Temporal Task Scheduling for Delay-constrained Applications in Geo-Distributed Cloud Data Centers

Jing Bi[1], Haitao Yuan[2], Jia Zhang[3], and MengChu Zhou[4]
[1]School of Software Engineering, Beijing University of Technology, Beijing, China
[2]School of Software Engineering, Beijing Jiaotong University, Beijing, China
[3]Department of Electrical and Computer Engineering, Carnegie Mellon University, Moffett Field, USA
[4]Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, USA
bijing@bjut.edu.cn, htyuan@bjtu.edu.cn, jia.zhang@sv.cmu.edu, zhou@njit.edu

***Abstract*—A growing number of global companies select Green Cloud Data Centers (GCDCs) to manage their delay-constrained applications. The fast growth of users' tasks dramatically increases the energy consumed by GCDC, e.g., Google. The random nature of tasks brings a big challenge of scheduling tasks of each application with limited infrastructure resources of GCDCs. This work accurately computes a mathematical relation between task service rates and the number of tasks refusal in GCDC. Besides, it proposes a Temporal Task Scheduling (TTS) algorithm investigating the temporal variation in geo-distributed cloud data centers to schedule all tasks within their delay constraints. Furthermore, a novel dynamic hybrid meta-heuristic algorithm is developed for the formulated profit maximization problem, based on genetic simulated annealing and particle swarm optimization. The proposed algorithm can guarantee that differentiated service qualities can be provided with higher overall performance and lower energy cost. Trace-driven simulations demonstrate that larger throughput and profit is achieved than several existing scheduling algorithms.**

***Keywords*-Green cloud data center, temporal task scheduling, delay-constrained application, profit maximization, hybrid meta-heuristic optimization**

I. Introduction

Current cloud data centers manage many large-scale infrastructures including server farms and cooling facilities [1]. In recent years, an increasing number of users deploy their delay-constrained applications, e.g., big data processing, video analysis, and high-performance computing in data centers. This significantly increases the amount of energy consumption of large-scale data centers. Data centers consumed around 70 billion kilowatt-hours that was 2% of domestic energy in U.S. in 2014 [2]. It is predicted that it will be doubled in 2020. Besides, more than 57% of electricity energy in U.S. is generated by burning brown energy, e.g., petroleum and coal, and it brings severe environmental damages. Current Green Cloud Data Centers (GCDCs) are primarily powered by three energy sources, i.e., power grid, solar energy and wind energy, and these works aim to reduce the brown energy consumption by adopting renewable energy devices [3]. The aperiodic arrival of tasks makes it difficult to accurately predict task arriving rate of each application. Thus, it is impossible to execute all tasks of each application with limited resources of the GCDC at peak time. For example, when Apple's new iPhones are released, more than two million pre-orders are sent to its data centers in the first 24 hours, and the Apple Store is unresponsive in some areas. In addition, the response time of the GCDC may be relatively long and application crash may happen at peak time. Thus, hybrid cloud is increasingly deployed by the GCDC to handle peak tasks by outsourcing some tasks to public clouds. Virtual Machines (VMs) are dynamically created to support multiple applications in cloud infrastructure environment and the operation cost of the GCDC can be reduced. Currently, more than 82% of companies choose hybrid cloud to deploy their applications [4]. In hybrid cloud, the GCDC needs to pay the execution cost of VMs due to the tasks scheduled to public clouds. Therefore, it is challenging to schedule tasks of each application among the GCDC and public clouds in a cost-effective way while strictly guaranteeing their expected delay constraints.

Similar to [5], this work investigates task scheduling of multiple applications whose delay constraints are relatively long, e.g., high-performance simulation, and large-scale data analysis. Within delay constraints of the applications, several factors in the GCDC and public clouds show the temporal variation. Specifically, revenue, price of power grid, solar irradiance, and wind speed in the GCDC all vary with time. Besides, price of VMs in public clouds also varies with time. Thus, the temporal variation in the factors makes it challenging to maximize profit of the GCDC while strictly meeting delay constraint of each task. What's more, many studies selectively admit some tasks to avoid overload of the GCDC [6]. However, they fail to explicitly present a mathematical relation between the number of tasks refused and task service rates of the GCDC. Different from them, this work explicitly presents the mathematical formulation of this relation. In addition, most of existing task scheduling algorithms are proposed according to the queueing theory. However, they can only meet the average delay constraints of all tasks. There exists a long tail effect in delay of tasks in clouds, and it leads to that delay constraints of some tasks are not met [7]. To address such issues, this work proposes a Temporal Task Scheduling (TTS) algorithm in Geo-Distributed Cloud Data Centers (GDCDCs) to meet delay constraints of all tasks.

In this work, the profit maximization problem of GDCDC is formulated and solved with a hybrid optimization algorithm that combines Genetic Algorithm (GA) [8], Simulated

Annealing (SA) [9] and Particle Swarm Optimization (PSO) [10]. The combination of PSO and GA improves global search capacity by establishing superior particles for PSO because of PSO's social learning and GA's global effectiveness. It consists of two layers, the first for generation of superior particles and the second for updating particles. By adopting superior particles produced by GA and Metropolis acceptance rule of SA, a novel hybrid algorithm named Genetic Simulated-annealing-based PSO (GSPSO) is proposed. The Metropolis acceptance rule of SA enables the updates that deteriorate the quality of solutions and aims to find global optima by escaping from local optima. The genetic operations of GA are used to produce superior particles from which PSO's particles learn. In addition, historical search information of particles in PSO directs the evolution of superior particles. Therefore, for increasing both efficiency and global search accuracy, superior particles are highly qualified to guide others particles.

To summarize, our contributions in this work are threefold.

1) We accurately compute a mathematical relation between the number of tasks refusal and task service rates of the GCDC;
2) We propose a novel TTS algorithm to consider the temporal variation within tasks' delay constraints , and smartly execute it to the GCDC and public clouds; and
3) We formulate a profit maximization problem of GDCDC as a constrained nonlinear optimization one and propose a novel hybrid algorithm to solve it.

The remaining of this work is described as follows. A motivation and GDCDC system architecture are presented in Section II. The profit maximization problem of GDCDC is formulated in Section III. The proposed solution algorithms are presented in Section IV. The experimental results and their analysis based on real-life data are shown in Section V. Section VI concludes this paper.

