, \ldots, m$. Also, $\sum_{k=1}^{m} R_k/S_k \leq T$, $0 < a \leq S_k \leq b$, and $\lambda \geq 0$ for $k = 1, 2, \ldots, m$. □

Case 1. When $\lambda > 0$, we have $C R_k (\alpha - 1)[S_k]^{\alpha - 2} - \lambda(R_k/S_k^2) = 0$. Thus, $S_k = (\lambda/C(\alpha - 1))^{1/\alpha} = S$ where S is a constant speed for $k = 1, 2, \ldots, m$. Because $\partial^2 L/\partial^2 S_k \,|S_k = S = C R_k (\alpha - 1)(\alpha - 2)[S]^{\alpha - 3} + 2\lambda(R_k/S^3) > 0$ for $\alpha = 1 + 2/\phi \geq 3$ and $0 < \phi \leq 1$, we obtain the minimum value of $E_k = C R_k [S_k]^{\alpha - 1}$ when $S_k = S$. We have $T - \sum_{k=1}^{m} R_k/S = 0$, so $S = \sum_{k=1}^{m} R_k/T$. Since $\sum_{k=1}^{m} R_k/\bar{S} < T$, $\sum_{k=1}^{m} R_k/T < \bar{S} \leq b$. Hence, $0 < S < \bar{S} \leq b$.

Case 1.1. If $0 < a < S < \bar{S} \leq b$, then the optimal speed is $S = \sum_{k=1}^{m} R_k/T$.

Case 1.2. If $0 < S \leq a$, then $0 < \sum_{k=1}^{m} R_k/T \leq a$. Thus, $0 < \sum_{k=1}^{m} R_k/a \leq T$, meaning that every task can be executed at the minimum speed a so the optimal speed is $S = a$.

From Cases 1.1 and 1.2, the optimal speed is maximum $(a, \sum_{k=1}^{m} R_k/T)$.

Case 2. When $\lambda = 0$, we have $C R_k (\alpha - 1)[S_k]^{\alpha - 2} = 0$. Thus, $S_k = 0$ which is invalid since $0 < a \leq S_k \leq b$.

From Cases 1 and 2, the optimal speed is $S^* = \text{maximum}\,(a, \sum_{k=1}^{m} R_k/T)$ [22].

Three green task scheduling algorithms, in which Theorem 1 is used for speed optimization, are proposed: (1) Shortest Task First for Computer with Minimum Energy (STF-CME) algorithm (shown in Fig. 1), (2) Longest Task First for Computer with Minimum Energy (LTF-CME) algorithm, and (3) Random Task for Computer with Minimum Energy (RT-CME) algorithm. The strategy for STF-CME (LTF-CME) algorithm is to assign the shortest (longest) task to a computer with the minimum energy slope. The strategy for RT-CME algorithm is to randomly assign a task to a computer with the minimum energy slope.

Similarly, three additional algorithms are proposed: (1) Shortest Task First for Random Computer (STF-RC) algorithm, (2) Longest Task First for Random Computer (LTF-RC) algorithm, and (3) Random Task for Random Computer (RT-RC) algorithm. The new strategy is to randomly select a computer to execute a task.

STF-CME algorithm is described below. N is the number of speed levels between minimum speed and maximum speed of all computers. t is the remaining time after finishing m_i tasks for computer i. M is the total number of successful task assignments.

STF-CME Algorithm [22,23]
Input: m tasks, n computers, deadline T, speeds S_k^i
Output: the successful task assignment with E for each computer and its optimal speed

Begin
Initially, all tasks in a task queue are sorted in an increasing order based on their numbers of instructions. Assign values to a_i and b_i. $j = 0$. $M = 0$.
for each speed level g (0 to $N-1$) **do**
Step 1: Calculate $S_k^i = a_i + g\left(\frac{b_i - a_i}{N-1}\right)$ for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m_i$.
Step 2: For each computer i at S_k^i, if there is enough remaining time to hold a task, calculate $\gamma_k^i = C_i[S_k^i]^{\alpha_i - 1}$.
Step 3: Assign every task to the computer with the minimum γ_k^i.
Step 4: If all tasks are assigned, store successful task assignment, $j = j + 1$, $M = M + 1$, and go to Step 5. Otherwise, there is no possible task assignment.
Step 5: Speed Optimization.
for each computer i (1 to n) **do**

$$t = T - \sum_{k=1}^{m_i} \frac{R_k^i}{S_k^i}$$

for each task k (1 to m) **do**
if ($t > 0$) **then**

$$S_k^i = \max\left(a_i, \frac{\sum_{k=1}^{m} R_k^i}{T}\right)$$

end if
end for
end for

Step 6: Calculate $E_j^* = \sum_{i=1}^{n}\sum_{k}^{m_i} \gamma_k^i R_k^i$ using optimized speed S_k^i for $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$.
end for
Step 7: $E = \min(E_1^*, E_2^*, \ldots, E_M^*)$.
Step 8: Output the successful task assignment with E for computer i and its optimal speed S_k^i for $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$.
End.

Different from STF-CME algorithm, LTF-CME algorithm initially sorts all tasks in a task queue in a decreasing order based on their numbers of instructions (R_k^i for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m_i$) and performs the same functionality of Steps 1–8. Different from STF-CME algorithm, RT-CME algorithm randomly assigns all tasks in a task queue to computers and performs the same functionality of Steps 1–8. STF-RC, LTF-RC and RT-RC algorithms have the same codes as those of STF-CME, LTF-CME and RT-CME algorithms except for Steps 2 and 3 since STF-RC, LTF-RC and RT-RC algorithms assign each task to a random computer.

4. Energy-efficient task scheduling algorithms with discrete speeds

The problem is to minimize $E = \sum_{i=1}^{n}\sum_{k=1}^{m_i} \gamma_k^i R_k^i$ with constraints: $1 \le m_i \le m - n + 1$, $m = \sum_{i=1}^{n} m_i$, $\sum_{k=1}^{m_i} R_k^i / S_k^i \le T$ and $0 < a_i \le \underline{S}_k^i \le b_i$ where $\underline{S}_k^i$ is a discrete speed, a_i is the minimum speed and b_i is the maximum speed of computer i, $\bar{S}_h^i$ is a discrete speed for computer i for a discrete speed level h, for $h = 1, 2, \ldots, g$, $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$.

