J. Parallel Distrib. Comput. 124 (2019) 70–77

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

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

Approximation algorithm for the energy-aware profit maximizing problem in heterogeneous computing systems

Weidong Li [a], Xi Liu [b], Xiaobo Cai [b], Xuejie Zhang [b,*]

[a] School of Mathematics and Statistics, Yunnan University, Kunming, 650504, PR China
[b] School of Information Science and Engineering, Yunnan University, Kunming, 650504, PR China

HIGHLIGHTS

- We design a task-type based for the energy-aware profit maximizing scheduling problem.
- We demonstrate that the approximation ratio is close to 2, which may be the best result.
- Experimental results show that our algorithm can produce a feasible solution with better objective value than the previous algorithm.

ARTICLE INFO

Article history:
Received 2 August 2017
Received in revised form 31 January 2018
Accepted 23 October 2018
Available online 2 November 2018

Keywords:
Task scheduling
High performance computing
Load balancing
Approximation algorithm
Bag-of-tasks

ABSTRACT

Trade-offs between energy and performance are important for energy-aware scheduling. Recently, a novel model, called *energy-aware profit maximizing scheduling problem* (EAPM), which combines energy and makespan into the objective of maximizing the profit per unit of time has been proposed. The user pay a given price to have a bag-of-tasks processed and the objective is to maximize the profit per unit time. In this study, we design a polynomial-time algorithm for the EAPM problem. The execution time of our algorithm is polynomial in the number of task types which is an improvement over the previous algorithm, whose execution time is polynomial in the number of tasks. Moreover, we demonstrate that the approximation ratio of our algorithm is close to 2 for a special case, which may be the best result we can obtain. Experimental results show that our algorithm can produce a feasible solution with better objective value than the previous algorithm.

1. Introduction

With the rapid increase in energy consumption by data centers, energy-efficient resource allocation within high-performance computing systems has become increasingly important. Recently, a static scheduling model has been proposed, where users submit a *bag-of-tasks* [6]. In contrast with previous classic scheduling models, the *estimated time to compute* (or processing time) for a task on a machine depends on the task type and machine type. Instead of allocating all tasks to all machines, this new model determines the number of tasks of each type allocated to machines of each type. Although a high-performance computing system contains hundreds of thousands of machines and the number of tasks is large, the number of machine (or task) types is small. This makes it possible to design a more efficient algorithm for finding near-optimal schedules [23,24].

Classic energy-aware scheduling models aim to minimize either the energy consumed by a bag-of-tasks or the makespan. However, an organization should attempt to maximize profit per time, where the profit is equal to the price that a user pays for the bag-of-tasks minus the cost of electricity consumed by the schedule. By incorporating the energy cost and makespan into the objective of maximizing the profit per unit of time, Tarplee et al. [25] studied a novel scheduling model for high-performance computing system, which has two important characteristics: (a) They are often composed of different types of machines; (b) They perform many tasks, but the number of task types is limited. By using a novel linear programming (LP, for short)-based rounding method, Tarplee et al. [25] designed an efficient algorithm for finding a feasible schedule close to the optimal schedule when every machine has to process a large number of tasks of the same type.

In [25], a lower bound on the finishing times for a machine type is used to replace makespan, which is defined as the maximum finishing time for all machines. Therefore, the proposed mathematical model is "weak". In the LP-based rounding step for the proposed heuristic algorithm, the cost of energy consumed may increase, which can be improved by utilizing a matching-based rounding technique. Moreover, the execution time depends on the number of tasks, which is large in practice. In addition, the approximation ratio of the LP-based rounding algorithm in [25] is not analyzed.

* Corresponding author.
E-mail address: xjzhang@ynu.edu.cn (X. Zhang).

https://doi.org/10.1016/j.jpdc.2018.10.013
0743-7315/

In this paper, we present a polynomial-time algorithm for the problem proposed in [25]. The main contributions of this paper are as follows:

(1) A "tighter" mathematical model is proposed.

(2) A task-type-based algorithm whose execution time in the worst-case is polynomial in the number of task types is proposed for finding a more accurate feasible solution.

(3) The approximation ratio for the task-type-based algorithm presented is analyzed.

The rest of the paper is organized as follows. In Section 2, we summarize relevant literature. In Section 2, we construct the integer program which coincides with the EAPM problem. In Section 4, we present the details of the task-type-based algorithm and analyzes the approximation ratio of the proposed algorithm. In Section 5, we present the experimental results. In the last section, we present conclusions and ideas for future work.

2. Related work

There is a large body of literature on scheduling models that allocate all tasks to all machines in heterogeneous computing systems. Braun et al. [1] compared eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, where the objective is to minimize makespan. Diaz, Pecero, and Bouvry [4] evaluated three scheduling heuristic algorithms for highly heterogeneous distributed computing systems, with the objective of minimizing makespan and flowtime. Assuming that users submit a set of independent tasks (known as a bag-of-tasks), Friese et al. [6,8] developed a modified multi-objective genetic algorithm to create different schedules illustrating the trade-offs between energy consumption and makespan (or the utility earned by a system). Friese et al. [7] created a tool that allows system administrators to study trade-offs between system performance and system energy allocation. Oxley et al. [21] developed and analyzed several heuristics for energy-aware resource allocation for both energy-constrained and deadline-constrained problems. For the scheduling model used to determine the number of tasks of each type to allocate to machines of each type, Tarplee et al. [23] presented a linear programming based resource allocation algorithm to efficiently compute high quality solutions for minimizing makespan. Tarplee et al. [24] designed a linear-programming-based rounding method to generate a set of high-quality solutions that represent the trade-off space between makespan and energy consumption.

In a cloud computing environment, cloud resource management is an important issue for the cloud provider. Ebrahimirad, Goudarzi, and Rajabi [5] presented two scheduling algorithms for precedence-constrained parallel virtual machines (VMs) in a virtualized data center, with the goal of minimizing energy consumption. Tchana et al. [26] proposed a solution for consolidating software onto VMs to reduce power consumption by the private cloud and the number of VMs charged for in a public cloud. Hsu et al. [11] presented an energy-aware task consolidation technique to minimize energy consumption in clouds. Khemka et al. [17] designed four energy-aware resource allocation heuristics for energy-constrained environments, aiming to maximize the total utility earned by the system. Khemka et al. [16] designed several heuristics to maximize the total utility that can be earned by completing tasks, where tasks arrive dynamically and a scheduler maps the tasks to machines for execution. Mashayekhy et al. [19] proposed a framework for improving the energy efficiency of MapReduce applications, aiming to minimize energy consumption given deadline constraints. More relevant results can be found in recent surveys on energy efficient resource scheduling [3,10,15,20].

