2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)
October 11-14, 2020. Toronto, Canada

Fine-grained Task Scheduling in Cloud Data Centers Using Simulated-annealing-based Bees Algorithm

Haitao Yuan[1], Jing Bi[2], MengChu Zhou[3], Jia Zhang[4] and Wei Zhang[5]

[1]School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China
[2]Faculty of lnformation Technology, Beijing University of Technology, Beijing 100124, China
[3]Dept. of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102, USA
[4]Dept. of Computer Science, Southern Methodist University, Dallas, TX 75275, USA
[5]Microsoft Redmond, WA 98052, USA

***Abstract*—Cloud computing is increasingly implemented by a growing number of organizations in recent years. Their critical business applications are deployed in distributed cloud data centers (CDCs) for fast response and low cost. The ever-increasing consumption of energy makes it highly important to schedule tasks efficiently in CDCs. In addition, many factors in CDCs, *e.g.*, the wind and solar energy and prices of power grid have spatial differences. It becomes a challenging problem of how to achieve the energy cost minimization for CDCs in such a market. This work applies a *G*/*G*/1 queuing system to evaluate the optimization of servers in each CDC. Furthermore, a single-objective constrained optimization problem is given and addressed by a proposed Simulated-annealing-based Bees Algorithm to yield a close-to-optimal solution. Based on it, a Fine-grained Task Scheduling (FTS) algorithm is designed to minimize the energy cost of CDCs by intelligently scheduling heterogeneous tasks among distributed CDCs. In addition, it also determines running speeds of servers and the number of switched-on servers in each CDC while strictly meeting tasks' delay bounds. Realistic data-driven results demonstrate that FTS outperforms its typical benchmark scheduling peers in terms of energy cost and throughput.**

***Index Terms*—Cloud data centers, simulated annealing, bees algorithm, energy management, intelligent optimization**

I. Introduction

Cloud computing has been changing the way that infrastructure of information technology is provided to meet business requirements of global users [1], [2]. It enables different organizations to dynamically adjust resources based on their requirements by provisioning infrastructure resources in a pay-as-you-go way. A growing number of large-scale enterprises, *e.g.*, Microsoft, Amazon, Facebook and Google, deploy their applications in cloud data centers (CDCs) and provide services in a more cost-efficient manner. According to [3], CDCs need roughly 2.2% of the total electricity in U.S., and produce over 43 million tons of CO_2 in each year. It is also shown that they would need over 101.3 billion kilowatt-hours and the electricity cost would be over $7 billion in 2020. Servers and other facilities, *e.g.*, cooling and lighting in each CDC usually consume as much energy as over 25,000 families. Therefore, the ever-increasing increase in energy cost makes it highly challenging to optimize the energy cost of CDC providers.

It is shown that servers in CDCs consume 28% of energy consumption of CDC providers [4], [5]. There are typically two ways to reduce the energy cost: switching off servers or deteriorating Quality of Service (QoS) of tasks. However, purely decreasing the energy cost also lowers their QoS. In addition, users' QoS requirements are often set in Service Level Agreements (SLAs) [6], [7]. Any compromises of QoS lead to the penalty to a CDC provider because users' performance needs tend to be strict in most cases, thus increasing the total cost. Consequently, CDC providers have to adopt some mechanisms to ensure that their total cost is not increased because of SLA compromises. In addition, users' tasks are transmitted to distributed CDCs located in different sites for cost and performance concerns. Many factors, *e.g.*, available renewable energy, prices of power grid, available number of servers, running speeds of servers, and available energy in each CDC all have spatial differences [8].

Therefore, it becomes a big challenge of how to achieve the energy cost minimization for a CDC provider in such a market. To address it, this work applies a *G*/*G*/1 queuing model to evaluate the performance of servers in CDCs. The interarrival time and running time of tasks can have any distributions of probability. Furthermore, a Fine-grained Task Scheduling (FTS) algorithm is proposed to minimize CDCs' energy cost by considering spatial differences such that tasks' delay bounds are met. The spatial variations are incorporated into a constrained optimization problem. It is addressed by an improved Simulated-annealing-based Bees Algorithm (SBA) to yield a close-to-optimal task scheduling strategy. It properly utilizes power grid and renewable energy by optimally scheduling tasks among CDCs, and determining running speeds of servers and the number of switched-on servers in CDCs. Realistic data including green energy data, prices of power grid, arriving tasks in Google cluster trace[1], are adopted to evaluate the performance of FTS. The experimental

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grants 61802015 and 61703011, in part by the Major Science and Technology Program for Water Pollution Control and Treatment of China under Grant 2018ZX07111005, and in part by the National Defense Pre-Research Foundation of China under Grants 41401020401 and 41401050102.

[1]https://github.com/google/cluster-data

results prove that FTS yields smaller energy cost and higher throughput than its typical benchmark scheduling peers with QoS assurance.

II. Problem Formulation

Fig. 1 illustrates the architecture of multiple CDCs. The number of CDCs is denoted by $\mathcal{N}^C$. Users send tasks through their different end devices, *e.g.*, laptops, smart phones and computers to $\mathcal{N}^C$ CDCs. *Task Scheduler* periodically runs FTS and schedules tasks in a First-Come-First-Served (FCFS) manner.