Three green task scheduling algorithms using N discrete speeds for each computer are proposed: (1) discrete STF-CME algorithm, (2) discrete LTF-CME algorithm, and (3) discrete RT-CME algorithm. The strategy for discrete STF-CME (discrete LTF-CME) algorithm is to assign the shortest (longest) task to a computer with the minimum energy slope. The strategy for discrete RT-CME algorithm is to randomly assign a task to the computer. Similarly, three additional algorithms using N discrete speeds are proposed: (1) discrete STF-RC algorithm, (2) discrete LTF-RC algorithm, and (3) discrete RT-RC algorithm. The new strategy is to randomly select a computer to execute a task.

Discrete STF-CME algorithm is described below. Now, N is the number of discrete speeds for each computer. t is the remaining time after finishing m_i tasks for computer i. M is the number of successful task assignments.

Discrete STF-CME Algorithm
Input: m tasks, n computers, deadline T, discrete speeds $\underline{S}_k^i$
Output: the successful task assignment with E for each computer and its optimal speed

Begin
Initially, all tasks in a task queue are sorted in an increasing order based on their numbers of instructions. Assign values to a_i and b_i. $j = 0$. $M = 0$.
for each speed level g (0 to $N-1$) **do**
Step 1: Calculate all discrete speeds $\underline{S}_k^i = a_i + g\left(\frac{b_i - a_i}{N-1}\right)$ and $\bar{S}_h^i = \underline{S}_k^i$ for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m_i$.
Step 2: For each computer i at $\underline{S}_k^i$, if there is enough remaining time to hold a task, calculate $\gamma_k^i = C_i[\underline{S}_k^i]^{\alpha_i - 1}$.
Step 3: Assign every task to the computer with the minimum γ_k^i.
Step 4: If all tasks are assigned, $j = j + 1$, $M = M + 1$, calculate $E^j = \sum_{i=1}^{n}\sum_{k}^{m_i} \gamma_k^i R_k^i$, and go to Step 5. Otherwise, there is no possible task assignment.
Step 5: Discrete Speed Adjustment
for each computer i (1 to n) **do**

$$t = T - \sum_{k=1}^{m_i} \frac{R_k^i}{\underline{S}_k^i}$$

for each task k (1 to m) **do**
if ($t > 0$) **then**
if $\left(\frac{R_k^i}{(R_k^i/\underline{S}_k^i + t)} <= a_i\right)$ **then**
$t = t - (R_k^i/a_i - R_k^i/\underline{S}_k^i)$
$\underline{S}_k^i = a_i$
else
for each discrete speed level h (1 to g) **do**
if $\left(\bar{S}_h^i >= \frac{R_k^i}{(R_k^i/\underline{S}_k^i + t)}\right)$ **then**
$t = t - (R_k^i/\bar{S}_h^i - R_k^i/\underline{S}_k^i)$
$\underline{S}_k^i = \bar{S}_h^i$
$h = N$
end if
end for
end if
end if
end for
end for
Step 6: Calculate $E_j^* = \sum_{i=1}^{n}\sum_{k}^{m_i} \gamma_k^i R_k^i$ using discrete speed $\underline{S}_k^i$ for $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$.
end for
Step 7: $E = \min(E_1^*, E_2^*, \ldots, E_M^*)$.
Step 8: Output the successful task assignment with E for computer i and its discrete speed $\underline{S}_k^i$ for $i = 1, 2, \ldots, n$, and $k = 1, 2, \ldots, m_i$.
End.

Different from discrete STF-CME algorithm, discrete LTF-CME algorithm initially sorts all tasks in a task queue in a decreasing order based on their numbers of instructions (R_k^i for $i = 1, 2, \ldots, n$ and $k = 1, 2, \ldots, m_i$) and performs the same functionality of

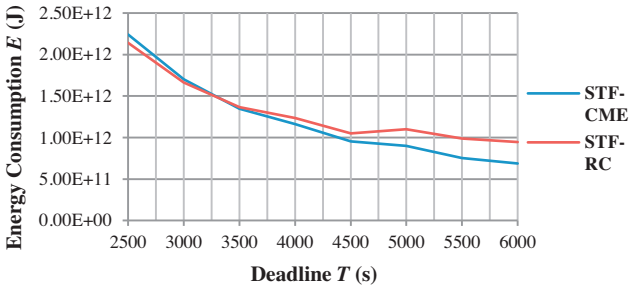


Fig. 1. STF-CME vs. STF-RC (100 tasks).

Steps 1–8 in discrete STF-CME algorithm. Different from discrete STF-CME algorithm, discrete RT-CME algorithm randomly assigns all tasks to computers and performs the same functionality of Steps 1–8 in discrete STF-CME algorithm. Discrete STF-RC, discrete LTF-RC, and discrete RT-RC algorithms have the same codes as those of discrete STF-CME, discrete LTF-CME, and discrete RT-CME algorithms except for Steps 2 and 3 in discrete STF-CME algorithm since discrete STF-RC, discrete LTF-RC, and discrete RT-RC algorithms assign each task to a randomly selected computer.

5. Simulations and performance analyses

Traditionally, computers use maximum speeds for real applications. Six conventional task scheduling algorithms, which correspond to the six new green task scheduling algorithms by using maximum speeds, are proposed: (1) Shortest Task First for Computer with Minimum Energy and Maximum Speeds (STF-CME-MS) algorithm, (2) Longest Task First for Computer with Minimum Energy and Maximum Speeds (LTF-CME-MS) algorithm, (3) Random Task for Computer with Minimum Energy and Maximum Speeds (RT-CME-MS) algorithm, (4) Shortest Task First for Random Computer and Maximum Speeds (STF-RC-MS) algorithm, (5) Longest Task First for Random Computer and Maximum Speeds (LTF-RC-MS) algorithm, and (6) Random Task for Random Computer and Maximum Speeds (RT-RC-MS) algorithm.

Similarly, six conventional task scheduling algorithms with discrete speeds are proposed: (1) discrete STF-CME-MS algorithm, (2) discrete LTF-CME-MS algorithm, (3) discrete RT-CME-MS algorithm, (4) discrete STF-RC-MS algorithm, (5) discrete LTF-RC-MS algorithm, and (6) discrete RT-RC-MS algorithm.

The energy reduction percentage is defined by $\beta = ((E_{MS} - E)/E_{MS}) \times 100\%$ where E_{MS} is the energy generated by an algorithm with maximum speeds (i.e. STF-CME-MS, LTF-CME-MS, RT-CME-MS, STF-RC-MS, LTF-RC-MS, RT-RC-MS, or discrete STF-CME-MS, discrete LTF-CME-MS, discrete RT-CME-MS, discrete STF-RC-MS, discrete LTF-RC-MS, and discrete RT-RC-MS), and where E is the energy calculated by an algorithm using optimal continuous or discrete speeds (i.e. STF-CME, LTF-CME, RT-CME, STF-RC, LTF-RC, RT-RC, or discrete STF-CME, discrete LTF-CME, discrete RT-CME, discrete STF-RC, discrete LTF-RC, discrete RT-RC).