II. Motivation and System Architecture

A data center provider usually owns GCDC and executes some tasks to external public clouds when its local resources are limited or the use of VMs in public clouds is more economic and cheaper. The proposed GDCDC architecture is presented in Fig. 1. Users' tasks are sent to GDCDC through multiple devices, e.g., computers, smart phones, laptops, and servers. Then, *Task Classifier* classifies arriving tasks and enqueues them into corresponding First-Come-First-Served (FCFS) queues according to their application types. Then, task arriving rates of all applications are further set. In Fig. 1, μ_τ^n and Λ_τ^{na} denote task service rate and the cumulated task arriving rate of application n in interval τ, respectively. This work focuses on *Task Scheduler* where TTS periodically executes and schedules all tasks of each application to GDCDC within their delay constraints.

GCDC mainly obtains electricity energy from three types of sources: grid energy ($\hbar$), solar energy (s), and wind energy (w). The *Task Scheduler* periodically collects the information of $\hbar$, s, and w including the price of power grid, the idle (peak) power of each server, the wind speed, the solar irradiance, and on-site air density. Section III-D formulates the task scheduling problem as a constrained Mixed Integer Non-Linear Program (MINLP) [14]. Then, it is solved by GSPSO in each iteration of TTS to obtain the scheduling strategy. Based on it, TTS is proposed to achieve profit maximization for GDCDC while strictly meeting delay constraints of all tasks of multiple applications. Then, the task service rate for each application in each interval is specified by *Task Scheduler*. Based on it, *Resource Allocator* configures each server in GCDC. It is assumed that servers for a same application are homogeneous and servers for different applications are heterogeneous with respect to in hardware setting [15]. Note that this work mainly considers *Task Scheduler* and *Resource Allocator*.

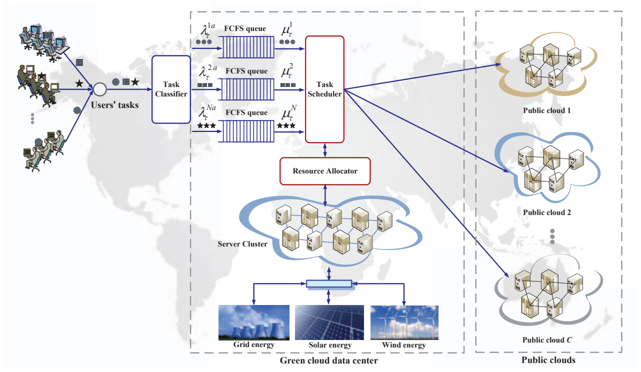


Fig. 1. GDCDC system architecture.

III. Problem Formulation

Based on the GDCDC system architecture, the profit maximization problem is formulated here. Similar to [13], GDCDC is modeled as a discrete-time system that evolves with intervals. An increasing number of high-performance servers are deployed in GDCDC. Therefore, it is reasonable to assume that each task can finish its execution in one single interval.

A. Delay Bound Constraint

Let B_n denote the delay constraint of tasks of application n. This means that before τ passes, application n's tasks that arrive in interval $\tau - B_n$ or before are all scheduled to GDCDC. The task arriving rate of application n in interval τ is denoted by Λ_τ^n. Task service rates in τ and $\tau+b$ ($1 \leq b \leq B_n$) are denoted by μ_τ^n and $\mu_{\tau+b}^n$, respectively. The task service rate denotes the rate at which tasks of application n are removed from their FCFS queue and executed to GDCDC in τ. The number of tasks of application n cumulated in τ intervals is denoted by Δ_τ^n. The number of tasks of application n executed in τ intervals is denoted by Ψ_τ^n. Besides, if application n's tasks are executed to public cloud c in τ, $y_\tau^{nc}=1$; otherwise, $y_\tau^{nc}=0$. The number of tasks of application n executed to public cloud c in τ is denoted by d_τ^{nc}. L denotes the length of an interval. Then, we can obtain Δ_τ^n and Ψ_τ^n:

$$\Delta_\tau^n = \sum_{i=1}^{\tau} \Lambda_i^n L \tag{1}$$

$$\Psi_\tau^n = \sum_{i=1}^{\tau}\left(\Lambda_i^{na}(1-\delta(\Lambda_i^{na},\mu_i^n))L + \sum_{c=1}^{C} x_i^{nc} d_i^{nc}\right) \quad (2)$$

where in (2), Λ_τ^{na} denotes the cumulated arriving rate of application n's tasks in τ. The remaining arriving rate of application n's tasks in τ is denoted by Λ_τ^{nr}. Thus, all application n's tasks that arrive in $\tau - B_n$ or earlier are executed in GDCDC before τ passes. Thus, $\Lambda_i^{nr}=0$ $(i \leq \tau - B_n - 1)$. We can obtain Λ_τ^{na}:

$$\Lambda_\tau^{na} = \Lambda_\tau^n + \sum_{i=\tau-B_n}^{\tau-1} \Lambda_i^{nr}. \quad (3)$$

$\delta(\Lambda_\tau^{na}, \mu_\tau^n)$ in (2) means the loss possibility of application n's tasks in τ. Similar to [16], all servers for application n are modeled as an M/M/1/C_n/∞ system. The largest number of tasks that application n's servers can execute is denoted by C_n. Then,

$$\delta(\Lambda_\tau^{na}, \mu_\tau^n) = \begin{cases} \frac{1-\frac{\Lambda_\tau^{na}}{\mu_\tau^n}}{1-\left(\frac{\Lambda_\tau^{na}}{\mu_\tau^n}\right)^{C_n+1}}\left(\frac{\Lambda_\tau^{na}}{\mu_\tau^n}\right)^{C_n} & \mu_\tau^n > 0, \\ 1 & \mu_\tau^n = 0. \end{cases} \quad (4)$$

In addition, all application n's tasks arriving in τ have to be executed in slots τ to $\tau + B_n$. It means that all application n's tasks in $\tau - B_n$ or earlier must be executed before τ passes. Then,

$$\Delta_{\tau-B_n-1}^n + \Lambda_{\tau-B_n}^n L \leq \Phi_{\tau-1}^n + \Lambda_\tau^{na}(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))L + \sum_{c=1}^{C} x_\tau^{nc} d_\tau^{nc}. \quad (5)$$