Recently, Tarplee et al. [25] studied a novel scheduling model which is to maximize the profit per unit of time. They presented an efficient LP-based rounding method to find a feasible solution, which contains two phases: rounds a fraction solution while taking care to maintain feasibility, and assigns tasks to actual machines to build the full task allocation. In this paper, we use different methods in two phases. In the first phase, we use b-matching-based rounding method motivated by [22] with approximate guarantee to substitute the simple rounding method in [25]. In the second phase, we use an intuitive method of assigning the tasks of the same type synchronously to reduce the execution time.

3. Problem modeling and formulation

In a high-performance computing system, there are a great deal of machines which can be divided to several groups and the machines in the same group are identical. Frequently, a user submits a *bag-of-tasks* consisting of several groups of identical tasks. Assume that there are M machine types in the heterogeneous computing system and T task types in the bag-of-tasks. Let $\mathcal{M}_j$ be the set of M_j identical machines of type j. Similarly, let $\mathcal{T}_i$ be the set of T_i identical tasks of type i. As frequently used in scheduling problems for heterogeneous computing systems [1], let $\mathbf{ETC} = (ETC_{ij})$ be a $T \times M$ matrix where ETC_{ij} is the *estimated time to compute* for an i-type task on a j-type machine. Similarly, let $\mathbf{APC} = (APC_{ij})$ be a $T \times M$ matrix where APC_{ij} is the *average power consumption* for a task of type i on a machine of type j. For convenience, assume that all values of APC_{ij} and ETC_{ij} are integers.

For $i = 1, 2, \ldots, T$ and $j = 1, 2, \ldots, M$, Tarplee et al. [25] introduce a decision variable x_{ij} to denote the number of tasks of type i allocated to the machines of type j. However, the process of allocating x_{ij} tasks of type i to the M_j machines of type j may cause uneven balance. Thus, for every task type $i = 1, 2, \ldots, T$ and every machine k of type $j = 1, 2, \ldots, M$ in $\mathcal{M}_j$, we introduce a new variable x_{ijk} to denote the number of tasks of type i allocated to machine $k \in \mathcal{M}_j$, Clearly,

$$x_{ij} = \sum_{k:k\in\mathcal{M}_j} x_{ijk}. \tag{1}$$

For each machine $k \in \mathcal{M}_j$, the finishing time F_{jk} is defined as

$$F_{jk} = \sum_{i=1}^{T} x_{ijk} ETC_{ij}. \tag{2}$$

Correspondingly, for a schedule $\mathbf{x} = (x_{ijk})$, the *makespan* $MS(\mathbf{x})$, which is the maximum finishing time of all machines, is defined as

$$MS(\mathbf{x}) = \max_{j} \max_{k:k\in\mathcal{M}_j} F_{jk}. \tag{3}$$

Since the machines of different types may not finish executing tasks at the same time, the power consumption must be considered for a finished machine that is powered on while an unfinished machine continues to execute tasks. When no task is being executed on a machine $k \in \mathcal{M}_j$, let $APC_{\varnothing j}$ be the average idle power consumption on machine k of type j. From the definitions of APC_{ij} and $APC_{\varnothing j}$, it is reasonable to assume that

$$APC_{\varnothing j} < APC_{ij}, \quad \text{for } i = 1, \ldots, T \text{ and } j = 1, \ldots, M. \tag{4}$$

Thus, the energy consumed by the tasks submitted, incorporating idle power, is defined as:

$$E(\mathbf{x}) = \sum_{j=1}^{M} \sum_{i=1}^{T} x_{ij} APC_{ij} ETC_{ij} + \sum_{j=1}^{M} \sum_{k:k\in\mathcal{M}_j} APC_{\varnothing j}(MS(\mathbf{x}) - F_{jk}), \tag{5}$$

where the second term accounts for the idle power. Let p be the price a user will pay and c be the cost per unit of energy. The profit of the high-performance computing system provider is defined as

$p - cE(\mathbf{x})$. As mentioned in [25], the profit per bag is not solely the quantity that the organization should maximize. The bag-of-tasks can take a considerable amount of time to compute when trying to increase the profit per bag-of-tasks by reducing electricity costs. Instead an organization should attempt to maximize the profit per unit time. Thus, Tarplee et al. [25] proposed the *energy-aware profit maximizing* (EAPM) problem which can be formulated as the following mixed-integer nonlinear programming (MINLP, for short).

$$\begin{cases} Maximize_{\mathbf{x}} \ \dfrac{p - cE(\mathbf{x})}{MS(\mathbf{x})} \\ \displaystyle\sum_{j=1}^{M} \sum_{k:k\in\mathcal{M}_j} x_{ijk} = \sum_{j=1}^{M} x_{ij} = T_i, \forall i; \\ F_{jk} \le MS(\mathbf{x}), \text{for every } k \in \mathcal{M}_j, \forall j; \\ x_{ijk} \in \mathbb{Z}_{\ge 0}, \text{for every } k \in \mathcal{M}_j, \forall i, j. \end{cases} \tag{6}$$

The objective of MINLP (6) is to maximize the profit per unit time. The first constraint of MINLP(6) ensures that all tasks submitted are allocated to the machines exactly once. Because the objective is to maximize profit per unit time (which requires that makespan be minimized), the second constraint of MINLP (6) ensures that $MS(\mathbf{x})$ is exactly the maximum finishing time of machines in the optimal solution.

4. An approximation algorithm

Note that MINLP (6) is a nonlinear integer program, whose optimal solution cannot be found in polynomial time. To obtain an approximate solution of MINLP (6), the most frequently used method contains the following three steps:

(1) Relax MINLP (6) to a linear program (LP);

(2) Solve the LP optimally and obtain a fractional optimal solution by using the interior point method [14];

(3) Round the fractional optimal solution to a feasible integer solution for MINLP (6).

In [25], when $APC_{\varnothing j} = 0$ for all machine types j, the authors obtained a linear program using variable substitution $r \leftarrow 1/MS_{LB}$ and $z_{ij} \leftarrow x_{ij}/MS_{LB}$, where

$$MS_{LB} = \max_j \frac{1}{M_j} \sum_{i=1}^{T} x_{ij} ETC_{ij} \tag{7}$$

is the lower bound on the makespan obtained by allowing tasks to be divided among all machines. However, this approximate method would be inappropriate in a situation where only one tasks of type i with large ETC_{ij} values is allocated to the machines of type j. The experimental results in [24] also verify this point.

To obtain better solution in worst-case scenario, we utilize a different LP-rounding method, which contains the following four steps:

(1) For a given constant MS, we construct an approximate integer linear program (ILP) for MINLP (6) by setting $MS(\mathbf{x}) = MS$;

(2) Relax ILP to obtain a linear program (LP);

(3) Solve the LP optimally and obtain a fractional optimal solution by using the interior point method [14];

(4) Round the fractional optimal solution to a feasible integer solution for MINLP (6), by using the matching based rounding method for the generalized assignment problem [22].