All algorithms are implemented in Java. For every simulation, the number of instructions for task k (I_k), a_i, b_i, C_i, and α_i are randomly generated. $1000 \le I_k \le 11{,}000$ for $k = 1, 2, \ldots, m_i$, $a_i = 10 + \mu$ and $b_i = 50 + \mu$ for μ is a random number between 0 and $20k$, $1000 \le C_i \le 1500$, and $3 \le \alpha_i \le 4$ for $i = 1, 2, \ldots, n$. For computer i, N speed levels are calculated as $S_k^i = a_i + g((b_i - a_i)/(N-1))$ for $g = 0, 1, \ldots, N-1$. $N = 10$ for each simulation. In order to produce statistically reliable results, every result is an average of 100 random simulations [22,23].

5.1. Comparing algorithms based on energy consumption

5.1.1. Energy-efficient task scheduling algorithms with continuous speeds

Tables 1–6 show the energy consumption (in J) for a deadline T (in s). 100 tasks and 5 computers are used in Tables 1 and 2, 500 tasks and 10 computers are used in Tables 3 and 4, and 1000 tasks and 20 computers are used in Tables 5 and 6 [22]. Tables 7 and 8 show the overall average energy values calculated by using Tables 1–6 based on the number of computers and the number of tasks. From Table 7, STF-CME, LTF-CME, and RT-CME algorithms are more effective than STF-RC, LTF-RC, and RT-RC algorithms in terms of average energy consumption.

Table 1
Energy consumption (10^{12}) for STF-CME, LTF-CME, and RT-CME.

T (s)	STF-CME (J)	LTF-CME (J)	RT-CME (J)
2500	2.2413	2.2557	2.2534
3000	1.7027	1.7079	1.7081
3500	1.3479	1.3522	1.3526
4000	1.1608	1.1618	1.1619
4500	0.9525	0.9553	0.9547
5000	0.8993	0.9030	0.9027
5500	0.7542	0.7567	0.7571
6000	0.6863	0.6870	0.6875

Table 2
Energy consumption (10^{12}) for STF-RC, LTF-RC, and RT-RC.

T (s)	STF-RC (J)	LTF-RC (J)	RT-RC (J)
2500	2.1408	2.1418	2.1428
3000	1.6622	1.6676	1.6647
3500	1.3655	1.3479	1.3615
4000	1.2349	1.2265	1.2296
4500	1.0506	1.0450	1.0482
5000	1.1001	1.1009	1.0990
5500	0.9879	0.9719	0.9810
6000	0.9455	0.9313	0.9396

Table 3
Energy consumption (10^{13}) for STF-CME, LTF-CME, and RT-CME.

T (s)	STF-CME (J)	LTF-CME (J)	RT-CME (J)
4600	2.7597	2.7532	2.7553
4700	2.6952	2.6990	2.6994
4800	2.5361	2.5353	2.5367
4900	2.4745	2.4746	2.4751
5000	2.5805	2.5863	2.5864
5100	2.4153	2.4168	2.4175
5200	2.2796	2.2816	2.2816
5300	2.2389	2.2474	2.2472

Table 4
Energy consumption (10^{13}) for STF-RC, LTF-RC, and RT-RC.

T (s)	STF-RC (J)	LTF-RC (J)	RT-RC (J)
4600	2.6648	2.6628	2.6598
4700	2.5886	2.5778	2.5842
4800	2.4979	2.4869	2.4854
4900	2.4877	2.4754	2.4790
5000	2.6166	2.6233	2.6219
5100	2.4895	2.4834	2.4885
5200	2.4429	2.4169	2.4222
5300	2.3293	2.3167	2.3173

Table 5
Energy consumption (10^{14}) for STF-CME, LTF-CME, and RT-CME.

T (s)	STF-CME (J)	LTF-CME (J)	RT-CME (J)
2800	1.9893	1.9873	1.9886
2900	1.9052	1.9001	1.9017
3000	1.7928	1.7931	1.7942
3100	1.6987	1.7021	1.7026
3200	1.5950	1.5968	1.5976
3300	1.5307	1.5309	1.5313
3400	1.4470	1.4475	1.4474

Table 6
Energy consumption (10^{14}) for STF-RC, LTF-RC, and RT-RC.

T (s)	STF-RC (J)	LTF-RC (J)	RT-RC (J)
2800	2.1443	2.1192	2.1300
2900	2.0698	2.0560	2.0527
3000	1.9983	1.9889	1.9811
3100	1.9138	1.8999	1.8969
3200	1.8581	1.8448	1.8492
3300	1.7846	1.7706	1.7679
3400	1.7025	1.6988	1.7016

Table 7
Average energy consumption (10^{12}) for STF-CME, LTF-CME, and RT-CME.

No. of tasks	STF-CME (J)	LTF-CME (J)	RT-CME (J)
100	1.2181	1.2224	1.2222
500	24.9748	24.9928	24.9990
1000	170.8386	170.8257	170.9057

Table 8
Average energy consumption (10^{12}) for STF-RC, LTF-RC, and RT-RC.

No. of tasks	STF-RC (J)	LTF-RC (J)	RT-RC (J)
100	1.3109	1.3041	1.3083
500	25.1466	25.0540	25.0729
1000	192.4486	191.1171	191.1343

5.1.2. Energy-efficient task scheduling algorithms with discrete speeds

Tables 9–14 show the energy consumption (in J) for a deadline T (in s). 100 tasks and 5 computers are used in Tables 9 and 10, 500 tasks and 10 computers are used in Tables 11 and 12, and 1000 tasks and 20 computers are used in Tables 13 and 14. Tables 15 and 16 show the overall average energy values calculated by using Tables 9–14 based on the number of computers and the number of tasks. From Tables 15 and 16, discrete STF-CME, discrete LTF-CME, and discrete RT-CME algorithms are more effective than discrete STF-RC, discrete LTF-RC, and discrete RT-RC algorithms in terms of average energy consumption.

Table 9
Energy consumption (10^{9}) for Discrete STF-CME, Discrete LTF-CME, and Discrete RT-CME.