Let $\widetilde{\Lambda}_u^{na}$ denote the cumulated arriving rate of application n's tasks in u $(\tau+1 \leq u \leq \tau + B_n)$. Let $\widetilde{\mu}_u^n$ denote the task service rate of application n in u $(\tau+1 \leq u \leq \tau+B_n)$. There are several existing prediction algorithms, e.g., stacked auto-encoder [17], and deep neural networks [18], to well predict $\widetilde{\Lambda}_u^{na}$. Here we assume that $\widetilde{\Lambda}_u^{na}$ is known in advance. Thus, application n's tasks in $\tau+b-B_n$ or earlier have to be executed before $\tau+b$ passes. Then,

$$\Delta_{\tau-B_n-1}^n + \sum_{u=\tau-B_n}^{\tau-B_n+b} \Lambda_u^n L \leq \Phi_{\tau-1}^n + \Lambda_\tau^{na}(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))L + \sum_{c=1}^{C} x_\tau^{nc} d_\tau^{nc} + \sum_{u=\tau+1}^{\tau+b}\left(\widetilde{\Lambda}_u^{na}(1-\delta(\widetilde{\Lambda}_u^{na},\widetilde{\mu}_u))L + \sum_{c=1}^{C}\widetilde{y}_u^{nc}\widetilde{d}_u^{nc}\right). \quad (6)$$

In addition, at the start of τ, Δ_τ^n is obtained as:

$$\Delta_\tau^n = \Delta_{\tau-B_n-1}^n + \sum_{u=\tau-B_n}^{\tau}(\Lambda_u^n L). \quad (7)$$

At the beginning of τ, the number of tasks of application n scheduled in τ is $\Lambda_\tau^{na}(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))L$. Besides, at the beginning of τ, the expected number of tasks of application n scheduled in $\tau+b$ $(1 \leq b \leq B_n)$ is $\left(\widetilde{\Lambda}_u^{na}(1-\delta(\widetilde{\Lambda}_u^{na},\widetilde{\mu}_u^n))L\right)$. Thus, in τ, the expected number of application n's tasks executed before $\tau+B_n$ passes is calculated as:

$$\Psi_{\tau+B_n}^n = \Psi_{\tau-1}^n + \Lambda_\tau^{na}(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))L + \sum_{c=1}^{C} x_\tau^{nc} d_\tau^{nc} + \sum_{u=\tau+1}^{\tau+B_n}\left(\widetilde{\Lambda}_u^{na}(1-\delta(\widetilde{\Lambda}_u^{na},\widetilde{\mu}_u))L + \sum_{c=1}^{C}\widetilde{y}_u^{nc}\widetilde{d}_u^{nc}\right). \quad (8)$$

Thus, the conservation of application n's tasks requires that Δ_τ^n should equal $\Phi_{\tau+B_n}^n$, i.e., $\Delta_\tau^n = \Phi_{\tau+B_n}^n$. Then,

$$\Delta_{\tau-B_n-1}^n + \sum_{u=\tau-B_n}^{\tau} \Lambda_u^n L = \Phi_{\tau-1}^n + \Lambda_\tau^{na}(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))L + \sum_{c=1}^{C} x_\tau^{nc} d_\tau^{nc} + \sum_{u=\tau+1}^{\tau+B_n}\left(\widetilde{\Lambda}_u^{na}(1-\delta(\widetilde{\Lambda}_u^{na},\widetilde{\mu}_u))L + \sum_{c=1}^{C}\widetilde{y}_u^{nc}\widetilde{d}_u^{nc}\right). \quad (9)$$

Therefore, constraints (5), (6), and (9) ensure that delay constraints of all tasks are strictly met.

B. Power Consumption Model

Next, we introduce a power consumption model adopted in GDCDC. Similar to [15], we assume that servers for application n are homogeneous in physical configuration. Therefore, the energy consumed by each server for the same application is identical. The number of application n's tasks executed by its switched-on server per minute is denoted by σ_n. In addition, m_τ^n denotes the number of such servers in τ. The service rate of servers for application n in τ is denoted by μ_τ^n, which is obtained application n.

$$\mu_\tau^n = \sigma_n m_\tau^n. \quad (10)$$

The total number of application n's servers is denoted by Ω_n. Let $\daleth = max_{n\in\{1,2,\cdots,N\}} B_n$. $m_{\tau+b}^n$ denotes the total number of switched-on servers for application n in $\tau+b$ and it cannot exceed Ω_n,

$$m_{\tau+b}^n \leq \Omega_n, 0 \leq b \leq \daleth. \quad (11)$$

The total energy consumption of GDCDC is calculated by summing up the energy consumed by servers and facilities including lighting and cooling. Power Usage Effectiveness (PUE) [19] is a critical metric to measure energy efficiency of a data center and it is the ratio of the total energy consumption of GDCDC to the total energy consumed by servers. PUE of GCDC, denoted by γ, is 1.2–2.0 for many data centers [20]. Let $\bar{P}^n$ and $\hat{P}^n$ denote the average idle and peak power of each server for application n, respectively. Besides, its CPU utilization is denoted by u_τ^n. Therefore, based on [20], the total power consumed by GCDC in τ is obtained as:

$$P_\tau = \sum_{n=1}^{N}\left(m_\tau^n(\bar{P}^n + (\gamma-1)\hat{P}^n + (\hat{P}^n - \bar{P}^n)u_\tau^n)\right). \quad (12)$$

The number of tasks that application n's switched-on server executes in τ is calculated as:

$$\frac{L\big(1-\delta(\Lambda_\tau^{na},\mu_\tau^n)\big)\Lambda_\tau^{na}}{m_\tau^n} \tag{13}$$

where $\delta(\Lambda_\tau^{na},\mu_\tau^n)$ denotes the task loss possibility.

The busy time of each switched-on server for application n is $\frac{L(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))\Lambda_\tau^{na}}{\sigma_n m_\tau^n}$ minutes. u_τ^n is obtained as:

$$u_\tau^n=\frac{(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))\Lambda_\tau^{na}}{\sigma_n m_\tau^n}. \tag{14}$$

Let E_τ denote the total energy consumption of GCDC in τ. Based on (10), (12), and (14), it is obtained as:

$$E_\tau=\sum_{n=1}^{N}\left(\frac{g_n\mu_\tau^n+h_n\Lambda_\tau^{na}(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))}{\sigma_n}L\right) \tag{15}$$

where

$$g_n \triangleq \bar{P}^n+(\gamma-1)\hat{P}^n \tag{16}$$

$$h_n \triangleq \hat{P}^n-\bar{P}^n. \tag{17}$$

The available amount of energy in GCDC is denoted by ϖ. Therefore, the total energy consumed in τ or $\tau+b$ ($1\leq b\leq\daleth$) satisfies:

$$\sum_{n=1}^{N}\left(\frac{g_n\mu_\tau^n+h_n\Lambda_\tau^{na}(1-\delta(\Lambda_\tau^{na},\mu_\tau^n))}{\sigma_n}L\right)\leq\varpi \tag{18}$$

$$\sum_{n=1}^{N}\left(\frac{g_n\widetilde{\mu}_{\tau+b}^n+h_n\widetilde{\Lambda}_{\tau+b}^{na}(1-\delta(\widetilde{\Lambda}_{\tau+b}^{na},\widetilde{\mu}_{\tau+b}^n))}{\sigma_n}L\right)\leq\varpi, 1\leq b\leq\daleth. \tag{19}$$