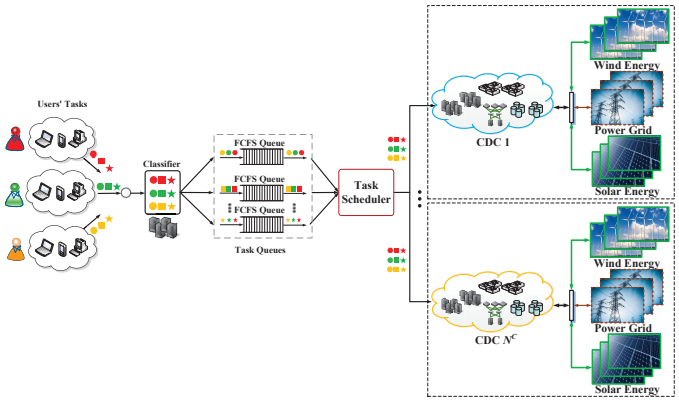


Fig. 1. Illustrative framework of multiple CDCs.

A. Task response time

The number of applications is denoted by $\mathcal{N}^A$. $\hat{N}_{c,n}$ is the number of application n's servers in CDC c ($1 \le c \le \mathcal{N}^C$). $N^o_{\tau,c,n}$ is the number of application n's switched-on servers in CDC c in time slot τ. Thus, $N^o_{\tau,c,n}$ cannot exceed $\hat{N}_{c,n}$, *i.e.*,

$$0 \le N^o_{\tau,c,n} \le \hat{N}_{c,n}, \;\; N^o_{\tau,c,n} \in N^+ \tag{1}$$

This work adopts a $G/G/1$ queuing model to evaluate the performance of servers of each application. $\mathcal{T}^{c,n}_\tau$ is the interarrival time for each application n's server in CDC c in τ. Its mean and variance are $\bar{\mathcal{T}}^{c,n}_\tau$ and $\tilde{\sigma}_{\tau,c,n}$, respectively, and they are determined by analyzing real-life tasks collected from, *e.g.*, Google cluster trace. $\lambda^{c,n}_\tau$ is application n's task arriving rate in CDC c in τ, and it is obtained as $\lambda^{c,n}_\tau = \frac{1}{\bar{\mathcal{T}}^{c,n}_\tau}$. r^n_c is the size of each application n's task scheduled to CDC c, and its probability distribution can be arbitrary. Its mean and variance are $\bar{r}^n_c$ and $\breve{\sigma}_{c,n}$, respectively.

$t^{c,n}_\tau$ and $\varrho^{c,n}_\tau$ are the running time and the running speed of each task on each application n's server in CDC c in τ. Thus, $t^{c,n}_\tau = \frac{r^n_c}{\varrho^{c,n}_\tau}$. $\bar{t}^{c,n}_\tau$, $\tilde{\sigma}_{\tau,c,n}$ and $\Omega^{c,n}_\tau$ are mean, variance and coefficient of variation of $t^{c,n}_\tau$, respectively. Then,

$$\bar{t}^{c,n}_\tau = \frac{\bar{r}^n_c}{\varrho^{c,n}_\tau} \tag{2}$$

$$\tilde{\sigma}_{\tau,c,n} = \frac{\breve{\sigma}_{c,n}}{\left(\varrho^{c,n}_\tau\right)^2} \tag{3}$$

$$\Omega^{c,n}_\tau = \frac{\tilde{\sigma}_{\tau,c,n}}{\bar{t}^{c,n}_\tau} \tag{4}$$

$\hat{\varrho}^n_c$ is each server's maximum running speed for application n in CDC c. Thus, $\varrho^{c,n}_\tau$ is less than or equal to $\hat{\varrho}^n_c$, *i.e.*,

$$0 \le \varrho^{c,n}_\tau \le \hat{\varrho}^n_c \tag{5}$$

Besides, each server's running speed for application n in CDC c has to be sufficient to run its scheduled tasks in τ. Thus, $\varrho^{c,n}_\tau$ needs to be larger than $\frac{\bar{r}^n_c}{\bar{\mathcal{T}}^{c,n}_\tau}$, *i.e.*,

$$\varrho^{c,n}_\tau > \frac{\bar{r}^n_c}{\bar{\mathcal{T}}^{c,n}_\tau} \tag{6}$$

λ^n_τ is task arriving rate of application n in τ, and $\lambda^{c,n}_\tau$ is task arriving rate of application n in CDC c in τ. Then,

$$\lambda^n_\tau = \sum_{c=1}^{\mathcal{N}^C} \lambda^{c,n}_\tau \tag{7}$$

Then, $T^{c,n}_\tau$ ia tasks' response time of application n in CDC c with $N^o_{\tau,c,n}$ servers in τ. L is the length of each time slot. Then, following [9], we have:

$$T^{c,n}_\tau = \frac{\bar{r}^n_c}{\varrho^{c,n}_\tau} + \left((\bar{r}^n_c)^2 + \breve{\sigma}_{c,n}\right) \Delta^1_{\tau,c,n} \tag{8}$$

$$\Delta^1_{\tau,c,n} = \frac{\hat{\sigma}_{\tau,c,n}(\varrho^{c,n}_\tau)^2 + \breve{\sigma}_{c,n}}{2\varrho^{c,n}_\tau(\bar{\mathcal{T}}^{c,n}_\tau \varrho^{c,n}_\tau - \bar{r}^n_c)\left((\bar{\mathcal{T}}^{c,n}_\tau)^2(\varrho^{c,n}_\tau)^2 + \breve{\sigma}_{c,n}\right)}$$

where $\bar{r}^n_c = \frac{\lambda^{c,n}_\tau L}{N^o_{\tau,c,n}}$ and $\bar{\mathcal{T}}^{c,n}_\tau = \frac{N^o_{\tau,c,n}}{\lambda^{c,n}_\tau}$.