T (s)	Discrete STF-CME (J)	Discrete LTF-CME (J)	Discrete RT-CME (J)
3000	12.864	12.883	11.978
4000	11.365	11.457	10.725
5000	7.892	7.893	7.857
6000	6.590	6.574	6.536
7000	5.323	5.273	5.276
8000	4.421	4.378	4.379
9000	3.712	3.687	3.687
10,000	3.206	3.186	3.186

Table 10
Energy consumption (10^{9}) for Discrete STF-RC, Discrete LTF-RC, and Discrete RT-RC.

T (s)	Discrete STF-RC (J)	Discrete LTF-RC (J)	Discrete RT-RC (J)
3000	12.911	12.201	12.925
4000	11.250	10.693	11.120
5000	8.732	8.367	8.672
6000	8.427	8.174	8.330
7000	7.823	7.727	7.806
8000	7.348	7.304	7.317
9000	6.527	6.558	6.508
10,000	6.740	6.871	6.959

Table 11
Energy consumption (10^{11}) for Discrete STF-CME, Discrete LTF-CME, and Discrete RT-CME.

T (s)	Discrete STF-CME (J)	Discrete LTF-CME (J)	Discrete RT-CME (J)
6000	2.260	2.245	2.185
7000	1.795	1.791	1.772
8000	1.435	1.427	1.424
9000	1.324	1.320	1.321
10,000	1.055	1.049	1.050
11,000	0.891	0.887	0.888
12,000	0.815	0.813	0.813
13,000	0.702	0.700	0.699

Table 12
Energy consumption (10^{11}) for Discrete STF-RC, Discrete LTF-RC, and Discrete RT-RC.

T (s)	Discrete STF-RC (J)	Discrete LTF-RC (J)	Discrete RT-RC (J)
6000	2.406	2.361	2.387
7000	2.052	2.032	2.027
8000	1.830	1.817	1.826
9000	1.804	1.797	1.795
10,000	1.663	1.673	1.678
11,000	1.539	1.530	1.538
12,000	1.584	1.581	1.572
13,000	1.525	1.509	1.529

Table 13
Energy consumption (10^{12}) for Discrete STF-CME, Discrete LTF-CME, and Discrete RT-CME.

T (s)	Discrete STF-CME (J)	Discrete LTF-CME (J)	Discrete RT-CME (J)
3000	2.130	2.125	2.101
4000	1.354	1.348	1.349
5000	0.966	0.964	0.965
6000	0.757	0.755	0.756
7000	0.569	0.568	0.568
8000	0.499	0.498	0.498
9000	0.418	0.417	0.417

Table 14
Energy consumption (10^{12}) for Discrete STF-RC, Discrete LTF-RC, and Discrete RT-RC.

T (s)	Discrete STF-RC (J)	Discrete LTF-RC (J)	Discrete RT-RC (J)
3000	2.286	2.245	2.264
4000	1.713	1.695	1.700
5000	1.478	1.474	1.469
6000	1.376	1.367	1.364
7000	1.268	1.264	1.265
8000	1.275	1.274	1.270
9000	1.272	1.268	1.270

5.2. Comparing algorithms based on energy reduction percentages

5.2.1. Energy-efficient task scheduling algorithms with continuous speeds

Tables 17–20 show the energy reduction percentages for the algorithms. When $a_i = 10 + \mu$ and $b_i = 50 + \mu$ for μ is a random number between 0 and $20k$ for $k = 1, 2, \ldots, m_i$, Tables 17 and 18 show that STF-CME, LTF-CME, RT-CME, STF-RC, LTF-RC, and RT-RC algorithms using optimal continuous speeds are more effective than conventional STF-CME-MS, LTF-CME-MS, RT-CME-MS, STF-RC-MS,

Table 15
Average energy consumption (10^9) for Discrete STF-CME, Discrete LTF-CME, and Discrete RT-CME.

No. of Tasks	Discrete STF-CME (J)	Discrete LTF-CME (J)	Discrete RT-CME (J)
100	6.922	6.916	6.703
500	128.5	127.9	126.9
1000	956.1	953.6	950.6

Table 16
Average energy consumption (10^9) for Discrete STR-RC, Discrete LTF-RC, and Discrete RT-RC.

No. of tasks	Discrete STF-RC (J)	Discrete LTF-RC (J)	Discrete RT-RC (J)
100	8.720	8.487	8.705
500	180.0	178.8	179.4
1000	1524	1512	1515

Table 17
Energy reduction percentages for STF-CME, LTF-CME, and RT-CME using Optimal Continuous Speeds.

No. of tasks	T (s)	STF-CME (%)	LTF-CME (%)	RT-CME (%)
100	2500	98.042	98.051	97.631
500	5000	98.570	99.227	98.969
1000	3000	98.414	99.235	98.924

Table 18
Energy reduction percentages for STF-RC, LTF-RC, and RT-RC using Optimal Continuous Speeds.

No. of tasks	T (s)	STF-RC (%)	LTF-RC (%)	RT-RC (%)
100	2500	97.605	97.818	97.624
500	5000	98.838	98.907	98.865
1000	3000	98.654	98.960	98.757

Table 19
Energy reduction percentages for STF-CME, LTF-CME, and RT-CME with a Speed Difference of 10.

No. of tasks	T (s)	STF-CME (%)	LTF-CME (%)	RT-CME (%)
100	6500	96.362	96.762	96.601
500	15,000	98.279	98.476	98.408
1000	3000	98.120	98.390	98.287

Table 20
Energy reduction percentages for STF-RC, LTF-RC, and RT-RC with a Speed Difference of 10.

No. of tasks	T (s)	STF-RC (%)	LTF-RC (%)	RT-RC (%)
100	6500	96.600	96.627	96.610
500	15,000	98.423	98.439	98.425
1000	3000	98.316	98.336	98.317

LTF-RC-MS, and RT-RC-MS algorithms using maximum speeds in reducing energy consumptions [22].

When $a_i = 10 + \eta$ and $b_i = 20 + \eta$ for η is a random number between 0 and 5 for $k = 1, 2, \ldots, m_i$, Tables 19 and 20 with a speed difference of 10 show energy reduction percentages of at least 96%, which are very close to those of at least 97% shown in Tables 17 and 18 with a speed difference of 40. This also indicates that the new algorithms are more effective than the conventional ones.

Table 21
Energy reduction percentages for discrete STF-CME, discrete LTF-CME, and discrete RT-CME.

No. of tasks	Discrete STF-CME (%)	Discrete LTF-CME (%)	Discrete RT-CME (%)
100	96.712	97.012	97.261
500	98.642	99.235	99.060
1000	98.765	99.376	99.172

Table 22
Energy reduction percentages for discrete STF-RC, discrete LTF-RC, and discrete RT-RC.