In the above method, if the tasks of type i satisfy that $ETC_{ij} > MS$, all the tasks of type i will not be allocated to machines of type j in the feasible solution; this is done to avoid increasing the maximum finishing time too much in the matching based rounding process. Another advantage of our method is that it can be used to obtain a feasible solution for an EAPM problem with zero profit.

For any decision variable $\mathbf{x}$, it is easy to verify that $MS(\mathbf{x})$ is not lower than the lower bound $LB = 1$ and not higher than the upper bound $UB = \sum_{i=1}^{T} x_{ij} \max_j ETC_{ij}$, which is obtained by allocating all tasks to the machine with the largest estimated time for computing ETC_{ij}. For any given constant $\epsilon > 0$, let $MS_t = LB(1+\epsilon)^t$, where $t \in \{1, 2, \ldots, \lceil \log_{(1+\epsilon)} UB/LB \rceil\}$. Clearly, the makespan $MS(\mathbf{x}^*)$ of the optimal solution $\mathbf{x}^*$ for MINLP (6) belongs the interval $[LB(1+\epsilon)^t, LB(1+\epsilon)^{t+1}]$, for some t. Because $\{\lceil \log_{(1+\epsilon)} UB/LB \rceil\}$ is polynomial in the input length, we can find a feasible solution with makespan MS satisfy that $MS(\mathbf{x}^*) \in [MS/(1+\epsilon), MS]$ by trying all possible values, where $MS = LB(1+\epsilon)^t$ for some t. Thus, we can take MS as a constant such that

$$MS(\mathbf{x}^*) \le MS \le (1+\epsilon) MS(\mathbf{x}^*). \tag{8}$$

As in [25], for any given constant MS, our method is decomposed into two phases. In phase one, we get a schedule such that x_{ij} is computed by rounding the fraction optimal solution. In the phase two, we obtain the feasible solution tasks x_{ijk} by allocating tasks to actual machines. The details of the proposed method are described in the following subsections.

4.1. Rounding

If $ETC_{ij} > MS$, as mentioned above, tasks of type i cannot be allocated to machines of type j in the optimal solution. This implies that $x_{ijk} = x_{ij} = 0$, for all i, j, k such that $k \in \mathcal{M}_j$ and $ETC_{ij} > MS$. Assuming that the tasks can be divided among all machines and that all tasks allocated to a machine of type j finish at the same time, Tarplee et al. [25] found a lower bound F_j on the finishing time of a machine of type j, which is given by

$$F_j = \frac{1}{M_j} \sum_{i=1}^{T} x_{ij} ETC_{ij}. \tag{9}$$

Thus, an obvious lower bound for the makespan is $\max_j F_j$, which implies that

$$\max_j F_j \le MS. \tag{10}$$

Because MS is approximately equal to $MS(\mathbf{x}^*)$, we can replace $MS(\mathbf{x})$ by MS in MINLP (6). Because p, MS, c are fixed constants, the objective in MINLP (6) is equivalent to minimize

$$E(\mathbf{x}) = \sum_{j=1}^{M} \sum_{i=1}^{T} x_{ij} APC_{ij} ETC_{ij} + \sum_{j=1}^{M} M_j APC_{\varnothing j} (MS - F_j) \tag{11}$$

$$= \sum_{j=1}^{M} \sum_{i=1}^{T} x_{ij} (APC_{ij} - APC_{\varnothing j}) ETC_{ij} + \sum_{j=1}^{M} M_j APC_{\varnothing j} MS, \tag{12}$$

where the last equation follows from (9).

Because $\sum_{j=1}^{M} M_j APC_{\varnothing j} MS$ is known, minimizing $E(\mathbf{x})$ is equivalent to minimizing the first term of $E(\mathbf{x})$ denoted by $E_{partial}(\mathbf{x})$. Therefore, we get an approximate equivalent integer program (AEIP) for MINLP (6):

$$\begin{cases} Minimize_{\mathbf{x}} \ E_{partial}(\mathbf{x}) = \displaystyle\sum_{j=1}^{M} \sum_{i=1}^{T} x_{ij} (APC_{ij} - APC_{\varnothing j}) ETC_{ij} \\ \displaystyle\sum_{j=1}^{M} x_{ij} = T_i, \forall i \\ \dfrac{1}{M_j} \displaystyle\sum_{i=1}^{T} x_{ij} ETC_{ij} \le MS, \forall j \\ x_{ij} \in \mathbb{Z}_{\ge 0}, \forall i, j \\ x_{ij} = 0, \text{if } ETC_{ij} > MS. \end{cases} \tag{13}$$

Theorem 1. *A feasible solution exists for AEIP (13).*

Proof. Given an optimal solution $\mathbf{x}^*$ for MINLP (6), let $x_{ij} = \sum_{k:k\in\mathcal{M}_j} x^*_{ijk}$. If $x^*_{ijk} > 0$ for some i, j, k, we have $ETC_{ij} \le MS(\mathbf{x}^*) \le MS$, following from (8). This implies that $x_{ij} = 0$, if $ETC_{ij} > MS$. Moreover,

$$\sum_{j=1}^{M} x_{ij} = \sum_{j=1}^{M} \sum_{k\in\mathcal{M}_j} x^*_{ijk} = T_i, \tag{14}$$

and

$$\frac{1}{M_j}\sum_{i=1}^{T} x_{ij}ETC_{ij} = \frac{1}{M_j}\sum_{i=1}^{T}\sum_{k:k\in\mathcal{M}_j} x^*_{ijk}ETC_{ij} \le \max_{j,k}\sum_{i=1}^{T} x^*_{ijk}ETC_{ij} = MS(\mathbf{x}^*) \le MS. \tag{15}$$

In a word, $\mathbf{x}^*$ is a feasible solution for AEIP (13). ■

Next, we will find a feasible solution for AEIP (13). Replacing constraint $x_{ij} \in \mathbb{Z}\ge 0$ by $x_{ij} \ge 0$, we get a relaxation of AEIP (13), which can be solved optimally in polynomial time by using the interior point method [14]. Let $\tilde{\mathbf{x}} = (\tilde{x}_{ijk})$ be the optimal solution for the relaxation of AEIP (13). Based on the main idea of the matching-based rounding technique in [22], we can obtain an integer solution $\hat{\mathbf{x}}$ for AEIP (13). Through the process of the algorithm, following from [22], we have

$$E_{partial}(\hat{\mathbf{x}}) \le E_{partial}(\tilde{\mathbf{x}}) \le E_{partial}(\mathbf{x}^*).$$