$\hat{T}_n$ is a user-specified delay bound of each application n. Then, $T^{c,n}_\tau$ cannot exceed $\hat{T}_n$, *i.e.*,

$$T^{c,n}_\tau \le \hat{T}_n \tag{9}$$

B. Energy cost

The power consumption is a major part of CDCs' energy cost [10]. $P^{c,n}_\tau$ is the power consumed by each application n's server in CDC c in τ. $\chi^{c,n}_\tau$ and $U^{c,n}_\tau$ are clock frequency and supply voltage of application n's servers in CDC c. Following [11], $U^{c,n}_\tau \propto (\chi^{c,n}_\tau)^{\gamma^0_{1,c,n}}$ for $\gamma^0_{1,c,n} > 0$ and $\gamma^0_{1,c,n}$ is a constant. Besides, $\varrho^{c,n}_\tau$ is proportional linearly to $\chi^{c,n}_\tau$, *i.e.*, $\varrho^{c,n}_\tau \propto \chi^{c,n}_\tau$.

Then, following [12], $U^{c,n}_\tau = \gamma^0_{2,c,n}(\chi^{c,n}_\tau)^{\gamma^0_{1,c,n}}$ and $\varrho^{c,n}_\tau = \gamma^0_{3,c,n}\chi^{c,n}_\tau$. $\gamma^0_{2,c,n}$ and $\gamma^0_{3,c,n}$ denote constants for application n's servers in CDC c. Then, $P^{c,n}_\tau$ is calculated as:

$$\begin{aligned} P^{c,n}_\tau &= \Psi^n_c \omega^n_c (U^{c,n}_\tau)^2 \chi^{c,n}_\tau \\ &= \Psi^n_c (\gamma^0_{2,c,n})^2 \omega^n_c (\chi^{c,n}_\tau)^{2\gamma^0_{1,c,n}+1} \\ &= \Psi^n_c (\gamma^0_{2,c,n})^2 \omega^n_c \frac{(\varrho^{c,n}_\tau)^{2\gamma^0_{1,c,n}+1}}{(\gamma^0_{3,c,n})^{2\gamma^0_{1,c,n}+1}} = \urcorner^n_c (\varrho^{c,n}_\tau)^{\Psi^n_c} \end{aligned} \tag{10}$$

where Ψ^n_c, $U^{c,n}_\tau$, ω^n_c and $\chi^{c,n}_\tau$ are activity factor, supply voltage, loading capacitance and clock frequency of each application n's server in CDC c. $\urcorner^n_c = \Psi^n_c (\gamma^0_{2,c,n})^2 \omega^n_c / (\gamma^0_{3,c,n})^{2\gamma^0_{1,c,n}+1}$ and $\Psi^n_c = 2\gamma^0_{1,c,n} + 1$.

Each idle server still needs some power due to the dissipation of short-circuit and static power, and other wasted power [13]. $\breve{\Phi}^c_n$ is the power consumed by each application

n's idle server in CDC c. P_τ^c is the total power of CDC c in τ. Following [14], P_τ^c is obtained as:

$$P_\tau^c=\sum_{n=1}^{\mathcal{N}^A} P_\tau^{c,n}=\sum_{n=1}^{\mathcal{N}^A}\left(N_{\tau,c,n}^o\left(\urcorner_c^n\left(\varrho_\tau^{c,n}\right)\Psi_c^n+\check{\Phi}_n^c\right)\right) \tag{11}$$

$\hat{\mathbb{E}}_c$ denotes the available energy in CDC c. The energy consumption of CDC c in time slot τ is $P_\tau^c L$, and it cannot exceed $\hat{\mathbb{E}}_c$, *i.e.*,

$$\sum_{n=1}^{\mathcal{N}^A}\left(N_{\tau,c,n}^o\left(\urcorner_c^n\left(\varrho_\tau^{c,n}\right)\Psi_c^n+\check{\Phi}_n^c\right)\right)\leq\hat{\mathbb{E}}_c \tag{12}$$

$\widetilde{P}_{\tau,c}$ and $P_{\tau,c}^\circ$ denote wind and solar power produced by CDC c in time slot τ. Following [14], $\widetilde{P}_{\tau,c}$ is obtained as:

$$\widetilde{P}_{\tau,c}=\frac{1}{2}\phi_{1c}\phi_{2c}\phi_{3c}\left(\phi_{\tau,4c}\right)^3 \tag{13}$$

where ϕ_{1c}, ϕ_{2c}, ϕ_{3c} and $\phi_{\tau,4c}$ denote conversion rate of wind to electricity, air density, area of wind turbine rotor, and wind speed of CDC c in time slot τ, respectively.