C. Green Energy Model

This work incorporates two types of green energy sources including wind and solar. The use of green energy can decrease the power grid energy consumption of GDCDC and further alleviate its harmful effect to environment. Similar to [11], the length of an interval is small enough and therefore we assume that the wind and solar energy stay the same within each interval and only vary from interval to interval. The amount of energy produced by solar panels in τ is denoted by E_τ^s. The rate at which solar radiation is converted into electricity is denoted by ψ. Besides, SI_τ denotes the solar irradiance in τ. Let κ denote the active irradiation area of solar panels. Based on [21], we have:

$$E_\tau^s=\kappa\psi(SI_\tau)L. \tag{20}$$

Let E_τ^w denote the amount of wind energy generated by wind turbines in τ. The conversion rate of wind to electricity is denoted by η. The on-site air density is denoted by ρ. Besides, ζ and ν_τ denote the rotor area of wind turbines and the wind speed. According to [22], we have:

$$E_\tau^w=\frac{1}{2}\eta\rho\zeta(\nu_\tau)^3L. \tag{21}$$

D. Constrained Optimization Problem

Typically, SLA is signed between GDCDC and cloud users, and it guarantees the performance for each application's tasks. The revenue corresponding to the execution of application n's tasks in τ and $\tau+b$ are denoted by $\eth_\tau^n$ and $\widetilde{\eth}_{\tau+b}^n$, respectively. Let θ_τ^n denote the payment of each task of application n executed in τ. Then, we have:

$$\eth_\tau^n=\left((1-\delta(\Lambda_\tau^{na},\mu_\tau^n))\Lambda_\tau^{na}+\sum_{c=1}^{C}(x_\tau^{nc}d_\tau^{nc})\right)\theta_\tau^n. \tag{22}$$

Let f_1 denote the total revenue of GDCDC brought by tasks of all applications executed in slots τ to $\tau+b$. Then,

$$f_1=\sum_{n=1}^{N}\left(\eth_\tau^n+\sum_{b=1}^{B_n}\widetilde{\eth}_{\tau+b}^n\right). \tag{23}$$

Besides, prices of power grid in τ and $\tau+b$ are denoted by p_τ and $\widetilde{p}_{\tau+b}$, respectively. Let f_2 denote the total cost of GDCDC. f_2 includes two parts that are the grid energy cost of GCDC and the execution cost of VMs in public clouds. Thus, the amount of grid energy of GDCDC in τ is $\mathbf{max}(E_\tau-E_\tau^s-E_\tau^w,0)$. In addition, the amount of grid energy of GDCDC in $\tau+b$ is $\mathbf{max}\left(\widetilde{E}_{\tau+b}-\widetilde{E}_{\tau+b}^s-\widetilde{E}_{\tau+b}^w,0\right)$. Besides, if application n's tasks are executed in public cloud c in $\tau+b$, $\widetilde{y}_{\tau+b}^{nc}$=1; otherwise, $\widetilde{y}_{\tau+b}^{nc}$=0. The number of application n's tasks executed in public cloud c in $\tau+b$ is denoted by $\widetilde{d}_{\tau+b}^{nc}$. The prices of VMs for application n in public cloud c in τ and $\tau+b$ are denoted by r_τ^n and $\widetilde{p}_{\tau+b}^{nc}$, respectively. The average execution time of tasks of application n executed in public cloud c in τ and $\tau+b$ is denoted by r_τ^{nc} and $\widetilde{r}_{\tau+b}^n$, respectively. Hence, the execution cost of VMs in public clouds in τ is $\sum_{n=1}^{N}\sum_{c=1}^{C}(x_\tau^{nc}p_\tau^{nc}r_\tau^n d_\tau^{nc})$. Similarly, the execution cost of VMs in public clouds in $\tau+b$ is $\sum_{n=1}^{N}\sum_{b=1}^{B_n}\left(\sum_{c=1}^{C}(\widetilde{y}_{\tau+b}^{nc}\widetilde{p}_{\tau+b}^{nc}\widetilde{r}_{\tau+b}^n\widetilde{d}_{\tau+b}^{nc})\right)$. Then, we have:

$$\begin{aligned}f_2=&\left(p_\tau\left(\mathbf{max}(E_\tau-E_\tau^s-E_\tau^w,0)\right)+\sum_{n=1}^{N}\sum_{c=1}^{C}(x_\tau^{nc}p_\tau^{nc}r_\tau^n d_\tau^{nc})\right)\\&+\sum_{b=1}^{\daleth}\left(\widetilde{p}_{\tau+b}\left(\mathbf{max}\left(\widetilde{E}_{\tau+b}-\widetilde{E}_{\tau+b}^s-\widetilde{E}_{\tau+b}^w,0\right)\right)\right)\\&+\sum_{n=1}^{N}\sum_{b=1}^{B_n}\left(\sum_{c=1}^{C}(\widetilde{y}_{\tau+b}^{nc}\widetilde{p}_{\tau+b}^{nc}\widetilde{r}_{\tau+b}^n\widetilde{d}_{\tau+b}^{nc})\right).\end{aligned} \tag{24}$$

Then, the GDCDC's profit maximization problem, marked as $\mathbf{P}_1$, is obtained as:

$$\mathbf{Max}\ f_1-f_2$$

s.t.

$$\sum_{c=1}^{C} x_{\tau}^{nc} \leq 1, \sum_{c=1}^{C} \widetilde{y}_{\tau+b}^{nc} \leq 1, 1 \leq b \leq B_n \tag{25}$$

$$x_{\tau}^{nc}, \widetilde{y}_{\tau+b}^{nc} \in \{0,1\}, 1 \leq b \leq B_n \tag{26}$$

$$\mu_{\tau}^{n} \geq 0, \widetilde{\mu}_{\tau+b}^{n} \geq 0, d_{\tau}^{nc} \geq 0, \widetilde{d}_{\tau+b}^{nc} \geq 0, 1 \leq b \leq B_n \tag{27}$$

$$\widetilde{\mu}_{\tau+b}^{n} = 0, \widetilde{d}_{\tau+b}^{nc} = 0, \widetilde{y}_{\tau+b}^{nc} = 0, B_n < b \leq \urcorner \tag{28}$$

The $\mathbf{P}_1$ should be subjected to the above constraints, where the ranges of delay constraints are given in (5), (6) and (9), constraint (11) denotes the total number of switched-on servers for application n in $\tau+b$, constraints (18) and (19) ensure the total energy consumed in τ or $\tau+b$, and the ranges of decision variables are given in constraints (25), (26), (27) and (28). It is also assumed that time-related parameters, e.g., r_{τ}^{n} and p_{τ}, can be well obtained in advance with existing prediction methods at the beginning of τ. The method to solve $\mathbf{P}_1$ is presented in Section IV and its final solution determines an optimal task schedule among GCDC and public clouds that maximizes the profit of GDCDC such that delay constraints of all tasks are strictly met.