No. of tasks	Discrete STF-RC (%)	Discrete LTF-RC (%)	Discrete RT-RC (%)
100	96.775	96.295	96.639
500	98.573	98.670	98.620
1000	98.458	98.649	98.536

5.2.2. Energy-efficient task scheduling algorithms with discrete speeds

Tables 21 and 22 show that discrete STF-CME, discrete LTF-CME, discrete RT-CME, discrete STF-RC, discrete LTF-RC, and discrete RT-RC algorithms are more effective than conventional discrete STF-CME-MS, discrete LTF-CME-MS, discrete RT-CME-MS, discrete STF-RC-MS, discrete LTF-RC-MS, and discrete RT-RC-MS algorithms, which use maximum speeds.

5.3. Hybrid algorithms

5.3.1. Energy-efficient task scheduling algorithms with continuous speeds

From Tables 1–6, the algorithm with the best task schedule varies under different conditions. For example, Table 2 shows that STF-RC is the best in two cases, LTF-RC is the best in five cases, and RT-RC is the best in one case. Thus, a hybrid algorithm for continuous speeds is developed to obtain the overall best task schedule among the six heuristic task schedules. It is shown below.

Begin
Step 1: Perform RT-CME algorithm.
Step 2: Perform LTF-CME algorithm.
Step 3: Perform STF-CME algorithm.
Step 4: Perform LTF-RC algorithm.
Step 5: Perform RT-RC algorithm.
Step 6: Perform STF-RC algorithm.
Step 7: Output the final successful task assignment with the minimum energy among six successful task assignments from Steps 1–6.
End

5.3.2. Energy-efficient task scheduling algorithms with discrete speeds

From Tables 9–15, the algorithm with the best task schedule varies under different conditions. Thus, a hybrid algorithm for discrete speeds is developed to obtain the overall best task schedule. It is shown below.

Begin
Step 1: Perform discrete RT-CME algorithm.
Step 2: Perform discrete LTF-CME algorithm.
Step 3: Perform discrete STF-CME algorithm.
Step 4: Perform discrete LTF-RC algorithm.
Step 5: Perform discrete RT-RC algorithm.
Step 6: Perform discrete STF-RC algorithm.
Step 7: Output the final successful task assignment with the minimum energy among six successful task assignments from Steps 1–6.
End

5.4. Comparing algorithms based on deadline T (for energy-efficient task scheduling algorithms with continuous speeds)

From Fig. 1, STF-RC performs better than STF-CME when T is less than about 3300. This observation is true for both LTF-RC from

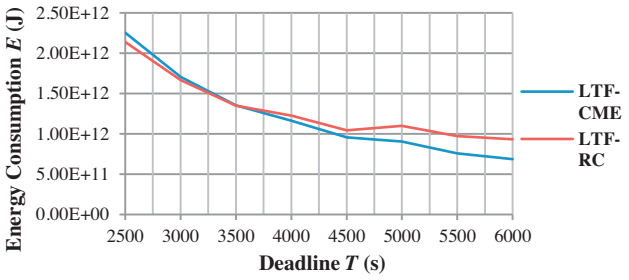


Fig. 2. LTF-CME vs. LTF-RC (100 tasks).

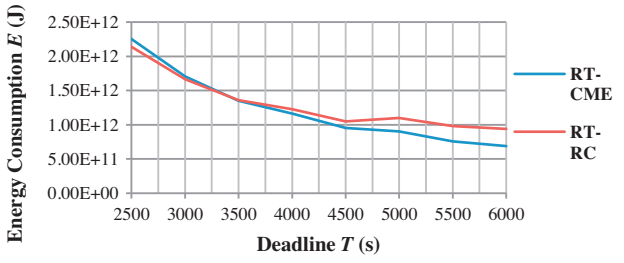


Fig. 3. RT-CME vs. RT-RC (100 tasks).

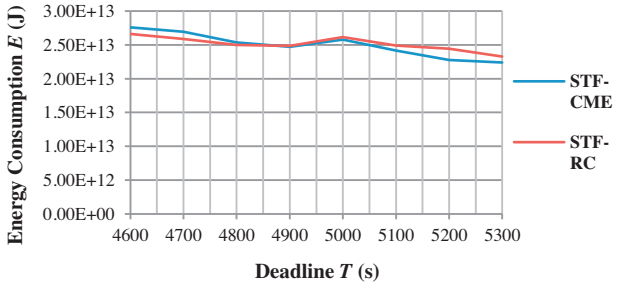


Fig. 4. STF-CME vs. STF-RC (500 tasks).

Fig. 2 and RT-RC from Fig. 3. The reason is that when T is very small, computers with large energy and those with small energy have similar chances of getting tasks by STF-CME, so it is possible for STF-RC to be more effective because of better task assignments on all computers than STF-CME. This reason also applies to LTF-RC and RT-RC.

However, STC-CME performs much better than STF-RC when T is larger than 3300. This observation is also true for both LTF-CME from Fig. 2 and RT-CME from Fig. 3. As the deadline increases, STC-CME assigns fewer tasks to the computer with high energy, but STF-RC equally assigns tasks to all computers. This fact applies to LTF-RC and RT-RC. Thus, total energy consumption when as many tasks as possible are given to the computer with lowest energy is much less than that when tasks are assigned to randomly selected computers.

From Fig. 4, STF-RC performs better than STF-CME when T is less than about 4880. This observation is true for LTF-RC from Fig. 5 and RT-RC from Fig. 6. However, from Fig. 7, STF-CME always performs better than STF-RC. The reason may be that there is a very small probability for STF-RC to assign tasks to computers with lower energy as the number of computers is increased to 20 and the number of tasks is increased to 1000, respectively. This finding is true for LTF-CME from Fig. 8 and RT-CME from Fig. 9.

The energy consumption difference between STF-RC and STF-CME also increases as the deadline T increases, indicating that STF-CME becomes increasingly more effective than STF-RC when the deadline passes a certain point. This is also true for LTF-RC and

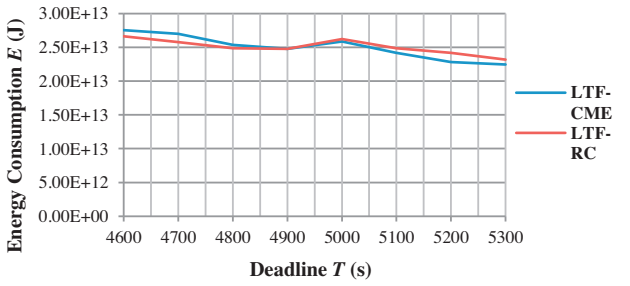


Fig. 5. LTF-CME vs. LTF-RC (500 tasks).