Following [14], $P_{\tau,c}^\circ$ is obtained as:

$$P_{\tau,c}^\circ=\psi_{\tau,1c}\psi_{2c}\psi_{3c} \tag{14}$$

where $\psi_{\tau,1c}$, ψ_{2c} and ψ_{3c} denote solar irradiance, area of solar panel irradiation and conversion rate of solar radiation to electricity of CDC c in time slot τ, respectively.

The price of power grid of CDC c in τ is denoted by p_τ^c, and CDC c's energy cost, E_τ^c, is obtained as:

$$E_\tau^c=\left(\max\left(P_\tau^c-P_{\tau,c}^\circ-\widetilde{P}_{\tau,c},0\right)\right)p_\tau^c L \tag{15}$$

Then, the energy cost of the CDC provider is given as:

$$E_\tau=\sum_{c=1}^{\mathcal{N}^C}E_\tau^c=\sum_{c=1}^{\mathcal{N}^C}\left(\max\left(P_\tau^c-P_{\tau,c}^\circ-\widetilde{P}_{\tau,c},0\right)\right)p_\tau^c L \tag{16}$$

Our optimization objective is to minimize E_τ, *i.e.*,

$$\underset{N_{\tau,c,n}^o,\lambda_\tau^{c,n},\varrho_\tau^{c,n}}{\mathbf{Min}}\ \{E_\tau\} \tag{17}$$

subject to (1), (5), (6), (7), (9), (12), (18)–(20).

$$N_{\tau,c,n}^o\in N^+,\ \varrho_\tau^{c,n}\geq 0,\ \lambda_\tau^{c,n}\geq 0 \tag{18}$$

$$\lambda_\tau^{c,n}=0,\ \text{if } N_{\tau,c,n}^o=0 \tag{19}$$

$$\lambda_\tau^{c,n}>0,\ \text{if } N_{\tau,c,n}^o>0 \tag{20}$$

III. Simulated-annealing-based Bees Algorithm

There are several traditional algorithms, *e.g.*, Lagrange multiplier and dynamic programming, to solve the formulated problem [15]. Yet they require the first-order or second-order derivatives of (17) that do not exist [16]. They can efficiently find good solutions to different types of complex optimization problems [17]. Among them, bees algorithm (BA) is motivated by foraging process of honey bees in nature. It is widely adopted because of its high feasibility and fast convergence [18]. A population of honey bees in BA change their locations in multiple directions. Flower patches that have more pollen attract more honey bees because they are easier to be collected. Each scout bee randomly moves among different patches. Then, it deposits pollen and performs a waggle dance to exchange its search information with others. Then, patches that have more pollen are searched by more honey bees.

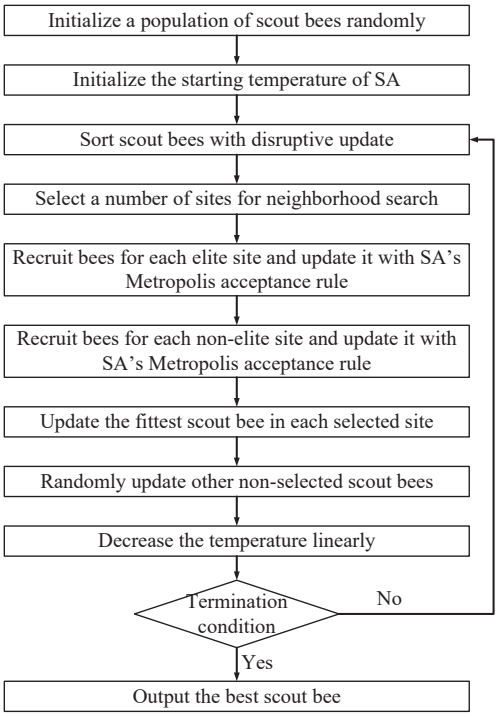


Fig. 2. Flowchart of SBA.

BA can efficiently yield high-quality solutions when it is applied to solve complicated optimization problems. but it often converges to local optima. Simulated annealing (SA) can jump out of local optima by conditionally allowing several moves that deteriorate quality of solutions with its Metropolis acceptance rule. It is shown that SA can yield global optima with high probability, and it can produce high-quality solutions to complex optimization problems [19]. Nevertheless, it suffers from slow convergence. To overcome such drawback, this work proposes an improved algorithm called SA-based Bees Algorithm (SBA) to solve (17)–(20) by adding SA's Metropolis acceptance rule into BA. Specifically, all non-elite and elite bees are updated conditionally with the rule. The flowchart of SBA is given in Fig. 2. The details of scout bee encoding, population initialization, disruptive update and SA-based selection are given next.

A. Population initialization

$\mathbf{x}_i$ is the position of each scout bee i, and it corresponds to decision variables including $N_{\tau,c,n}^o$, $\lambda_\tau^{c,n}$ and $\varrho_\tau^{c,n}$. $|\mathbb{X}|$ is the population size. $\mathbf{x}_{i,d}^0$ is the initial decision variable d of scout bee i ($i\in\{1,2,\ldots,|\mathbb{X}|\}$), *i.e.*,

$$\mathbf{x}_{i,d}^0=\check{\boldsymbol{\theta}}^d+\theta_{1,i}*\left(\hat{\boldsymbol{\theta}}^d-\check{\boldsymbol{\theta}}^d\right) \tag{21}$$

where $\check{\boldsymbol{\theta}}^d$ and $\hat{\boldsymbol{\theta}}^d$ are lower and upper limits of d, and $\theta_{1,i}$ ($\theta_{1,i}\in(0,1)$) is a random number for scout bee i.