However, the execution time of the matched-based rounding algorithm [22] is polynomial in the number of tasks and number of machines, which are very large in reality. To design a more efficient algorithm, we replace matching with *b-matching* [12], and propose a new efficient algorithm whose execution time is polynomial in the number of task types and number of machine types, where the b-matching of a given bipartite graph $B = (U \cup V, E, b)$ with $b : U \to Z^+ \cup \{0\}$ is a subset of E that exactly matches b_i instances of the vertex u_i in E. For completeness, we present modified b-matching-based rounding method from [22] as follows. For simplicity, we describe the details of constructing the auxiliary bipartite graph $B(\tilde{\mathbf{x}})$ and the weights of edges, where the fraction solution of matching is ignored. Given an optimal fraction solution $\tilde{\mathbf{x}}$, for $i = 1, \ldots, T$ and $j = 1, \ldots, M$, let

$$x'_{ij} = \tilde{x}_{ij} - \lfloor \tilde{x}_{ij} \rfloor, \text{ and } k_j = \lceil \sum_{i=1}^{T} x'_{ij} \rceil.$$

The auxiliary bipartite $B(\tilde{\mathbf{x}}) = (U, V, E; w)$ graph constructed as follows, where the node set $U = \{u_1, \ldots, u_T\}$ represents the task-type set, and V consists of *machine-type* nodes

$$V = \{v_{js} | j = 1, \ldots, M, s = 1, \ldots, k_j\}. \tag{16}$$

Here, for $j = 1, \ldots, M$, k_j nodes $\{v_{js} | s = 1, \ldots, k_j\}$ represent the machine types j.

Following the matching based method in [22], for $i = 1, \ldots, T$ and $j = 1, \ldots, M$, if $x'_{ij} > 0$, we construct the edges of type (u_i, v_{js}) of the auxiliary graph $B(\tilde{\mathbf{x}})$ as follows. For $j = 1, \ldots, M$, without loss of generality, assume that

$$ETC_{1j} \ge ETC_{2j} \ge \cdots \ge ETC_{Tj}. \tag{17}$$

For $j = 1, \ldots, M$, we distinguish the following two cases.

Case 1. $\sum_{i=1}^{T} x'_{ij} \le 1$. By the definition of k_j, we have $k_j = 1$. For $i = 1, \ldots, T$ such that $x'_{ij} > 0$, put edge (v_{i1}, u_j) into E.

Case 2. $\sum_{i=1}^{T} x'_{ij} > 1$. Find the minimum index i_1 such that $\sum_{i=1}^{i_1} x'_{ij} \ge 1$. For $i = 1, \ldots, i_1$, if $x'_{ij} > 0$, put the edge (v_{j1}, u_i) into E. Similarly, for $s = 2, \ldots, k_j - 1$, let i_s be the minimum integer such that $\sum_{i=1}^{i_s} x'_{ij} \ge s$. For $i = i_{s-1} + 1, \ldots, i_s$, if $x'_{ij} > 0$, put the edge (v_{js}, u_i) into E. If $\sum_{i=1}^{i_s} x'_{ij} > s$, put edge $(v_{j,s+1}, u_{i_s}) \in E$. Finally, for $i = i_{k_j-1} + 1, \ldots, T$, if $x'_{ij} > 0$, put the edges (v_{jk_j}, u_i) into E.

For each edge $(v_{js}, u_i) \in E$, let

$$w(v_{is}, u_j) = (APC_{ij} - APC_{\varnothing j})ETC_{ij}$$

the weight of edge (v_{js}, u_i). For every task-type vertex $u_i \in U$, let $b_i = \sum_{j=1}^{M} x'_{ij}$ be the capacity of u_i. Clearly, b_i is an integer as

$$\sum_{j=1}^{M} x'_{ij} = \sum_{j=1}^{M} \tilde{x}_{ij} - \sum_{j=1}^{M} \lfloor \tilde{x}_{ij} \rfloor = T_i - \sum_{j=1}^{M} \lfloor \tilde{x}_{ij} \rfloor$$

is an integer. From the above process of constructing the auxiliary bipartite graph $B(\tilde{\mathbf{x}})$, there are at most $\sum_{j=1}^{M} k_j \le MT$ vertices in V and $T + M$ constraints in AEIP (13). Thus, by the properties of linear program, the number of positive variables in the optimal solution $\tilde{\mathbf{x}}$ is no more than $T + M$. For each $x'_{ij} > 0$, from the above construction of edges of $B(\tilde{\mathbf{x}})$, there are two edges corresponding to $x'_{ij} > 0$ in E, implying that the number of edges in E is no more than $2(T + M)$ edges in E. Thus, by utilizing the polynomial-time algorithm in [12], whose execution time is determined by T and M, we obtain the optimal *b-matching* $\mathcal{BM}$, which exactly matches b_i instances of the vertex u_i in E.

The modified b-matching based rounding method used to construct a schedule from a feasible solution $\tilde{\mathbf{x}}$ is presented as follows.

Algorithm 1: Modified b-matching based rounding method.

1: Form the bipartite graph $B(\tilde{\mathbf{x}})$ with weights on its edges.
2: Find a minimum-weight (integer) b-matching $\mathcal{BM}$ that exactly matches b_i instances of the task-type node u_i in $B(\tilde{\mathbf{x}})$.
3: For each edge $(v_{js}, u_i) \in \mathcal{BM}$, allocate an i-type task to a j-type machine. Output the resulting schedule $\hat{\mathbf{x}}$, where $\hat{x}_{ij}$ is the number of i-type tasks allocated to the j-type machines.

Note that $\sum_{i=1}^{T} \hat{x}_{ij} \le \sum_{i=1}^{T} \tilde{x}_{ij} + 1$ for every machine type j, following the definitions of x'_{ij} and k_j. It is proved in [22] that the above matching-based rounding method can produce a schedule with makespan increasing at most the maximum value of ETC_{ij} and without increasing the cost. Similarly, we have

Theorem 2. *The schedule $\hat{\mathbf{x}}$ obtained by Algorithm 1 satisfying $E_{partial}(\hat{\mathbf{x}}) \le E_{partial}(\tilde{\mathbf{x}})$, and*

$$\frac{1}{M_j}\sum_{i=1}^{T} \hat{x}_{ij}ETC_{ij} \le \frac{1}{M_j}\left(\sum_{i=1}^{T} \tilde{x}_{ij}ETC_{ij} + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}\right) \le MS + \frac{1}{M_j}\max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}. \tag{18}$$

4.2. Local assignment

In the last subsection, we describe how to compute the number of i-type tasks to the j-type machines. However, a feasible solution for the EAPM problem is to compute the number of tasks of every type allocated to each machine. Clearly, for each machine type $j = 1, 2, \ldots, M$, allocating $\hat{x}_{ij}$ tasks to machines of type j will not change the value of $E_{partial}$. Combining the (6) and (13), our objective is to minimize makespan. Since ETC_{ij} are constant when allocate $\hat{x}_{ij}$ tasks to the machines of type j, this problem is exactly the parallel machine scheduling problem [9], where the proposed longest processing time (LPT) algorithm is to allocate every task to the machine that will complete the task at the earliest.