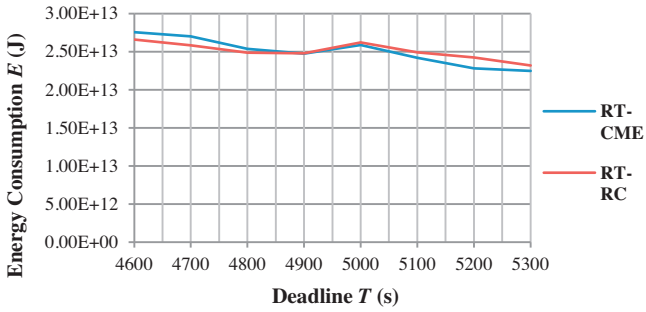


Fig. 6. RT-CME vs. RT-RC (500 tasks).

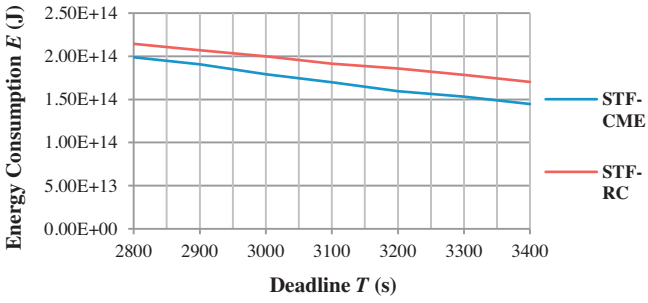


Fig. 7. STF-CME vs. STF-RC (1000 tasks).

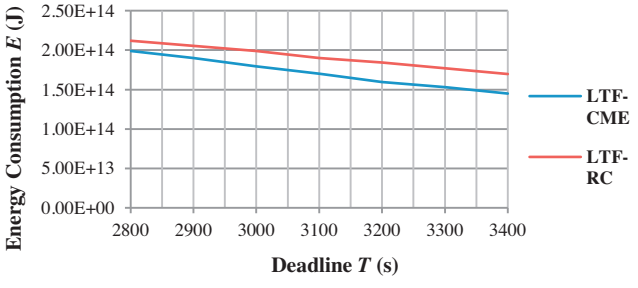


Fig. 8. LTF-CME vs. LTF-RC (1000 tasks).

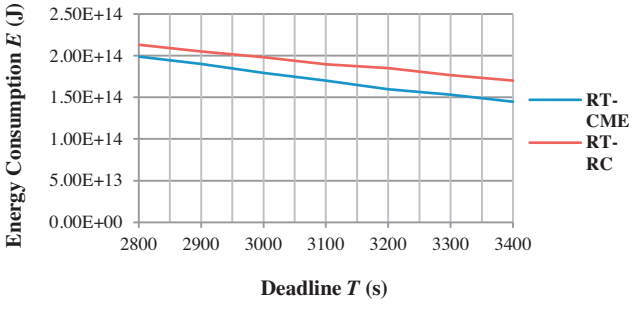


Fig. 9. RT-CME vs. RT-RC (1000 tasks).

Table 23
Energy reduction percentages β for $\alpha_i = 3$.

$\bar{S}_i/b_i$	0.1	0.3	0.5	0.7	0.9
β	99%	91%	75%	51%	19%

LTF-CME and RT-RC and RT-CME. Furthermore, each energy curve generally slopes downward because as the deadline increases, there is more time for each computer's speed to be further reduced.

5.5. Theoretical analysis (for energy-efficient task scheduling algorithms with continuous speeds)

When the optimal speed is $\bar{S}_i = \text{maximum}(a_i, \sum_{k=1}^{m_i} R_k^i/T)$ for computer i, we have

$$\beta = \frac{E_{MS} - E_{OS}}{E_{MS}} \times 100\% = \frac{\sum_{i=1}^{n}\sum_{k=1}^{m_i} C_i R_k^i [b_i]^{\alpha_i - 1} - \sum_{i=1}^{n}\sum_{k=1}^{m_i} C_i R_k^i [\bar{S}_i]^{\alpha_i - 1}}{\sum_{i=1}^{n}\sum_{k=1}^{m_i} C_i R_k^i [b_i]^{\alpha_i - 1}} \times 100\%.$$

Hence, $\beta = ([b_i]^{\alpha_i-1} - [\bar{S}_i]^{\alpha_i-1})/[b_i]^{\alpha_i-1} \times 100\% = (1 - (\bar{S}_i/b_i)^{\alpha_i-1}) \times 100\%$. The minimum of β is 0% when $\bar{S}_i = b_i$, and the maximum of β is $(1 - (a_i/b_i)^{\alpha_i-1}) \times 100\%$ when $\bar{S}_i = a_i$. Table 23 shows a few values for β. From Table 23, the smaller $\bar{S}_i/b_i$ is, the more energy is reduced. When $\bar{S}_i/b_i = 0.9$, $\beta = 19\%$, which is still a significant percentage of energy reduction [22].

5.6. Experimental analysis

5.6.1. Energy-efficient task scheduling algorithms with continuous speeds

Fig. 10 summarizes all data in Tables 1–6 and includes the data produced by the hybrid algorithm. These results are statistically valid because each value in Tables 1–8 and 17–20 is an average of 100 random simulation results, and different conditions (the deadline, the number of tasks, and the number of computers) are used for a sufficient number of simulations. It is clear that STF-CME, LTF-CME, and RT-CME are more effective than STF-RC, LTF-RC, and RT-RC. The method of assigning tasks to the computer with lower energy reduces more energy consumption than that of assigning tasks randomly to any computer. Fig. 10 shows the algorithms ranked (best to worst) as Hybrid, RT-CME, LTF-CME, STF-CME, LTF-RC, RT-RC, and STF-RC.

5.6.2. Energy-efficient task scheduling algorithms with discrete speeds

Fig. 11 summarizes all data in Tables 9–14 and includes the data produced by the discrete hybrid algorithm. These results are statistically valid because each value in Tables 9–16 and 21 and 22 is an average of 100 random simulation results, and different conditions (the deadline, the number of tasks, and the number of computers)

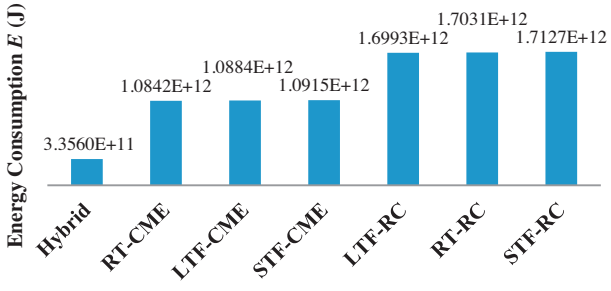


Fig. 10. Average energy consumption for Tables 1–6.