IV. Solution Algorithm

A. Temporal Task Scheduling (TTS)

In $\mathbf{P}_1$, the objective function is nonlinear with respect to decision variables. Decision variables y_{τ}^{nc} and $\widetilde{y}_{\tau+b}^{nc}$ are integer variables while μ_{τ}^{n}, $\widetilde{\mu}_{\tau+b}^{n}$, d_{τ}^{nc}, and $\widetilde{d}_{\tau+b}^{nc}$ ($1 \leq b \leq \urcorner$, $1 \leq n \leq N$, $1 \leq c \leq C$) are continuous. Hence, $\mathbf{P}_1$ is a constrained MINLP. This work adopts a penalty function method [14] to convert $\mathbf{P}_1$ into an unconstrained problem $\mathbf{P}_2$. The vector of all decision variables is denoted by $\vec{\hbar}$ and it consists of μ_{τ}^{n}, $\widetilde{\mu}_{\tau+b}^{n}$, y_{τ}^{nc}, $\widetilde{y}_{\tau+b}^{nc}$, d_{τ}^{nc}, $\widetilde{d}_{\tau+b}^{nc}$ ($1 \leq b \leq \urcorner$, $1 \leq n \leq N$, $1 \leq c \leq C$). Thus,

$$\underset{\vec{\hbar}}{\mathbf{Min}} \left\{ \hat{f} = \varsigma \varepsilon - (f_1 - f_2) \right\}. \tag{29}$$

where $\hat{f}$ denotes the augmented objective function, ς is a large positive constant, and ε is the penalty of all constraints.

Algorithm 1 TTS

```
1: Set $\Lambda_{\tau}^{n}(\urcorner - B_n \leq \tau \leq \urcorner - 1)$, $\Delta_{\urcorner - B_n - 1}^{n}$ and $\Phi_{\urcorner - 1}^{n}$ to 0
2: Set $\Lambda_{\tau}^{na}$ and $\Lambda_{\tau}^{nr}$ ($\urcorner \leq \tau \leq N_I$) to $\Lambda_{\tau}^{n}$
3: $\tau \leftarrow \urcorner$
4: while $\tau \leq \beth$ do
5:     Update $\Lambda_{\tau}^{na}$ based on (3).
6:     Solve $\mathbf{P}_2$ with GSPSO
7:     Execute $(\Lambda_{\tau}^{na}(1 - \delta(\Lambda_{\tau}^{na}, \mu_{\tau}^{n}))L)$ tasks in GCDC, and
       schedule $y_{\tau}^{nc} d_{\tau}^{nc}$ tasks to public cloud $c$
8:     Update $\Lambda_{\tau}^{na}$ and $\Lambda_{i}^{nr}$ $(\tau - B_n \leq i \leq \tau)$
9:     $\Phi_{\tau}^{n} \leftarrow \Phi_{\tau-1}^{n} + \sum_{c=1}^{C} x_{\tau}^{nc} d_{\tau}^{nc} + (\Lambda_{\tau}^{na}(1 - \delta(\Lambda_{\tau}^{na}, \mu_{\tau}^{n}))L)$
10:    $\Delta_{\tau - B_n}^{n} \leftarrow \Delta_{\tau - B_n - 1}^{n} + \Lambda_{\tau - B_n}^{n} L$
11:    $\tau \leftarrow \tau + 1$
12: end while
```

The pseudo code of TTS is shown and explained in Algorithm 1. Line 1 initializes Λ_{τ}^{n} $(\urcorner - B_n \leq \tau \leq \urcorner - 1)$, $\Phi_{\urcorner - 1}^{n}$ and $\Delta_{\urcorner - B_n - 1}^{n}$ with 0. Line 2 initializes Λ_{τ}^{nr} and Λ_{τ}^{na} $(\urcorner \leq \tau \leq N_I)$ with Λ_{τ}^{n}. Let $\beth$ denote the total number of intervals. Λ_{τ}^{na} is updated based on (3) in Line 5. Line 6 solves $\mathbf{P}_2$ with GSPSO to determine μ_{τ}^{n}, y_{τ}^{nc} and d_{τ}^{nc}. Line 7 executes $(\Lambda_{\tau}^{na}(1 - \delta(\Lambda_{\tau}^{na}, \mu_{\tau}^{n}))L)$ tasks in GCDC, and schedules $y_{\tau}^{nc} d_{\tau}^{nc}$ tasks to public cloud c. Then, Lines 9–10 update Φ_{τ}^{n} and $\Delta_{\tau - B_n}^{n}$.

B. Genetic Simulated-annealing-based Particle Swarm Optimization (GSPSO)

Note that $\mathbf{P}_2$ is an unconstrained MINLP. Its pseudos are described in Algorithm 2. The lower and upper bounds of inertia weight w are $\underline{w}$ and $\overline{w}$, respectively. The temperature cooling rate is denoted by ℓ. t^0 denotes the initial temperature. The range of each particle's velocity is limited to $[-\overline{v}, \overline{v}]$. The number of iterations is denoted by I. $\bar{\varrho}$ denotes the predefined percentage threshold. ϱ denotes the percentage of particles with identical fitness values in the swarm.

Algorithm 2 GSPSO

```
1: Initialize positions and velocities of PSO's particles
2: Calculate fitness values of particles
3: Update $g$ and $l_i$
4: $i \leftarrow 0$
5: while $\varrho \leq \bar{\varrho}$ and $i \leq I$ do
6:     Conduct GA's crossover, mutation, and selection on $g$
       and $l_i$
7:     Update velocities and positions of particles based on
       SA's Metropolis acceptance rule
8:     Update $g$ and $l_i$
9:     Decrease temperature by $\ell$
10:    Decrease $w$ linearly from $\overline{w}$ to $\underline{w}$
11:    Update $\varrho$
12:    $i \leftarrow i + 1$
13: end while
14: Return $g$
```

In Algorithm 2, positions and velocities of particles in PSO are randomly initialized in Line 1. g and l_i are updated in Line 3. The **while** loop ends if $\varrho \leq \bar{\varrho}$ and I is achieved. Line 6 conducts the crossover, mutation, and selection of GA on g and l_i to generate superior particles. Line 7 updates positions and velocities of particles based on SA's Metropolis acceptance rule. Line 8 updates g and l_i. Line 9 decreases the temperature by ℓ. Line 10 linearly decreases w from $\overline{w}$ to $\underline{w}$. Line 11 updates ϱ. At last, g is returned and transformed into decision variables including μ_{τ}^{n}, $\widetilde{\mu}_{\tau+b}^{n}$, y_{τ}^{nc}, $\widetilde{y}_{\tau+b}^{nc}$, d_{τ}^{nc}, $\widetilde{d}_{\tau+b}^{nc}$ ($1 \leq b \leq \urcorner$, $1 \leq n \leq N$, $1 \leq c \leq C$).