B. Disruptive update

Disruptive update prefers lower and higher scout bees in each population. To achieve it, the fitness ($\tilde{\mathcal{F}}(\mathbf{x}_i)$) of each scout bee i is defined as:

$$\tilde{\mathcal{F}}(\mathbf{x}_i)=|\acute{\mathcal{F}}(\mathbf{x}_i)-\bar{\mathcal{F}}| \tag{22}$$

$$\bar{\mathcal{F}}=\frac{\sum_{i=1}^{|\mathbb{X}|}\acute{\mathcal{F}}(\mathbf{x}_i)}{|\mathbb{X}|},\ \acute{\mathcal{F}}(\mathbf{x}_i)=\frac{E_\tau(\mathbf{x}_i)}{\sum_{i=1}^{|\mathbb{X}|}E_\tau(\mathbf{x}_i)}$$

where $\acute{\mathcal{F}}(\mathbf{x}_i)$ denotes the normalized value of $E_\tau(\mathbf{x}_i)$ and $\bar{\mathcal{F}}$ denotes the average value of all normalized values. The disruptive selection increases the diversity of scout bees by keeping diverse ones. Then, all scout bees are sorted by $\tilde{\mathcal{F}}(\mathbf{x}_i)$.

C. SA-based selection

The SA-based selection is adopted to update all elite or non-elite bees. $\mathbf{x}_i^g$ and $\mathbf{x}_i^{g+1}$ are positions of scout bee i in iterations g and $g+1$. If $\tilde{\mathcal{F}}(\mathbf{x}_i^g)\geq\tilde{\mathcal{F}}(\mathbf{x}_i^{g+1})$, $\mathbf{x}_i^{g+1}$ is selected; otherwise, it is selected only if

$$\frac{e^{\left(\tilde{\mathcal{F}}(\mathbf{x}_i^g)-\tilde{\mathcal{F}}(\mathbf{x}_i^{g+1})\right)}}{\theta^g}>\theta_{2,i} \tag{23}$$

where $\theta_{2,i}$ is a random number in (0,1) and θ^g is temperature in iteration g.

IV. Performance Evaluation

A. Parameter setting

As shown in Fig. 3, this work collects real-life task arriving rates of three types of applications in Google cluster trace to evaluate SBA and its peers. As illustrated in Fig. 4(a), this work collects real-life prices of power grid in three locations[2] for three CDCs. In addition, L=5 minutes, $\mathcal{N}^C$=3 and $\mathcal{N}^A$=3.

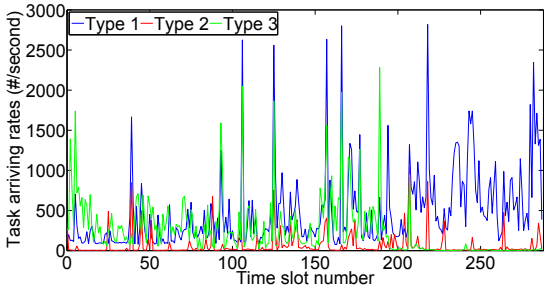


Fig. 3. Task arriving rates of three types.

According to [8], energy-related parameters are set in Table I. This work collects realistic data about solar irradiance[3] and wind speed[4] for one day. Figs. 4(b) and 4(c) illustrate solar irradiance and wind speed in three CDCs, respectively. $\daleth_c^n$, Ψ_c^n, $\check{\Phi}_n^c$, $\hat{N}_{c,n}$ and $\hat{\varrho}_c^n$ are given in Table II. $\hat{T}_1$=0.05 seconds, $\hat{T}_2$=0.1 seconds, and $\hat{T}_3$=0.15 seconds. In addition, $\hat{\mathbb{E}}_1=2.1\times10^{11}$ (WH), $\hat{\mathbb{E}}_2=2.5\times10^{11}$ (WH), and $\hat{\mathbb{E}}_3=1.25\times10^{11}$ (WH). Based on [19], [20], the total number of iterations of SBA is 1000. The population size is 30. The number of selected sites is 15. The number of elite sites is 6. The number of recruited bees for each elite site and each non-elite one is 30 and 15, respectively. The initial temperature of SBA is 5×10^6 and its cooling rate is 0.985.