As shown in [25], the effect of LPT sub-optimality on the overall performance of the considered systems is insignificant, as the number of tasks is generally large. This leads to another problem, where the execution time for the LPT algorithm increases dramatically as the number of tasks rapidly increases. Note that in the HPC system, the number of task types (or machines) is always less than the number of tasks. For example, in the simulations of [25], there are nine machine types, 30 task types, and 11,000 tasks. An important observation is that we do not need to allocate one task at a time when allocating tasks of same type. For example, when we allocate the first 3000 i-type tasks on the sorted list to 10 j-type machines, because the tasks are identical, we can allocate 300 tasks to each machine. This motivates us to design a fast polynomial-time algorithm based on the task types.

For each machine type $j = 1, 2, \ldots, M$, we allocate the tasks to the M_j machines of type j independently. As before, sort the task types in descending order according to the execution time ETC_{ij}. For every machine $k \in \mathcal{M}_j$, define the *load* as the sum of ETC_{ij} of the tasks allocated to it. Correspondingly, define $L(i, k)$ as the load of machine k after allocating i-type tasks. For every machine $k \in \mathcal{M}_j$, set $L(0, k) = 0$. For $i = 1, 2, \ldots, T$, let

$$AL_i = \frac{\sum_{k\in\mathcal{M}_j} L(i,k)}{M_j} = \frac{\sum_{k\in\mathcal{M}_j} L(i-1,k) + ETC_{ij}\hat{x}_{ij}}{M_j}. \tag{19}$$

be the average load of the M_j j-type machines after allocating the first i-type tasks.

For each machine $k \in \mathcal{M}_j$, assuming that there are $N_{unallocated}$ unallocated tasks, simultaneously schedule $\min\{N_{unallocated}, N_{ik}\}$ i-type tasks to machine k, where

$$N_{ik} = \max\{\lfloor \frac{AL_i - L(i-1,k)}{ETC_{ij}} \rfloor, 0\}. \tag{20}$$

If $N_{ik} > 0$, following from the definition of N_{ik}, we have

$$AL_i - ETC_{ij} < L(i,k) = L(i-1,k) + N_{ik}ETC_{ij} \le AL_i. \tag{21}$$

It implies that the number of remaining i-type tasks is no more than M_j, which can be allocated by utilizing the LPT algorithm [9]. In fact, this method produces the same solution as Algorithm 2 in [25]. However, the execution time in the worst case is $O(\sum_{j=1}^{M}(T \log T + TM_j))$ and is determined by the number of task types, which is very small in most cases.

Algorithm 2: Local Assignment.

1: **For** $j = 1$ to M do
2: Relabel the indices of task types such that $ETC_{1j} \ge \cdots \ge ETC_{Tj}$;
3: **For** $i = 1$ to T do
4: **For** each machine $k \in \mathcal{M}_j$ do
5: Allocate N_{ik} (defined in (19)) tasks of type i to it;
6: **End for**
7: Use LPT algorithm to allocate the remaining tasks of type j (at most M_j);
8: **End for**
9: **End for**

Let $\bar{\mathbf{x}} = (\bar{x}_{ijk})$ be the resulting solution, where $\bar{x}_{ijk}$ is the number of i-type tasks allocated to machine $k \in \mathcal{M}_j$. Obviously, $\sum_{k:k\in\mathcal{M}_j} \bar{x}_{ijk} = \hat{x}_{ij}$ for all i, j. Thus,

Lemma 1. *The makespan of $\bar{\mathbf{x}}$ is at most $MS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}$, where MS is defined in (8).*

Proof. For convenience, let $\bar{F}_{jk} = \sum_{i=1}^{T} \bar{x}_{ijk}ETC_{ij}$ be the completion time for machine $k \in \mathcal{M}_j$, which implies that $\max_{j,k} \bar{F}_{jk}$ is the makespan of $\bar{\mathbf{x}}$. For every machine type j, if there is a machine $k \in \mathcal{M}_j$ such that $\bar{F}_{jk} > MS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}$, then the completion times for all machines in $\mathcal{M}_j$ are at least MS, following from the choice of LPT. Therefore,

$$\begin{aligned} M_jMS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij} &< \sum_{k:k\in\mathcal{M}_j} \bar{F}_{jk} = \sum_{i=1}^{T}\sum_{k:k\in\mathcal{M}_j} \bar{x}_{ijk}ETC_{ij} \\ &= \sum_{i=1}^{T} \hat{x}_{ij}ETC_{ij} && (22)\\ &\le \sum_{i=1}^{T} \tilde{x}_{ij}ETC_{ij} + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij} && (23)\\ &\le M_jMS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}, && (24) \end{aligned}$$

where the second inequality follows from (18), and the last inequality follows from the constraints in AEIP (13). A contradiction. Thus, for every machine $k \in \mathcal{M}_j$, we have $\bar{F}_{jk} \le MS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}$. The theorem holds. ■

Theorem 3. *When $APC_{\varnothing j} = 0$ for all j and $OPT > 0$, the objective value of feasible solution $\bar{\mathbf{x}}$ is no less than $\frac{1}{2+2\epsilon}OPT$, where OPT is the objective value of $\mathbf{x}^*$ for MINLP (6).*

Proof. Clearly, $E_{partial}(\bar{\mathbf{x}}) = E_{partial}(\hat{\mathbf{x}})$, as $\bar{x}_{ij} = \sum_{k:k\in\mathcal{M}_j} \bar{x}_{ijk} = \hat{x}_{ij}$ for all i, j. Because the makespan of $\bar{\mathbf{x}}$ is increased, the energy consumed by idleness is increased. Following from Theorem 2 and (12), we have

$$\begin{aligned} E(\bar{\mathbf{x}}) &\le E_{partial}(\bar{\mathbf{x}}) + \sum_{j=1}^{M} M_jAPC_{\varnothing j}(MS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}) && (25)\\ &\le E_{partial}(\mathbf{x}^*) + \sum_{j=1}^{M} M_jAPC_{\varnothing j}(MS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}) && (26)\\ &\le E(\mathbf{x}^*) + \sum_{j=1}^{M} M_jAPC_{\varnothing j}(\epsilon MS(x^*) + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}), && (27) \end{aligned}$$

where the second inequality follows from the fact that $\tilde{\mathbf{x}}$ is an optimal fraction solution while $\mathbf{x}^*$ is a feasible solution for AEIP (13), and the last inequality follows from (8). Following from Lemma 1, the objective value of $\bar{\mathbf{x}}$ is no less than