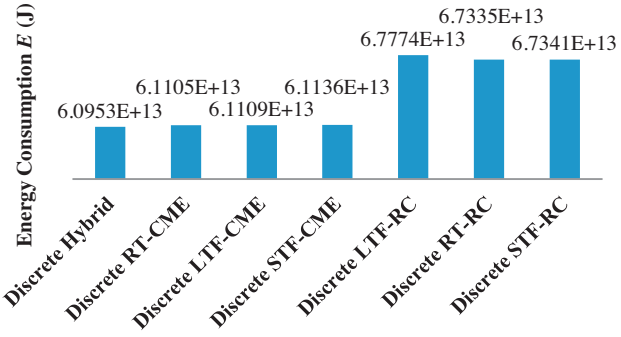


Fig. 11. Average energy consumption for Tables 9–14.

are used for a sufficient number of simulations. It is clear that discrete STF-CME, discrete LTF-CME, and discrete RT-CME can save more energy on average than discrete STF-RC, discrete LTF-RC, and discrete RT-RC. The method of assigning tasks to the computer with lower energy reduces more energy consumption than that of assigning tasks randomly to any computer. Fig. 11 shows the algorithms ranked (best to worst) as discrete Hybrid, discrete RT-CME, discrete LTF-CME, discrete STF-CME, discrete LTF-RC, discrete RT-RC, and discrete STF-RC.

6. Conclusions and future works

For heterogeneous computers with continuous speeds, seven new energy-efficient task scheduling algorithms are successfully developed to substantially decrease energy usage and finish all tasks before a deadline. Different results from simulations under various conditions indicate that the overall best algorithm is the hybrid algorithm.

For heterogeneous computers with discrete speeds, seven new energy-efficient task scheduling algorithms are also successfully developed to reduce energy consumption and finish all tasks before a deadline. Various results from simulations under different conditions indicate that the discrete hybrid algorithm is the overall best algorithm.

This paper presents new energy-efficient task scheduling algorithms for both continuous and discrete processor speeds with detailed simulation and analyses. Based on the simulation results, the hybrid algorithm outperforms others for both continuous and discrete speeds.

In the future, a new energy-efficient task scheduling algorithm using discrete speeds will generate all N^n speed permutations if N^n is small in order to find the best discrete speed permutation or will generate a large number of discrete speed permutations if N^n is too large in order to effectively find a better discrete speed permutation than the current discrete hybrid algorithm. In addition, a novel discrete energy-efficient task scheduling algorithm for homogeneous microcontrollers will be incorporated into embedded software to increase the battery lifetime for mobile devices [19–21].

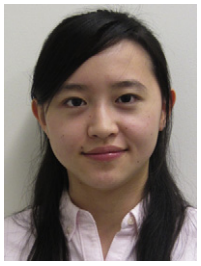
The future energy-efficient task scheduling algorithms with intelligent heuristics will be integrated into a myriad of practical applications ranging from a small scale such as mobile devices, to a large scale such as data centers and cloud computing systems. These new energy-efficient algorithms will have a globally positive impact on building a more energy-efficient society, making a cleaner environment, and creating a greener world.

Further questions to be studied remain. The first is finding the best task assignment among an enormous number of successful task assignments. The second is finding the best speeds of all heterogeneous cloud servers among a huge number of speed levels. The third is determining C_i and α_i for $E = \sum_{i=1}^{n}\sum_{k=1}^{m_i} C_i R_k^i [S_k^i]^{\alpha_i - 1}$ through practical experiments and statistical data analysis.

Moreover, new effective task scheduling algorithms with intelligent heuristics will be developed for data centers and cloud computing systems. Real simulations will be conducted to evaluate the future energy-efficient task scheduling algorithms.