TABLE I
Parameter Setting of Wind and Solar Energy

	Wind energy			Solar energy	
	ϕ_{1c}	ϕ_{2c}(kg/m^3)	ψ_{3c}(m^2)	ψ_{2c}(m^2)	ψ_{3c}
c=1	0.3	1.225	250	150	0.2
c=2	0.375	1.5313	312.5	187.5	0.25
c=3	0.45	1.8375	375	225	0.3

B. Experimental results

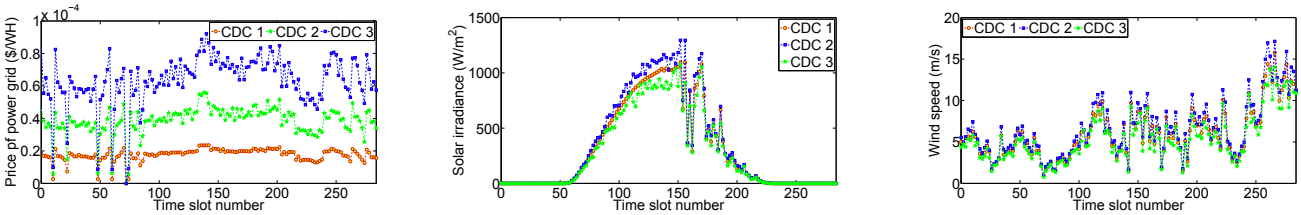
To demonstrate the performance of SBA, this work compares SBA with its two typical meta-heuristic optimization peers, including BA [18] and Genetic Learning Particle Swarm Optimization (GL-PSO) [21]. All algorithms are repeated independently for 30 times to yield comparison results. The reasons of choosing them for comparison are described as:

1) BA [18]: As a population-based optimization algorithm, BA can efficiently obtain high-quality solutions to different problems. However, its several parameters need to be tuned carefully, and it is easy to converge to local optima.
2) GL-PSO [21]: GL-PSO adopts crossover, mutation and selection operations on search information of particles to yield diverse and high-quality elite particles that guide its search. Then, PSO's search accuracy is improved.

BA has the same setting of parameters as SBA. GL-PSO's parameters are given as follows. The iteration number and the population size are 1000 and 100, respectively. The intertia weight and maximum velocity are 0.7298 and 10, respectively. The mutation probability and the learning coefficient of elite particles are 0.1 and 1.49618, respectively. The learning coefficients of the globally and the locally best particles are both set to 2. All three algorithms stop their searches if their current solutions are not improved in consecutive 10 iterations.

Fig. 5(a) illustrates the energy cost of BA, SBA and GL-PSO in each time slot, respectively. Compared with BA and GL-PSO, SBA's energy cost is reduced by 59.07% and 92.83% on average, respectively. Fig. 5(b) illustrates their running time comparison. SBA's running time is reduced by 26.31% and 46.15% on average, respectively in comparison with BA and GL-PSO. Furthermore, Fig. 5(c) shows iterations of their energy cost in time slot 1. It is observed that GL-PSO and BA require 996 and 951 iterations to stop their searches, and their energy cost are 218339.94$ and 76165.77$, respectively. SBA only requires 201 iterations to stop its search, and its energy

[2]http://www.energyonline.com/Data/
[3]http://www.nrel.gov/midc/srrl_bms/
[4]http://www.nrel.gov/midc/nwtc_m2/

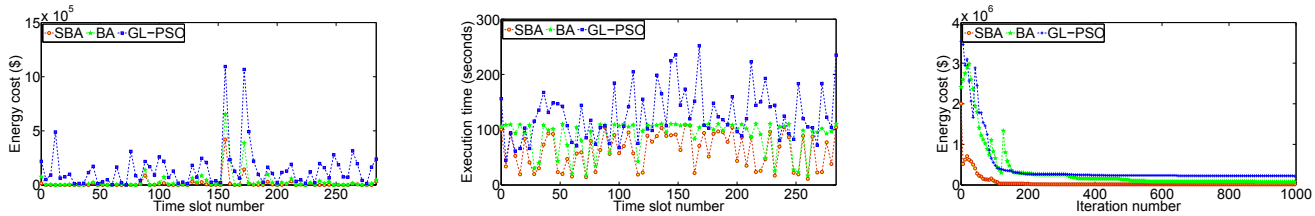


(a) Prices of power grid (b) Solar irradiance (c) Wind speeds

Fig. 4. Prices of power grid, solar irradiance and wind speeds.

TABLE II
PARAMETER SETTING OF THREE CDCS

	$\lceil_c^n$ (tasks/second)			Ψ_c^n (W)			$\check{\Phi}_n^c$ (W)			$\hat{N}_{c,n}$			$\hat{\varrho}_c^n$		
	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3	n=1	n=2	n=3
c=1	10.5	7.4	4.4	1.1	1.15	1.25	200	300	400	110	120	130	2.9×10^4	3.0×10^4	3.1×10^4
c=2	12.5	9.4	6.4	1.0	1.1	1.2	400	500	600	140	150	160	3.0×10^4	3.1×10^4	3.2×10^4
c=3	14.5	11.4	6.4	1.1	1.2	1.3	600	700	800	160	170	180	3.2×10^4	3.3×10^4	3.4×10^4



(a) Energy cost (b) Execution time (c) Energy cost of each iteration in time slot 1

Fig. 5. Comparison of SBA, BA and GL-PSO.

cost is 14832.58$. Therefore, SBA achieves less energy cost in much less time than GL-PSO and BA.

To demonstrate the performance of FTS, this work compares it with the following up-to-date scheduling peers [16], [22]–[24] in terms of energy cost and throughput.