$$\begin{aligned} &\frac{p - cE(\bar{\mathbf{x}})}{MS + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}} && (28)\\ \ge\ &\frac{p - cE(\mathbf{x}^*) - c\sum_{j=1}^{M} M_jAPC_{\varnothing j}(\epsilon MS(x^*) + \max_{i,j:\tilde{x}_{ij}>0} ETC_{ij})}{2MS} && (29)\\ =\ &\frac{p - cE(\mathbf{x}^*)}{2MS} - \frac{c}{2}\left(\frac{\epsilon MS(\mathbf{x}^*)}{MS} + \frac{\max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}}{MS}\right)\sum_{j=1}^{M} M_jAPC_{\varnothing j} && (30)\\ \ge\ &\frac{p - cE(\mathbf{x}^*)}{2(1+\epsilon)MS(\mathbf{x}^*)} - \frac{c}{2}\left(\frac{\epsilon}{1+\epsilon} + \frac{\max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}}{(1+\epsilon)MS(\mathbf{x}^*)}\right)\sum_{j=1}^{M} M_jAPC_{\varnothing j} && (31)\\ =\ &\frac{1}{2+2\epsilon}OPT - \frac{c}{2}\left(\frac{\epsilon}{1+\epsilon} + \frac{\max_{i,j:\tilde{x}_{ij}>0} ETC_{ij}}{(1+\epsilon)MS(\mathbf{x}^*)}\right)\sum_{j=1}^{M} M_jAPC_{\varnothing j}, && (32) \end{aligned}$$

where the first inequality follows from (27) and the last constraint in AEIP (13), and the second inequality follows from (8). When $APC_{\varnothing j} = 0$ for all j, the approximation ratio of the above algorithm is $2 + 2\epsilon$, for any $\epsilon > 0$. ■

In summary, for each $MS_t = (1+\epsilon)^t$, $t = 1, \ldots, \lceil \log_{(1+\epsilon)} UB \rceil$, solve the relaxation of AEIP(13) with $MS = MS_t$ to find an optimal

Table 1
Objective values, with γ varied from 1.05 to 1.5.

$\gamma=$	1.05	1.1	1.15	1.12	1.25	1.3	1.35	1.4	1.45	1.5
TMS	−2301.59	−1808.88	−1316.18	−351.97	751.23	2334.70	5174.07	7460.17	9847.61	12235.05
TTB	−2301.59	−1808.88	−1226.64	−347.98	757.78	2341.78	5192.13	7480.57	9868.02	12255.46

fraction solution $\tilde{\mathbf{x}}$. Then, utilize the algorithm described above to compute a feasible solution for MINLP (6). Finally, select the best solution among these $\lceil \log_{(1+\epsilon)} UB \rceil$ feasible solutions. Clearly, the objective value is at least $\frac{OPT}{2+2\epsilon}$ when $APC_{\varnothing j} = 0$ for all j. This implies that we have found a $(2 + 2\epsilon)$-approximation solution.

5. Simulation results

For convenience, the method in [25] and the method we proposed are called Tarplee, Maciejewski, and Siegelet (TMS) method and task-type-based (TTB) method, respectively. We did extensive simulation experiments to compare two method, where the software was written in C++ and the LP solver used the simplex method from COIN-OR CLP [2]. As in [25], we considered 9 machine types and 40 machines of each type, for a total of 360 machines for every simulations.

Without the loss of generality, as in [25], assume that $c = 1$ for all experiments. Define E_{min} as the lower bound on the energy consumed, ignoring makespan'. Clearly, $E_{min} = \sum_{i=1}^{n} \min_j APC_{ij}$, which is obtained by allocating each task to the machine with the minimum average power consumption. To model very-low-power sleep states, the idle power of a machine of type j is set to 5% of the average power consumption of each machine, as in [25]. Therefore,

$$APC_{\emptyset j} = \frac{0.05}{T} \sum_{i=1}^{T} APC_{ij}. \tag{33}$$

For convenience, let $p = \gamma E_{min}$, where $\gamma = p/E_{min}$ is a parameter which is to affect the price per bag. In this section, a service level agreement is not in place, which implies that the organization will choose not to process any tasks when $\gamma \leq 1$. On the other hand, when γ is large enough ($\gamma > 1.5$, as noted in [25]), the maximum profit solution forces the makespan to be minimized. For this case, our TTB method finds a solution whose objective value is very close to that of the solution produced by the TMS method. Therefore, only $\gamma \in (1, 1.5]$ is considered in this section.

5.1. Different values of γ

The first experiment is executed on a benchmark [13] ignoring the missing values. The workload contains 11,000 tasks divided equally among 5 task types. The feasible solutions produced by two methods for variations in γ are presented in Table 1. Table 1 shows that each feasible solution produced by the TTB method is no worse than that produced by the TMS method. In particular, when γ is small, the objective value of any feasible solution is negative. Two methods will find the solution such that the energy cost is minimized. When γ is large, the TTB method produces a better solution, because the rounding process in the TMS method will increase the energy consumption greatly.

5.2. Different values of the bag size n

As observed by [25], the accuracy of the maximum profit solution produced by the TMS method improves as the bag size n increases. To evaluate the effect of number of tasks in situations where n is increasing, we compare the abovementioned methods with varying n for $\gamma = 1.2$, 1.3 and 1.5, respectively, where there are 360 machines and 5 task types, as before.'

Fig. 1 shows that when the bag size n is sufficiently large, the objective values of the solutions produced by the TTB and TMS methods are almost equal, which implies that the accuracy of the solution produced by the TTB method has also been improved, as the TMS method can produce a near-optimal solution when n is sufficiently large which is observed in [25]. The main reason is that the makespans of the schedules produced by two methods are almost equal to the makespan of optimal schedule when the bag size n is sufficiently large.

The expected execution time for solving a linear program is determined by the number of machines and task types. However, the execution time of a local assignment algorithm in the TMS method depends on the number of tasks. In contrast, the execution time of the local assignment algorithm in the TTB method depends on the number of task types. To evaluate the execution time of the local assignment algorithm, we compare two methods mentioned for varying values of n for $\gamma = 1.2$, 1.3 and 1.5,

Fig. 2 shows that when the number of tasks increases, the execution time of the local assignment algorithm in the TMS method increases approximately linearly, while the execution time of the local assignment algorithm in the TTB method changes slightly.

5.3. Heterogeneous values of ETC_{ij} and APC_{ij}

Because ETC_{ij} and APC_{ij} are not dissimilar in the benchmark [13], we perform five extra experiments where ETC_{ij} and APC_{ij} are random numbers uniformly from [0, 1]. For $q = 1, \ldots, 5$, the workload consists of $360q$ tasks divided equally among 5 task types in the qth experiment. To eliminate the influence of random numbers, we averaged over 100 such simulations to obtain data points.