V. Performance evaluation

This section evaluates TTS with real-life data, e.g., VM prices, price of power grid, arriving tasks, and green energy data. TTS is implemented with MATLAB 2017 and it is executed on a server with an Intel Xeon CPU at 2.4 GHz and a 32-GB DDR4 memory.

TABLE I
RANGES OF VM PRICES ($/HOUR)

Public clouds	Small	Large	Xlarge
1	[0.07,0.08]	[0.06,0.07]	[0.18,0.20]
2	[0.14,0.16]	[0.12,0.14]	[0.10,0.12]
3	[0.22,0.24]	[0.20,0.22]	[0.05,0.06]

A. *Parameter Setting*

We choose realistic Google task data of three applications [1] in 24 hours on May 10, 2011. We set delay constraints to 3, 4, and 5 intervals, i.e., B_1=3, B_2=4, and B_3=5. Besides, we choose real-life price of power grid in 24 hours on the same day in New York, U.S.[2]. According to [20], the parameters in the power consumption model are set as: $\bar{P}^1$=200(W), $\bar{P}^2$=100(W), $\bar{P}^3$=50(W), $\hat{P}^1$=400(W), $\hat{P}^2$=200(W), $\hat{P}^3$=100(W), ϖ=5(MWH), σ_1=0.05 tasks/minute, σ_2=0.1 tasks/minute, σ_3=0.2 tasks/minute, and γ=1.2. Based on [3], Ω_1=3×10^6, Ω_2=1.5×10^6, Ω_3=3×10^6, C_1=12, C_2=25, and C_3=50.

We assume that a single task finishes its execution in GDCDC in each interval. Thus, we randomly produce tasks' execution time for each application based on the uniform distribution in the range of (0, L). Based on [12], we randomly produce the prices ($/hour) of tasks executed in GDCDC based on the uniform distribution in the ranges of [0.24, 0.48], [0.16, 0.32], and [0.08, 0.16], respectively. In this way, θ_τ^n is obtained. Besides, we adopt the data of wind speed[3] and solar irradiance[4] in 24 hours on the same day. According to [11], κ=1.5∗10^5(m^2), ψ=0.2, η=0.3, ρ=1.225(kg/m^3), and ζ=2.5∗10^5(m^2). The power ratings of wind and solar energy are set to 9∗10^8 (W) and 1.65∗10^8 (W), respectively. Based on [5], GSPSO's parameters in Algorithm 2 are set as: $\overline{w}$=0.95, $\underline{w}$=0.4, $\bar{\varrho}$=95%, ξ=100, φ=200, c_1=c_2=0.5, ℓ=0.975, t^0=10^{30}, ς=10^{20} and α=β=2.

Based on the pricing model in Amazon EC2[5], we adopt three types of VM instances (Small, Large, and Xlarge) in public clouds. This work chooses two most typical types of resources including CPU and memory to describe VMs because these are the most important configurations for selecting a VM instance in public clouds. Table I presents the price ranges of three VM instances in public clouds.

B. *Experimental Results*

To demonstrate the performance of GSPSO, GSPSO is compared to PSO and SA. We first give the reasons of choosing them for comparison. It is proven that SA can finally find global optima in theory by carefully setting the temperature cooling rate due to the fact that it can conditionally escape from local optima. Thus, the comparison between them demonstrates the accuracy of GSPSO's final solution. In addition, it is shown that PSO's convergence speed is quick. The comparison between them demonstrates GSPSO's convergence speed.

[1]https://github.com/google/cluster-data
[2]http://www.nyiso.com/public/index.jsp
[3]http://www.nrel.gov/midc/nwtc_m2/
[4]http://www.nrel.gov/midc/srrl_bms/
[5]https://aws.amazon.com/cn/ec2/

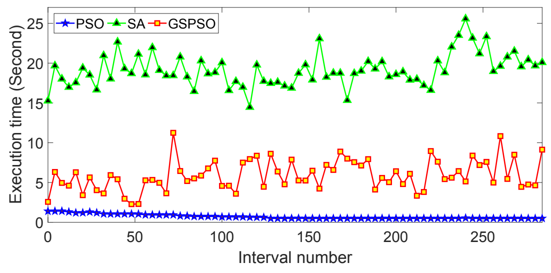


Fig. 2. Comparison of execution time.

The execution time comparison of GSPSO, PSO, and SA is shown in Fig. 2. The average execution time of SA is 19.13s and it is larger than that of PSO, 0.70s, and that of GSPSO, 5.91s. PSOs execution time the least because of its quick trap into locally optimal solutions. The profit comparison of GSPSO, PSO, and SA in each iteration of the 50th interval is presented in Fig. 3. Here the iteration of GSPSO is shown in Lines 6–12 in Algorithm 2. The meaning of iterations in SA and PSO are similar to that of GSPSO. The evolutionary curve of penalty calculated is shown in Fig. 4.

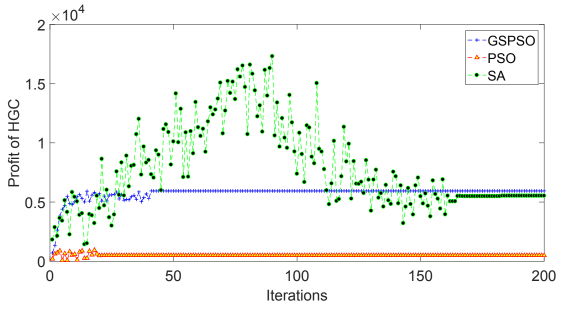
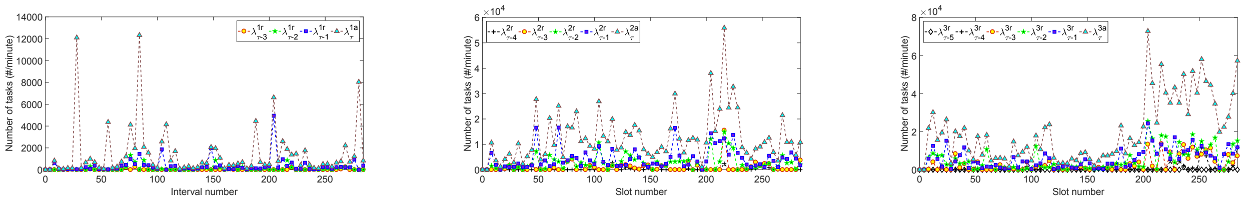


Fig. 3. Profit of each iteration in the 50th interval.