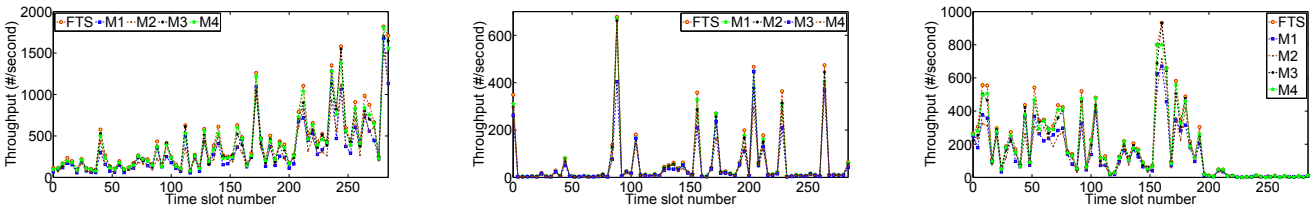
1) M1. Similar to cheap-power-grid-price-first scheduling in [22], it executes tasks in CDCs according to the order of power grid prices.
2) M2. Similar to renewable-energy-first scheduling in [16], it executes tasks in CDCs according to the order of amount of renewable energy.
3) M3 [23]. It executes tasks among CDCs by considering spatial and temporal differences of power grid prices.
4) M4 [24]. It smartly executes tasks among CDCs by considering spatial differences of power grid prices.

Fig. 6 illustrates the throughput of FTS, and M1–M4, respectively. It is shown that FTS's throughput is larger than those of M1–M4 for each type, respectively. For example, FTS's throughput of application 1 is larger than those of M1–M4 by 25.99%, 25.37%, 10.30% and 7.74% on average, respectively. This is because that running speeds of servers, maximum number of servers, and available energy in different CDCs are all constrained. FTS jointly considers these limits and schedules tasks among CDCs, and determines running speeds of servers and number of switched-on servers in CDCs.

Fig. 6 shows the energy cost of FTS, and M1–M4, respectively. According to SLAs [25], the penalty of each refused task is larger than its largest energy cost among CDCs. In Fig. 6, the energy cost is obtained by calculating the energy cost of executed tasks, and the penalty of refused tasks in each time slot. Fig. 7 shows that compared with M1–M4, FTS reduces the energy cost by 50.11%, 51.55%, 29.15%, and 25.27% on average, respectively. The reason is that FTS schedules tasks among CDCs by jointly considering spatial variations in available renewable energy and prices of power grid of CDCs.

V. CONCLUSION

Cloud computing enables organizations to cost-effectively improve the performance of their key tasks and decrease their capital cost. However, it suffers from a problem of high energy consumption due to increase of users' arriving tasks. More and more large-scale organizations adopt cloud data centers (CDCs) to manage their applications and provide services to global users. Current studies fail to analyze tasks' performance and energy cost of a CDC provider in a fine-grained manner. Besides, several factors, *e.g.*, amount of renewable energy and prices of power grid in different CDCs have spatial variations. Thus, it brings an opportunity to achieve the energy cost minimization for the CDC provider. This work adopts a



(a) Type 1 (b) Type 2 (c) Type 3

Fig. 6. Throughput of FTS, M1, M2, M3 and M4.

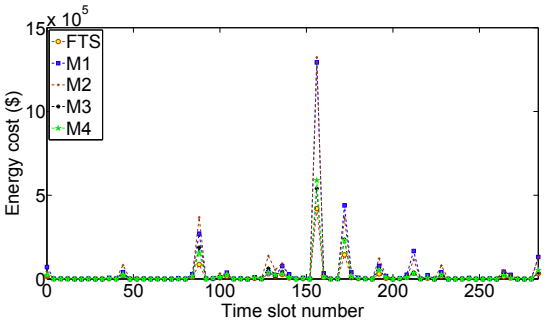


Fig. 7. Energy cost of FTS, M1, M2, M3 and M4.

$G/G/1$ queuing model to evaluate the performance of servers in CDCs, based on which a constrained optimization problem is given and solved by a designed Simulated-annealing-based Bees Algorithm to yield a high-quality solution. In this way, a Fine-grained Task Scheduling (FTS) method is developed to minimize energy cost of the CDC provider. Simulations verify that FTS outperforms its several up-to-date peers in terms of energy cost and throughput.

REFERENCES

[1] T. Wood, K. K. Ramakrishnan, J. Hwang, G. Liu and W. Zhang, "Toward a software-based network: integrating software defined networking and network function virtualization," *IEEE Network*, vol. 29, no. 3, pp. 36–41, Jun. 2015.

[2] P. Pierleoni, R. Concetti, A. Belli and L. Palma, "Amazon, Google and Microsoft Solutions for IoT: Architectures and a Performance Comparison," *IEEE Access*, vol. 8, pp. 5455–5470, Dec. 2019.

[3] E. Baccarelli, N. Cordeschi, A. Mei, *etc*., "Energy-efficient Dynamic Traffic Offloading and Reconfiguration of Networked Data Centers for Big Data Stream Mobile Computing: Review, Challenges, and a Case Study," *IEEE Network*, vol. 30, no. 2, pp. 54–61, Apr. 2016.

[4] E. Cortez, A. Bonde, A. Muzio, M. Russinovich, M. Fontoura, and R. Bianchini, "Resource Central: Understanding and Predicting Workloads for Improved Resource Management in Large Cloud Platforms," in *Proc. of 26th Symposium on Operating Systems Principles*, New York, NY, USA, 2017, pp. 153–167.