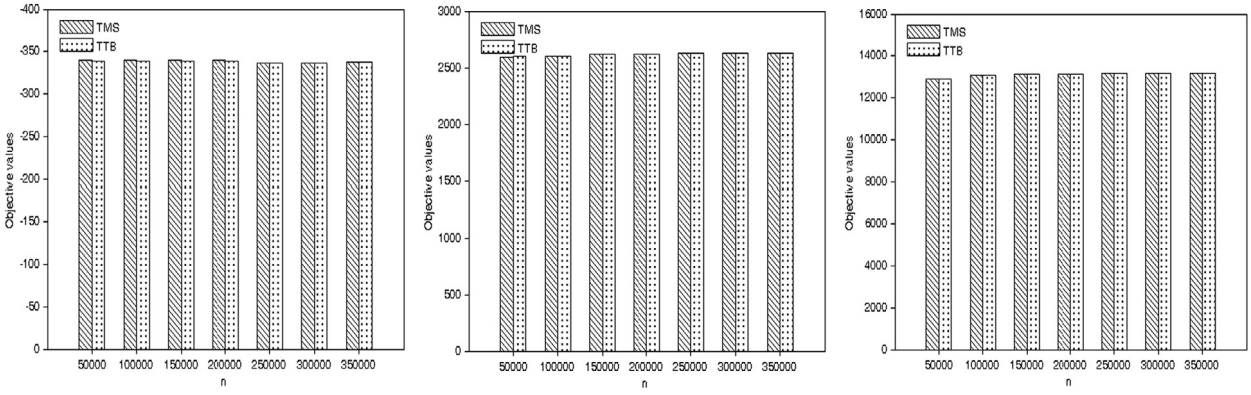
Fig. 3 shows the objective value of the solution computed by two methods for $\gamma = 1.2$, 1.3 and 1.5, where each solution computed by the TTB method has a higher objective value. In particular, when the optimal value is close to zero, the TTB method performs quite well.

Actually, for each experiment we have performed where γ is also a random number, the TTB method produces a better solution. Moreover, when the optimal value is close to zero or the average number of tasks per machine is smaller, the TTB method always produce a better solution.

6. Conclusion and future work

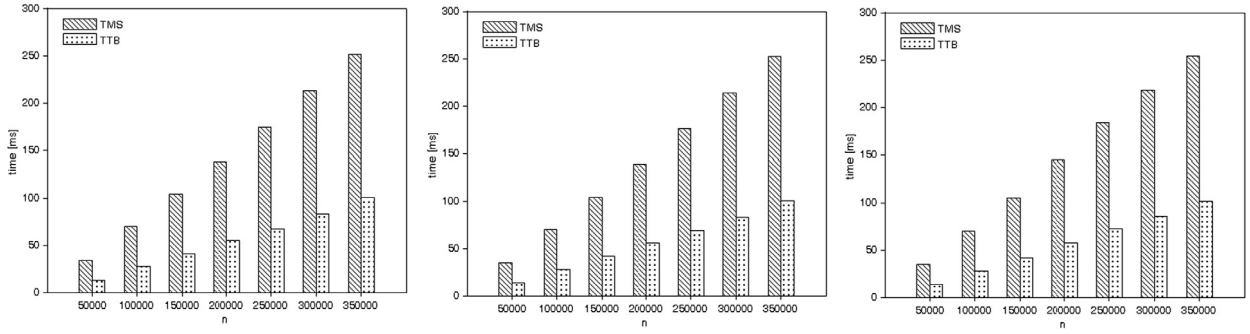
We proposed a polynomial-time algorithm, called the TTB method, whose execution time depends on the amount of task types. Most importantly, we proved that the approximation ratio of the TTB method is close to 2 when the idle power consumption of every machine is zero. Although the experiments showed that the TTB method always produce a near-optimal solution, this may not be true in the worst-case scenario. Designing an efficient algorithm with approximation ratio no more than 2 for the case of arbitrary power consumptions would be interesting and challenging.

In addition, both the methods studied in this paper must solve linear programs, which are not combinatorial. It is interesting to design a fast combinatorial algorithm that can find a near-optimal solution without solving the linear program.



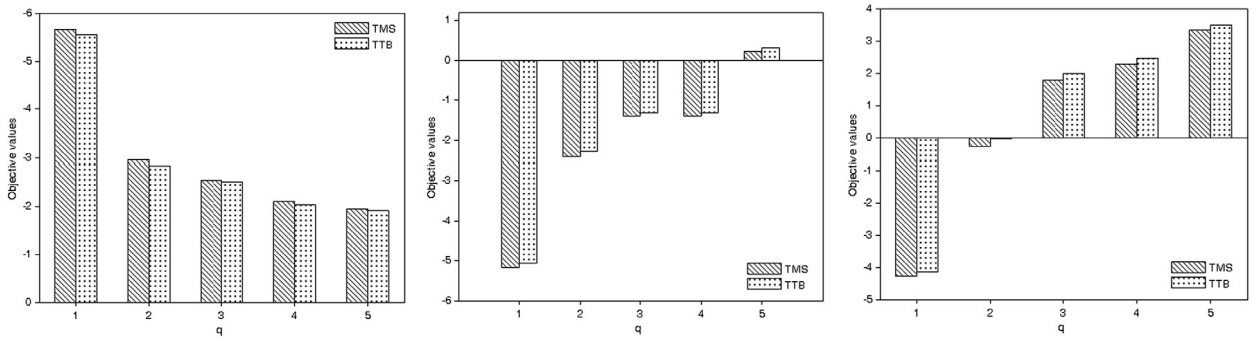
(a) $\gamma = 1.2$ (b) $\gamma = 1.3$ (c) $\gamma = 1.5$

Fig. 1. Objective values of two methods with varying n.



(a) $\gamma = 1.2$ (b) $\gamma = 1.3$ (c) $\gamma = 1.5$

Fig. 2. Execution times of two local assignment algorithms with varying n.



(a) $\gamma = 1.2$ (b) $\gamma = 1.3$ (c) $\gamma = 1.5$

Fig. 3. Objective values of the two methods with randomized data.

Acknowledgments

We are grateful to the anonymous referees for numerous helpful comments and suggestions which helped to improve the presentation of our work. Partial of the results in this paper appeared as a poster in the proceeding of IEEE/ACM CCGrid [18]. The work is supported in part by the National Natural Science Foundation of China [Nos. 61662088, 61762091, 11301466], the Natural Science Foundation of Yunnan Province of China [No. 2014FB114], the Program for Excellent Young Talents, Yunnan University, and IRTSTYN.

References

[1] T.D. Braun, H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, B. Yao, D. Hensgen, R.F. Freund, A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, J. Parallel Distrib. Comput. 61 (6) (2001) 810–837.

[2] Coin-or clp. 2013, March, [Online]. Available: https://projects.coinor.org/Clp.

[3] M. Dayarathna, Y. Wen, R. Fa, Data center energy consumption modeling: A survey, IEEE Commun. Surv. Tutor. 18 (1) (2016) 732–794.

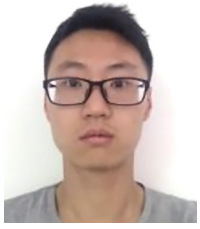
[4] C.O. Diaz, J.E. Pecero, P. Bouvry, Scalable, low. complexity, low complexity and fast greedy scheduling heuristics for highly heterogeneous distributed computing systems, J. Supercomput. 67 (3) (2014) 837–853.