It is shown that PSO converges and loses its search ability after the least number of iterations. However, it is shown in Fig. 4 that the penalty of the final solution of PSO is large (about 4×10^4). This shows that PSO's final solution cannot meet all constraints in $\mathbf{P}_1$. Therefore, the final solution of PSO is the worst. Besides, SA converges to its final solution after about 165 iterations. The profit of its final solution is 11.11 times more than that of PSO but less than that of GSPSO. GSPSO converges to its final solution after 41 iterations and its profit is 5941.06$. Therefore, GSPSO's profit is increased by 414.42$ in much fewer iterations and much less time than SA. In addition, it is shown in Fig. 4 that the penalty of the final solution of GSPSO is nearly 0. This result demonstrates that GSPSO converges to a high-quality solution satisfying all constraints in $\mathbf{P}_1$. Therefore, Figs. 2–4 show that the incorporation of superior particles produced by



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

Fig. 5. Cumulative and remaining tasks

SA's Metropolis acceptance rule and GA's genetic operations in PSO increases the quality of GSPSO's final solution and leads to larger profit for GDCDC.

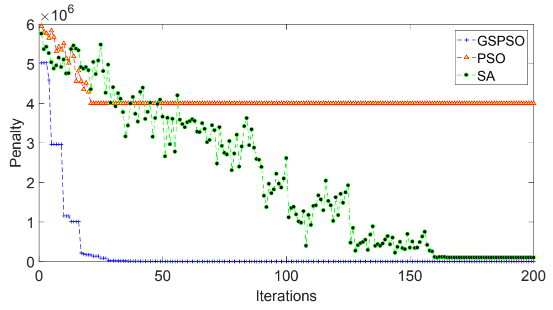


Fig. 4. Penalty of each iteration.

Fig. 5 illustrates the cumulative arriving and remaining tasks of three applications. For example, Fig. 5(b) presents them for application 2 where Λ^{2a}_{τ}=Λ^{2}_{τ} + $\Lambda^{2r}_{\tau-1}$+$\Lambda^{2r}_{\tau-2}$+$\Lambda^{2r}_{\tau-3}$+$\Lambda^{2r}_{\tau-4}$. It is observed that $\Lambda^{2r}_{\tau-4}$ is least among $\Lambda^{2r}_{\tau-4}$, $\Lambda^{2r}_{\tau-3}$, $\Lambda^{2r}_{\tau-2}$ and $\Lambda^{2r}_{\tau-1}$. The reason is that TTS puts all arriving tasks of each application into its seperate FCFS queues, and prefers to schedule earlier-arrived tasks. TTS aims to guarantee that by interval τ, all tasks of application 2 in interval $\tau-4$ or before must have been scheduled to GCDC and public clouds before interval τ passes. Tasks that arrive in intervals $\tau-3$, $\tau-2$ and $\tau-1$ can only be executed when $\Lambda^{2r}_{\tau-4}$=0. Similarly, tasks that arrive in intervals $\tau-2$ and $\tau-1$ can only be executed when $\Lambda^{2r}_{\tau-4}$=0 and $\Lambda^{2r}_{\tau-3}$=0. Tasks that arrive in interval $\tau-1$ are executed when all tasks that arrive before have been executed, i.e., $\Lambda^{2r}_{\tau-4}$=0, $\Lambda^{2r}_{\tau-3}$=0 and $\Lambda^{2r}_{\tau-2}$=0. Therefore, it is observed that $\Lambda^{2r}_{\tau-1}$ is larger than $\Lambda^{2r}_{\tau-4}$, $\Lambda^{2r}_{\tau-3}$ and $\Lambda^{2r}_{\tau-2}$ in each interval.

The Cumulative Scheduled Tasks (CSTs) and Cumulative Arriving Tasks (CATs) for each application are shown in Fig. 6. It is shown that tasks of all applications are scheduled to GCDC and public clouds within their delay constraints. For example, the number of CATs of application 2 in interval 143 equals that of CSTs in interval 140. This means that tasks of application 2 arriving in interval 140 or before are all scheduled to GCDC and public clouds before interval 143 passes. Therefore, it demonstrates that TTS strictly meets delay constraints of tasks of all applications.

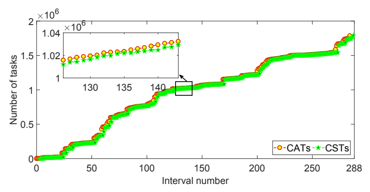
Fig. 7 shows CSTs of each application in GCDC and public clouds. It is shown that the number of tasks executed in GCDC is much larger than that of any public cloud. The reason is that GCDC aims to schedule all in the cost-effective way. Therefore, TTS tries to execute tasks in GCDC, thus maximizing its profit. In addition, the number of tasks executed in different public clouds shows the difference of prices of VMs. For example, it is shown in Fig. 7(b) that the number of application 1 tasks executed in public cloud 1 is much larger than those of public clouds 2 and 3 in each interval. The reason is that the prices of VMs in public cloud 1 is smaller than those of public clouds 2 and 3. As another example, the number of application 3 tasks executed in public cloud 3 is much larger than those of public clouds 1 and 2 in each interval. The reason is that the prices of VMs in public cloud 3 is smaller than those of public clouds 1 and 2 in each interval. Thus, the numbers of tasks of each type executed in three public clouds is the reflection of the variation in prices of their VMs. It also demonstrates that GDCDC's profit can be maximized by smartly scheduling tasks among GCDC and public clouds.