[5] J. Bi, H. Yuan, M. Zhou and Q. Liu, "Time-Dependent Cloud Workload Forecasting via Multi-Task Learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2401–2406, Jul. 2019.

[6] S. Mubeen, S. A. Asadollah, A. V. Papadopoulos, M. Ashjaei, H. Pei-Breivold and M. Behnam, "Management of Service Level Agreements for Cloud Services in IoT: A Systematic Mapping Study," *IEEE Access*, vol. 6, pp. 30184–30207, Aug. 2018.

[7] B. Bai, Y. Fan, W. Tan, and J. Zhang, "DLTSR: A Deep Learning Framework for Recommendation of Long-tail Web Services", *IEEE Trans. on Services Computing*, vol. 13, no. 1 pp. 73–85, Feb. 2020.

[8] A. Kiani and N. Ansari, "A Fundamental Tradeoff Between Total and Brown Power Consumption in Geographically Dispersed Data Centers," *IEEE Communications Letters*, vol. 20, no. 10, pp. 1955–1958, Oct. 2016.

[9] K. Li, "Optimal Power and Performance Management for Heterogeneous and Arbitrary Cloud Servers," *IEEE Access*, vol. 7, pp. 5071–5084, 2019.

[10] H. Yuan, J. Bi, W. Tan and B. H. Li, "CAWSAC: Cost-Aware Workload Scheduling and Admission Control for Distributed Cloud Data Centers," *IEEE Trans. Aut. Sci. and Eng.*, vol. 13, no. 2, pp. 976–985, Apr. 2016.

[11] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, "Theoretical and Practical Limits of Dynamic Voltage Scaling," in *Proc. 41st Design Automation Conference*, San Diego, CA, USA, 2004, pp. 868–873.

[12] L. Gu, D. Zeng, A. Barnawi, *etc*., "Optimal Task Placement with QoS Constraints in Geo-Distributed Data Centers Using DVFS," *IEEE Trans. on Computers*, vol. 64, no. 7, pp. 2049–2059, Jul. 2015.

[13] W. Dou, X. Xu, S. Meng, X. Zhang, C. Hu, S. Yu, and J. Yang, "An Energy-aware Virtual Machine Scheduling Method for Service QoS Enhancement in Clouds over Big Data," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 14, 1–20, Jul. 2017.

[14] H. Yuan, J. Bi and M. Zhou, "Spatiotemporal Task Scheduling for Heterogeneous Delay-Tolerant Applications in Distributed Green Data Centers," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1686–1697, Oct. 2019.

[15] Z. Zhao, S. Liu, M. Zhou, X. Guo and L. Qi, "Decomposition Method for New Single-Machine Scheduling Problems From Steel Production Systems," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1376–1387, Jul. 2020.

[16] J. Bi, H. Yuan, W. Tan, and B. H. Li. "TRS: Temporal Request Scheduling with Bounded Delay Assurance in a Green Cloud Data Center," *Information Sciences*, vol. 360, pp. 57–72, Sept. 2016.

[17] S. Pare, A. Kumar, V. Bajaj and G. K. Singh, "A Context Sensitive Multilevel Thresholding Using Swarm Based Algorithms," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1471–1486, Nov. 2019.

[18] E. Yigit and H. Duysak, "Determination of Optimal Layer Sequence and Thickness for Broadband Multilayer Absorber Design Using Double-Stage Artificial Bee Colony Algorithm," *IEEE Trans. on Microwave Theory and Techniques*, vol. 67, no. 8, pp. 3306–3317, Aug. 2019.

[19] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li and J. Li, "TTSA: An Effective Scheduling Approach for Delay Bounded Tasks in Hybrid Clouds," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[20] H. Yuan, J. Bi, W. Tan and B. H. Li, "Temporal Task Scheduling With Constrained Service Delay for Profit Maximization in Hybrid Clouds," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 1, pp. 337–348, Jan. 2017.

[21] Y. Gong, J. Li, Y. Zhou, Y. Li, H. Chung, Y. Shi, and J. Zhang, "Genetic Learning Particle Swarm Optimization," *IEEE Transactions on Cybernetics*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[22] X. Deng, D. Wu, J. Shen, and J. He, "Eco-Aware Online Power Management and Load Scheduling for Green Cloud Datacenters," *IEEE Systems Journal*, vol. 10, no. 1, pp. 78–87, Mar. 2016.

[23] J. Luo, L. Rao and X. Liu, "Spatio-Temporal Load Balancing for Energy Cost Optimization in Distributed Internet Data Centers," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 387–397, Jul. 2015.

[24] H. Yuan, J. Bi and M. Zhou, "Spatial Task Scheduling for Cost Minimization in Distributed Green Cloud Data Centers," *IEEE Trans. on Automation Sci. and Eng.*, vol. 16, no. 2, pp. 729–740, Apr. 2019.

[25] H. Yuan, M. Zhou, Q. Liu, and A. Abusorrah, "Fine-Grained Resource Provisioning and Task Scheduling for Heterogeneous Applications in Distributed Green Clouds," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 5, pp. 1380–1393, Sept. 2020.