References

[1] K. Li, Performance analysis of power-aware task scheduling algorithms on multiprocessor computers with dynamic voltage and speed, IEEE Transactions on Parallel and Distributed Systems 19 (11) (2008) 1484–1497.
[2] Technical Area of Green Computing, IEEE Technical Committee on Scalable Computing (TCSC). Available: http://sites.google.com/site/greencomputingproject/
[3] Report to Congress on Server and Data Center Energy Efficiency: Public Law 109-431, U.S. Environmental Protection Agency ENERGY STAR Program, August 2, 2007.
[4] Efficient Computing. Available: http://www.google.com/corporate/green/datacenters/
[5] Save energy. Save money. Make a difference. Available: http://www.google.com/powermeter/about/index.html
[6] S. Ryan, General Electric Saves nearly $6.5 M with Computer Power Management Features, US EPA ENERGY STAR, Program 202-343-9123.
[7] Green Computing: A CoSN Leadership Initiative. Available: http://www.cosn.org/Initiatives/GreenComputing/InterestingFacts/tabid/4639/Default.aspx
[8] M. Nielsen, A. Kucera, P.B. Miltersen, C. Palamidessi, P. Tuma, F. Valencia (Eds.), SOFSEM 2009: Theory and Practice of Computer Science 35th Conference on Current Trends in Theory and Practice of Computer Science, Špindleruv Mlýn, Czech Republic, January 24-30, 2009. Proceedings, Springer, Berlin/Heidelberg, 2009.
[9] H. Cademartori, Green Computing Beyond the Data Center, 2007. Available: http://www.powersavesoftware.com/Download/PS_WP_GreenComputing_EN.pdf
[10] S. Albers, Energy-efficient algorithms, Communications of the ACM 53 (May (5)) (2010) 86–96.
[11] V.T.D. Truong, Y. Sato, Y. Inoguchi, Performance evaluation of a green scheduling algorithm for energy savings in cloud computing, in: Proc. 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), April, 2010, pp. 1–8.
[12] A. Kamthe, S.-Y. Lee, Stochastic approach to scheduling multiple divisible tasks on a heterogeneous distributed computing system, in: Proc. IEEE International Parallel and Distributed Processing Symposium, 2007, pp. 1–11.
[13] L.K. Goh, B. Veeravalli, S. Viswanathan, Design of fast and efficient energy-aware gradient-based scheduling algorithms for heterogeneous embedded multiprocessor systems, IEEE Transactions on Parallel and Distributed Systems 20 (1) (2009) 1–12.
[14] N.B. Rizvandi, J. Taheri, A.Y. Zomaya, Y.C. Lee, Linear combinations of DVFS-enabled processor frequencies to modify the energy-aware scheduling algorithms, in: Proc.10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, 2010, pp. 388–397.
[15] N.B. Rizvandi, J. Taheri, A.Y. Zomaya, Some observations on optimal frequency selection in DVFS-based energy consumption minimization, Journal of Parallel and Distributed Computing 71 (8) (2011) 1154–1164.
[16] Y. Shin, K. Chois, T. Sakurait, Power optimization of real-time embedded systems on variable speed processors, in: IEEE/ACM International Conference on Computer Aided Design, 2000, pp. 365–368.
[17] Y.C. Lee, A.Y. Zomaya, On effective slack reclamation in task scheduling for energy reduction, Journal of Information Processing Systems 5 (4) (2009) 175–186.
[18] D. Zhu, R. Melhem, B.R. Childers, Scheduling with dynamic voltage/speed adjustment using slack reclamation in multiprocessor real-time systems, IEEE Transactions on Parallel and Distributed Systems 14 (7) (2003) 686–700.
[19] Embedded Systems. Available: http://cse.spsu.edu/clo/gLab/EmbeddedSystems.htm
[20] C.-O. Lee, M. Lee, D. Han, Energy-efficient location logging for mobile device, in: Proc. 2010 10th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT), 2010, pp. 84–90.
[21] J.F.M. Bernal, L. Ardito, M. Morisio, P. Falcarin, Towards an efficient context-aware system: problems and suggestions to reduce energy consumption in mobile devices, in: Proc. 2010 Ninth International Conference on Mobile Business and 2010 Ninth Global Mobility Roundtable (ICMB-GMR), 2010, pp. 510–514.
[22] L.M. Zhang, K. Li, Y.-Q. Zhang, Green task scheduling algorithms with speeds optimization on heterogeneous cloud servers, in: Proc. of 2010 IEEE/ACM International Conference on Green Computing and Communications (GreenCom2010), Hangzhou, December 18–20, 2010, pp. 76–80.
[23] L.M. Zhang, K. Li, Y.-Q. Zhang, Green task scheduling algorithms with energy reduction on heterogeneous computers, in: Proc. of 2010 International Conference on Progress in Informatics and Computing (PIC-2010), Shanghai, December 10–12, 2010, pp. 560–563.
[24] MATH2640 Introduction to Optimisation: 4. Inequality Constraints, Complementary slackness condition, Maximisation and Minimisation, Kuhn-Tucker method: summary, http://www.maths.leeds.ac.uk/~cajones/math2640/notes4.pdf
[25] Google green: big picture. Available: http://www.google.com/green/



Luna Mingyi Zhang, a John McMullen Dean's Scholar, is a 2nd year undergraduate student studying Computer Science in the College of Engineering at Cornell University. Already, she has 4 paper publications at international conferences; many of them have been cited by at least 10 other researchers. Currently, she serves as a reviewer for the Elsevier journal *Sustainable Computing: Informatics and Systems* and also an editor of *The Research Paper* (Cornell's Undergraduate Research Magazine). At Cornell, she is a member of the Cornell Undergraduate Research Board (CURB), the Cornell Student Section of the Society of Women Engineers (Cornell SWE), and the Association of Computer Science Undergraduates (Cornell ACSU). Furthermore, she was honored as an individual semifinalist in the 2010–11 Siemens Competition in Math, Science & Technology for her research on green computing. She conducted research at Georgia State University for about 3 years (2009–2012) and also completed a 5-month research internship at Southern Polytechnic State University in 2010. Her current research interests include green computing, artificial intelligence, data mining, and optimization; however, she is always open to new fields of interest in computer science. She plans to attend graduate school and hopes to then become a university professor in computer science in addition to encourage more women to pursue professional careers in computer science and engineering.

Keqin Li is a SUNY Distinguished Professor of computer science and an Intellectual Ventures endowed visiting chair professor at the National Laboratory for Information Science and Technology, Tsinghua University, Beijing, China. His research interests are mainly in 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. Dr. Li has published over 245 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 is currently or has served 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.

Dan Chia-Tien Lo received the Ph.D. degree in computer science from Illinois Institute of Technology, Chicago, the M.S. degree in Electrical Engineering from National Taiwan University, Taiwan and the B.S. degree in Applied Mathematics from National Chung-Hsing University, Taiwan in 2001, 1992 and 1990, respectively. In 2002, he joined the department of computer science at University of Texas at San Antonio, San Antonio, Texas, as an assistant professor. After 2009, he has been with Southern Polytechnic State University. He started his teaching career in 1999 and courses he has taught include the Unix Systems Programming, Computer Architecture, Computer Network, Algorithms, and Programming Languages. His research interests include VLSI Design, Operating Systems, Neural Networks, Concurrent Algorithms, Computer Networks, Computer Architecture, and Computing Theory. His current research thrusts are Reconfigurable Computing, Computer Science Education, FPGA-Based Computation, Hardware-Based Network Intrusion Detection, and Low-power Embedded Systems.

Yanqing Zhang is Professor of the Computer Science Department at Georgia State University, Atlanta, USA. He received B.S. and M.S. in computer science from Tianjin University in China in 1983 and 1986, respectively, and Ph.D. in computer science from the University of South Florida in USA in 1997. His research interests include hybrid intelligent systems, green computing, computational intelligence, machine learning, data mining, bioinformatics, health informatics, computational Web intelligence, Yin-Yang computation, nature-inspired computing, security, game theory and cloud computing. He has co-authored two books, co-edited two books and four conference proceedings. He published 18 book chapters, 77 journal papers and 147 conference/workshop papers. He is Managing Editor of *International Journal of Functional Informatics and Personalised Medicine*. He is Program Co-Chair of the 2013 IEEE/ACM/WIC International Conference on Web

Intelligence. He was Program Co-Chair of 2009 International Symposium on Bioinformatics Research and Applications, Program Co-Chair and Bioinformatics Track Chair of IEEE 7th International Conference on Bioinformatics & Bioengineering and Program Co-Chair of 2006 IEEE International Conference on Granular Computing. He received Outstanding Academic Service Award at IEEE 7th International Conference on Bioinformatics & Bioengineering, Achievement Award of the 2007 World Congress in Computer Science, Computer Engineering and Applied Computing, and 2005 IEEE-Granular Computing Outstanding Service Award at 2005 IEEE International Conference on Granular Computing.