VI. Conclusion and Future Work

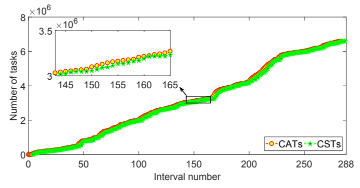
An increasing number of companies deploy their delay-constrained applications in Green Cloud Data Centers (GCD-Cs). The unprecedented growth of tasks significantly increases the energy consumption and therefore a hybrid cloud scheme is growingly chosen to tackle aperiodicity and uncertainty in tasks. The temporal variation in price of power grid, revenue, solar irradiance, wind speed, and price of public clouds brings a big challenge to cost-effectively execute all tasks among the GCDC and public clouds while strictly satisfying all tasks' delay constraints. This work presents a Temporal Task Scheduling (TTS) algorithm that investigates the temporal variation. It can smartly schedule all tasks to the GCDC and public clouds within delay constraints. Beside, the mathematical relation between task refusal and service rates are explicitly presented. Then, the profit maximization problem is solved with a novel hybrid optimization algorithm. Extensive simulation experiments demonstrate that TTS outperforms several existing scheduling algorithms in terms of throughput and profit. In the future, we plan to evaluate the indeterminacy in green energy and to predict the arrivals of tasks with deep neural network algorithms.

Acknowledgment

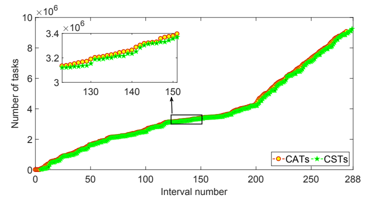
We thank all the reviewers for their insightful comments. This work was supported in part by the National Natu-



(a) Type 1

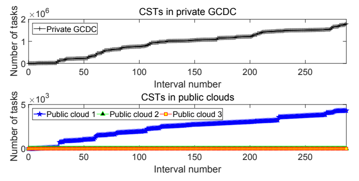


(b) Type 2

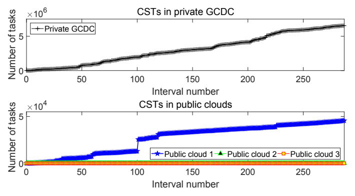


(c) Type 3

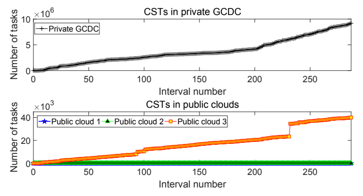
Fig. 6. CATs and CSTs



(a) Type 1



(b) Type 2



(c) Type 3

Fig. 7. CSTs in GCDC and public clouds.

ral Science Foundation of China under Grant 61703011, the National Science and Technology Major Project under Grant 2018ZX07111005, and the National Defense Foundation Research Common Project under Grants 41401020401 and 41401050102.

References

[1] D. A. Chekired and L. Khoukhi, "Smart grid solution for charging and discharging services based on cloud computing scheduling," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 6, pp. 3312–3321, Dec. 2017.

[2] F. Yao, J. Wu, S. Subramaniam, and G. Venkataramani, "WASP: Workload adaptive energy-latency optimization in server farms using server low-power states," in *Proc. 2017 IEEE 10th International Conference on Cloud Computing*, 2017, pp. 75–84.

[3] 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, no. 1, pp. 57–72, Sep. 2016.

[4] Y. Niu, F. Liu, X. Fei, and B. Li, "Handling flash deals with soft guarantee in hybrid cloud," in *Proc. 2017 IEEE Conference on Computer Communications*, 2017, pp. 1–9.

[5] 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.

[6] K. Konstanteli, T. Cucinotta, K. Psychas, and T. A. Varvarigou, "Elastic admission control for federated cloud services," *IEEE Transactions on Cloud Computing*, vol. 2, no. 3, pp. 348–361, Jul. 2014.

[7] L. Suresh, M. Canini, S. Schmid, and A. Feldmann, "C3: Cutting tail latency in cloud data stores via adaptive replica selection," in *Proc. 12th USENIX Conference on Networked Systems Design and Implementation*, 2015, pp. 513–527.

[8] S. B. Sulistyo, W. L. Woo, and S. S. Dlay, "Regularized neural networks fusion and genetic algorithm based on-field nitrogen status estimation of wheat plants," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 103–114, Feb. 2017.

[9] S. Lyden and M. E. Haque, "A simulated annealing global maximum power point tracking approach for pv modules under partial shading conditions," *IEEE Transactions on Power Electronics*, vol. 31, no. 6, pp. 4171–4181, Jun. 2016.

[10] M. Mahi, mer Kaan Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Applied Soft Computing*, vol. 29, no. 4, pp. 484–490, Jun. 2015.

[11] 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.

[12] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564–573, Apr. 2014.

[13] J. Bi, H. Yuan, W. Tan, M. Zhou, Y. Fan, J. Zhang, and J. Li, "Application-aware dynamic fine-grained resource provisioning in a virtualized cloud data center," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1172–1184, Apr. 2017.

[14] M. Shokrian and K. A. High, "Application of a multi objective multi-leader particle swarm optimization algorithm on NLP and MINLP problems," *Computers & Chemical Engineering*, vol. 60, no. 1, pp. 57–75, Jan. 2014.

[15] H. Yuan, J. Bi, W. Tan, and B. Li, "CAWSAC: Cost-aware workload scheduling and admission control for distributed cloud data centers," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 976–985, Apr. 2016.

[16] Z. Zhu, J. Bi, H. Yuan, and Y. Chen, "SLA based dynamic virtualized resources provisioning for shared cloud data centers," in *Proc. 2011 IEEE International Conference on Cloud Computing*, 2011, pp. 630–637.

[17] B. Du, W. Xiong, J. Wu, L. Zhang, L. Zhang, and D. Tao, "Stacked convolutional denoising auto-encoders for feature representation," *IEEE Transactions on Cybernetics*, vol. 47, no. 4, pp. 1017–1027, Apr. 2017.

[18] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. R. Ḿuller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2660–2673, Nov. 2017.

[19] A. N. Toosi, K. Vanmechelen, F. Khodadadi, and R. Buyya, "An auction mechanism for cloud spot markets," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 11, no. 1, pp. 1–33, Apr. 2016.

[20] M. Ghamkhari and H. Mohsenian-Rad, "Energy and performance management of green data centers: A profit maximization approach," *IEEE Transactions on Smart Grid*, vol. 4, no. 2, pp. 1017–1025, Jun. 2013.

[21] N. Mahdavi, J. H. Braslavsky, M. M. Seron, and S. R. West, "Model predictive control of distributed air-conditioning loads to compensate fluctuations in solar power," *IEEE Transactions on Smart Grid*, vol. 8, no. 6, pp. 3055–3065, Nov. 2017.

[22] H. M. K. Al-Masri, A. A. Almehizia, and M. Ehsani, "Accurate wind turbine annual energy computation by advanced modeling," *IEEE Transactions on Industry Applications*, vol. 53, no. 3, pp. 1761–1768, May 2017.