[5] V. Ebrahimirad, M. Goudarzi, A. Rajabi, Energy-aware scheduling for precedence-constrained parallel virtual machines in virtualized data centers, J. Grid Comput. 13 (2) (2015) 233–253.

[6] R. Friese, T. Brinks, C. Oliver, H.J. Siegel, A.A. Maciejewski, Analyzing the trade-offs between minimizing makespan and minimizing energy consumption in a heterogeneous resource allocation problem, in: The Second International Conference on Advanced Communications and Computation, 2012, pp. 81–89.

[7] R. Friese, T. Brinks, C. Oliver, H.J. Siegel, A.A. Maciejewski, S. Pasricha, A machine-by-machine analysis of a bi-objective resource allocation problem, in: International Conference on Parallel and Distributed Processing Technologies and Applications, 2013, pp. 3–9.
[8] R. Friese, B. Khemka, A.A. Maciejewski, H.J. Siegel, G.A. Koenig, S. Powers, M. Hilton, J. Rambharos, G. Okonski, S.W. Poole, An analysis framework for investigating the trade-offs between system performance and energy consumption in a heterogeneous computing environment, in: IEEE 27th International Parallel and Distributed Processing Symposium Workshops, 2013, pp. 19–30.
[9] R. Graham, Bounds on multiprocessing timing anomalies, SIAM J. Appl. Math. 17 (2) (1969) 416–429.
[10] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P.h Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q.M. Malluhi, N. Tziritas, A. Vishnu, S.U. Khan, A. Zomaya, A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems, Computing 98 (2016) 751–774.
[11] C.H. Hsu, K.D. Slagter, S.C. Chen, Y.C. Chung, Optimizing energy consumption with task consolidation in clouds, Inform. Sci. 258 (10) (2014) 452–462.
[12] B.C. Huang, T. Jebara, Fast b-matching via sufficient selection belief propagation, J. Mach. Learn. Res. - Proc. Track 15 (2011) 361–369.
[13] Intel core i7 3770k power consumption, thermal. 2013, May, [Online]. Available: http://openbenchmarking.org/result/1204229-SU-CPUMONITO81.
[14] N. Karmarkar, A new polynomial-time algorithm for linear programming, in: Proceedings of the sixteenth annual ACM symposium on Theory of computing, STOC, 1984, pp. 302–311.
[15] T. Kaur, I. Chana, Energy efficiency techniques in cloud computing: A survey and taxonomy, ACM Comput. Surv. 48 (2) (2015) 22.
[16] B. Khemka, R. Friese, L.D. Briceno, H.J. Siegel, A.A. Maciejewski, G.A. Koenig, C. Groer, G. Okonski, M.M. Hilton, R. Rambharos, S. Poole, Utility functions and resource management in an oversubscribed heterogeneous computing environment, IEEE Trans. Comput. 64 (1) (2015) 2394–2407.
[17] B. Khemka, R. Friese, S. Pasricha, A.A. Maciejewski, H.J. Siegel, G.A. Koenig, S. Powers, M. Hilton, R. Rambharos, S. Poole, Utility maximizing dynamic resource management in an over subscribed energy-constrained heterogeneous computing system, Sustain. Comput.: Inform. Syst. 5 (2015) 14–30.
[18] W. Li, X. Liu, X. Zhang, X. Cai, A task-type-based algorithm for the energy-aware profit maximizing scheduling problem in heterogeneous computing systems, in: 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2015, pp. 1107–1110.
[19] L. Mashayekhy, M.M. Nejad, D. Grosu, Q. Zhang, W. Shi, Energy-aware scheduling of mapreduce jobs for big data applications, IEEE Trans. Parallel Distrib. Syst. 26 (10) (2015) 2720–2733.
[20] A.-C. Orgerie, M.D. de Assuncao, L. Lefevre, A survey on techniques for improving the energy efficiency of large-scale distributed systems, ACM Comput. Surv. 46 (4) (2014) 47.
[21] M.A. Oxley, S. Pasricha, A.A. Maciejewski, H.J. Siegel, J. Apodaca, D. Young, L. Briceno, J. Smith, S. Bahirat, B. Khemka, A. Ramirez, Y. Zou, Makespan and energy robust stochastic static resource allocation of bags-of-tasks to a heterogeneous computing system, IEEE Trans. Parallel Distrib. Syst. 26 (10) (2015) 2791–2805.
[22] D.B. Shmoys, E. Tardos, An approximation algorithm for the generalized assignment problem, Math. Program. 62 (1–3) (1993) 461–474.
[23] K.M. Tarplee, R. Friese, A.A. Maciejewski, H.J. Siegel, Scalable linear programming based resource allocation for makespan minimization in heterogeneous computing systems, J. Parallel Distrib. Comput. 84 (2015) 76–86, Previous version in;
K.M. Tarplee, R. Friese, A.A. Maciejewski, H.J. Siegel, Efficient and scalable pareto front generation for energy and makespan in heterogeneous computing systems, in: Recent Advances in Computational Optimization: Results of the Workshop on Computational Optimization, 2015, 161–180.
[24] K.M. Tarplee, R. Friese, A.A. Maciejewski, H.J. Siegel, E. Chong, Energy and makespan tradeoffs in heterogeneous computing systems using efficient linear programming techniques, IEEE Trans. Parallel Distrib. Syst. 27 (6) (2016) 1633–1646, Previous version presented in;
K.M. Tarplee, R. Friese, A.A. Maciejewski, H.J. Siegel, Efficient and scalable computation of the energy and makespan pareto front for heterogeneous computing systems, in Federated Conference on Computer Science and Information Systems, Workshop on Computational Optimization, 2013, pp. 401–408.
[25] K.M. Tarplee, A.A. Maciejewski, H.J. Siegel, Energy-aware profit maximizing scheduling algorithm for heterogeneous computing systems, in: 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2014, pp. 595–603.
[26] A. Tchana, N.D. Palma, I. Safieddine, D. Hagimont, Software consolidation as an efficient energy and cost saving solution, Future Gener. Comput. Syst. 58 (2016) 1–12.

Weidong Li received the Ph.D. degree from Department of Mathematics, Yunnan University in 2010. He is currently an Associate Professor at Yunnan University. His main research interests are combinatorial optimization, approximation algorithm, randomized algorithm and cloud computing.



Xi Liu received his Master's degree from Yunnan University in 2012, and he is currently a Ph.D. student at the same university. His main research interests include big data and cloud computing.

Xiaobo Cai received her Ph.D. degree from Yunnan University in 2016. His main research interests include big data and cloud computing.

Xuejie Zhang received the Master's degree in Computer Science Engineering, from Harbin Institute Of Technology in 1990. He received the Ph.D. degree in Computer Science Engineering from the Chinese University of Hong Kong in 1998. He is currently a Professor with the Department of Computer Science and Engineering of Yunnan University. His main research interests are high performance computing, distributed systems, and computer networks.